PHP マニュアル

Mehdi Achour
Friedhelm Betz
Antony Dovgal
Nuno Lopes
Hannes Magnusson
Georg Richter
Damien Seguy
Jakub Vrana
その他
2007/12/11
Philip Olson
PHP マニュアル翻訳プロジェクト

序文

PHP は、"PHP: Hypertext Preprocessor" を意味し、広く使用されているオープンソースの汎用スクリプト言語です。 HTML に埋め込むことができ、Web アプリケーションの開発に特に適しています。 PHP の構文の多くは C、Java、Perl 言語から転用したもので、 簡単に習得することができます。この言語は、動的に生成されるウェブページを Web 開発者が速やかに作成できるようにすることを主な目標として つくられました。しかし、それだけにとどまらず、 もっと多くのことを PHP を使って行うことができます。

このマニュアルは、 関数リファレンス を中心として、言語リファレンス、 PHP の主な機能、 そして、その他の 付録 から構成されています。

本マニュアルを様々な形式で » http://www.php.net/download-docs.php からダウンロードすることが出来ます。 このマニュアルがどのように作成されているか、といった詳細な情報は 付録の 本マニュアルについて にあります。 PHP の歴史 に興味がある場合は、 関連する付録も参照して下さい。

著者と貢献者

我々は、現在最も活動的な人々をこのマニュアルの先頭に掲げています。しかし、他にも多くの我々の活動を支援してくれる人がおり、また、過去にこのプロジェクトを支援してくれた人がいます。マニュアルのユーザ注記により支援してくれた名前がわからない人々がいますが、彼らの支援は非常に有益です。以下に掲げたリストはアルファベット順です。

著者と編集者

以下の方々は本マニュアルについてコンテンツを提供したことで大きな貢献をしています。 Bill Abt, Jouni Ahto, Alexander Aulbach, Daniel Beckham, Stig Bakken, Jesus M. Castagnetto, Ron Chmara, Sean Coates, John Coggeshall, Simone Cortesi, Markus Fischer, Wez Furlong, Sara Golemon, Rui Hirokawa, Brad House, Pierre-Alain Joye, Etienne Kneuss, Moriyoshi Koizumi, Rasmus Lerdorf, Andrew Lindeman, Stanislav Malyshev, Rafael Martinez, Rick McGuire, Yasuo Ohgaki, Derick Rethans, Rob Richards, Sander Roobol, Egon Schmid, Thomas Schoefbeck, Sascha Schumann, Dan Scott, Masahiro Takagi, Michael Wallner, Lars Torben Wilson, Jim Winstead, Jeroen van Wolffelaar そして Andrei Zmievski.

以下の方々は本マニュアルについて多くの編集作業を行ったことで大きな貢献をしています。 Stig Bakken, Gabor Hojtsy, Hartmut Holzgraefe そし

て Egon Schmid.

ユーザ注記の管理者

現在最もアクティブな管理者。 Mehdi Achour, Etienne Kneuss, Nuno Lopes, Hannes Magnusson, Bobby Matthis そして Maciek Sokolewicz.

以下の方々もユーザ注記の管理に注力してくれました。 Daniel Beckham, Friedhelm Betz, Victor Boivie, Jesus M. Castagnetto, Nicolas Chaillan, Ron Chmara, Sean Coates, James Cox, Vincent Gevers, Sara Golemon, Zak Greant, Szabolcs Heilig, Oliver Hinckel, Hartmut Holzgraefe, Rasmus Lerdorf, Matthew Li, Andrew Lindeman, Aidan Lister, Maxim Maletsky, James Moore, Philip Olson, Sebastian Picklum, Derick Rethans, Sander Roobol, Damien Seguy, Jason Sheets, Tom Sommer, Jani Taskinen, Yasuo Ohgaki, Jakub Vrana, Lars Torben Wilson, Jim Winstead, Jared Wyles そして Jeroen van Wolffelaar.

翻訳者

本マニュアルの日本語への翻訳は、PHP マニュアル翻訳プロジェクトにて行われています。 主な翻訳者は、以下の通りです（名前のアルファベット順）。

Chihiro Higuchi, Haruki Setoyama, Hideyuki Shimooka, Kazuhiko Ogura, Machino Satoshi, Masaharu Iwai, Masahiro Takagi, Masaki Fujimoto, Michihide Hotta, Moriyoshi Koizumi, Rui Hirokawa, Shigeru Kanemoto, Tadashi Jokagi, Yasuo Ohgaki, Yu Watanabe, Yusuke Hata.

はじめに

目次

入門

目次

PHP とはなんでしょう?

PHP ("PHP: Hypertext Preprocessor" を再帰的に略したものです) は、広く使われているオープンソースの汎用スクリプト言語です。 PHP は、特に Web 開発に適しており、HTML に埋め込むことができます。

この答は簡単ですが、実際どういう意味でしょう? 以下に例を示します。

Example#1 初歩的な例

```
<html>
     <head>
          <title>Example</title>
     </head>
     <body>

         <?php
         echo "Hi, I'm a PHP script!";
         ?>

      </body>
</html>
```

PerlやCのような他の言語で書かれたスクリプトとの違いは、 HTMLを出力するために多くのコマンドを記述する代わりに、 何かを行う(この場合はテキストを出力する)コードを埋め込んだ HTMLスクリプトを書くということです。PHPのコードは、"PHPモード" に入っ たり出たりする特別な 開始および終了のタグで括られています。

PHP がクライアントサイド JavaScript のようなものと異なっている点は、 コードがサーバーで実行されるということです。上のようなスクリプトをサーバー上においていたとしたら、クライアントは、スクリプトを実行し た結果を受け取りますが、その出力を作成したコードに関する情報を得る ことはできません。全てのHTMLファイルをPHPで処理するようにWebサーバー を設定することさえ可能で、この場合、ユーザーが袖の内に何があるかを 見分けることは不可能になることでしょう。

PHPを使用する上で最も優れている点は、初心者に対しては非常に分かり 易いと同時に、プロフェッショナルのプログラマに対しては多くの進んだ 機

能を提供している点です。PHPの機能を羅列した長い一覧表を読まなけ ればならないのかと心配する必要はありません。PHPはすぐに始められま すし、数時間の内に簡単なスクリプトが書けるようになります。

PHPを使用した開発ではサーバサイドでの動作に焦点が当てられますが、 他にも多くのことが可能です。 PHPにできることは?まで読み進めてみてください。 Webプログラミングのみに関心がある場合には、 簡易チュートリアルに進んでください。

PHPにできることは?

あらゆることができます。PHPでは主にサーバサイドでの活用に焦点が 当てられているため、フォームからデータを取得したり、動的にページ の内容を生成したり、クッキーを送信・受信するといった他のCGI プログラムに出来ることは全て行うことが出来ます。しかし、これが 全てではありません。

PHPスクリプトが使用される場所は主に3つあります。

- サーバサイドでのスクリプティング。これは最も古くからあり PHPの中心となる分野です。ここでPHPを動作させるには 3つのものが必要です。PHPパーサ(CGIもしくはサーバモジュール)、 ウェブサーバ、そしてブラウザです。ウェブサーバは インストールされたPHPと連結して起動されなければなりません。 ブラウザでウェブサーバにアクセスし、PHPページを閲覧することで PHPプログラムの出力を得ることが出来ます。 インストール手順の章に詳しい 情報があります。
- コマンドラインでのスクリプティング。PHPスクリプトは サーバもブラウザも無しで動作させるようにすることも出来ます。 この場合、PHP パーサだけが必要となります。このタイプは cron(Windowsではタスクスケジューラ)を使用して一定間隔で スクリプトを実行したい場合や、ちょっとした文書処理を 行うのに最適な方法です。 コマンドラインでPHPを使うの章に詳しい情報があります。
- クライアントサイドでのGUIアプリケーション。PHPはおそらく ウィンドウを使用したアプリケーションを書くのに最適な言語 では無いと思いますが、PHPに非常に慣れている場合には、そして PHPの進んだ機能をクライアントサイドでのアプリケーションで 使用した場合にはPHP-GTKを使ってプログラムを書くことが出来ます。 同様の方法でクロスプラットフォーム名アプリケーションを書く ことも出来ます。PHP-GTKはPHPを拡張するもので、通常の ディストリビューションには含まれません。もし興味があるなら » PHP-GTKのサイトを 訪れてみてください。

PHPは Linux, 多くのUnix系システム(HP-UX, Solaris, OpenBSD等), Microsoft Windows, Mac OS X, RISC OS, その他全ての有名なOSで 動作します。PHPはまた現在使用されているほとんど全てのウェブサーバを サポートします。これには、Apache, Microsoft Internet Information Server, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd等 が含まれます。そうしたウェブサーバの大部分に対してPHPはモジュールを 提供し、その他のものに対してはCGIが提供されます。

つまりPHPを使用する場合にはOSとウェブサーバを自由に選ぶことが出来ます。 さらに手続き型のプログラミングかオブジェクト指向のプログラミングか、 もしくはそれらを混在させるかといった選択を行うこともできます。 現在のバージョンのPHPでは標準的なOOPの機能が全て実現されている訳ではありませんが、(PEARライブラリを含め)多くのコードライブラリ や大きなアプリケーションがOOPのみを使用して書かれています。

PHPはHTMLを出力するだけではありません。PHPはイメージやPDFファイル、 そして(libswfやMingを使って)Flashムービーまでもをその場で生成する 機能を備えています。またXHTMLやXMLといったその他の文書も自動的に 生成することが出来ますし、ファイルシステムに保存したり、印刷したり サーバ側でキャッシュすることも出来ます。

PHP の機能の中で最も強力で優れた機能は、広範なデータベースをサポー トしていることでしょう。データベース機能を用いたWebページの作成は、非常に簡単です。以下のデータベースが現在サポートされています。

- Adabas D
- dBase
- Empress
- FilePro (読込みのみ)
- Hyperwave
- IBM DB2
- Informix
- Ingres
- InterBase
- FrontBase
- mSQL
- Direct MS-SQL
- MySQL
- ODBC
- Oracle (OCI7 および OCI8)
- Ovrimos
- PostgreSQL
- SQLite
- Solid
- Sybase
- Velocis
- Unix dbm

また、データベース抽象化モジュールもあります (PDO といいます)。これは、 このモジュールでサポートされているデータベースに透過的にアクセス

する機能を提供します。 加えて、PHP は ODBC (Open Datebase Connection) をサポートするので、 この世界標準の機構をサポートするどんなデータベースにもアクセスすることが出来ます。

PHP は、IMAP、SNMP、NNTP、POP3、HTTP、COM (Windowsのみ) やその他 数え切れない程多くのプロトコルを用いる他のサービスの状態を追跡する 機能もサポートしています。低レベルのネットワークソケットをオープンし、 他のプロトコルを用いて通信を行うことも可能です。また、PHPはWDDXを サポートし、基本的に全てのウェブプログラミング言語間で複雑なデータ交換 を行うことができます。相互接続機能としては、他にJavaオブジェクトの インスタンスを作成してそれをPHPのオブジェクトとして透過的にアクセス する機能や、CORBA拡張モジュールを使用してリモートオブジェクトにアクセス する機能があります。

PHPにはPOSIX拡張正規表現もしくはPerl正規表現からXML文書の解析に至るまで 非常に便利なテキスト処理の機能があります。XML文書の解析や操作のために SAXとDOMをサポートしています。XML文書の変換にはXSLT拡張モジュールを 使用することが出来ます。

他にも多くの興味深い拡張モジュールがあります。mnoGoSearch サーチエンジン関数、IRC ゲートウェイ関数、多くの圧縮ユーティリティ (gzip, bz2, zip)、カレンダー関数、翻訳関数などなど……。

お分かりの通り、このページではPHPの機能やPHPを使用することの利点を全て 紹介することは出来ません。PHPのインストール の章を読んでみてください。紹介された拡張モジュールに関しては関数リファレンスを読んでみてください。

簡易チュートリアル

目次

ここで、PHP の基礎の基礎について簡単なチュートリアルで説明したいと思います。 PHP は Web ページを作成する機能だけを有しているわけではありませんが、 ここでは PHP で動的な Web ページを作成することのみを扱います。詳細は、 PHP でできること と題するセクションを参照してください。

PHP を使用できる Web ページは、通常の HTML ページと全く同様に扱われ、 通常の HTML ページを作成するのと同様の方法で編集することができます。

必要なものは?

本チュートリアルでは、使用するサーバで PHP が使用可能であり、 *.php* で終わる全てのファイルが PHP で処理されることを仮定します。多くのサーバでは、PHP ファイルに関してこれがデフォルトの拡張子ですが、 確実なのはサーバの管理者にきいてみることです。サーバが PHP をサポートする場合、何もする必要はありません。 *.php* ファイルを 作成して Web ディレクトリに置くだけで、 サーバがこれを自動的にパースしてくれます。 何もコンパイルする必要はなく、他のツールをインストールする必要もありません。 PHP のファイルは、あなたが行なう全ての処理を実装した特殊なタグを通常の HTML ファイルに追加したものと考えると良いでしょう。 ほとんどの Web ホストは PHP サポートを提供していますが、 使用しているホストがサポートしていない場合、 » PHP リンク集のセクションで PHP が利用可能な Web ホストを探すためのリソースを読んでみてください。

ここでは、貴重なネットワーク帯域を節約するために、 ローカルに開発を行うことにしましょう。この場合、 » Apache のような Web サーバと、 当然、» PHP をインストールすることになります。また、多くの場合には、 » MySQL のようなデータベースもインストールすることになるでしょう。

これらは個別にインストールすることもできますし、 より簡単な方法でインストールすることも可能です。 このマニュアルには、 PHP のインストール手順 (Web サーバが設定済みであると仮定しています) があります。 PHP 自体をインストールする際に問題が発生した場合、 » インストールに関するメーリングリストで質問することをお薦めします。 より簡単にインストールを行いたい場合には、 使用するオペレーティングシステム用の » 設定済みのパッケージ を利用することもできます。これにより、 数回のマウスクリックで自動的にこれらをインストールすることができます。 MacOSX、Linux や Windows を含む、あらゆるオペレーティングシステムにおいて Web サーバで PHP を使用できるように設定することは簡単です。 Linux の場合、RPM の場所を知るために » rpmfind と » PBone が有用でしょう。 Debian 用パッケージをみつけるには、» apt-get にアクセスするとよいでしょう。

PHP を使用する初めてのページ

以下の内容で *hello.php* という名前のファイルを作成し、Web サーバのルートディレクトリ (*DOCUMENT_ROOT*) に置いてください。

Example#1 初めての PHP スクリプト: *hello.php*

```
<html>
  <head>
    <title>PHP  Test</title>
  </head>
```

```
  <body>
 <?php  echo  '<p>Hello  World</p>'; ?>
  </body>
</html>
```

ブラウザを使用して、"/hello.php" で終わる Web アクセス用 URL を指定し、 このファイルにアクセスしてください。 ローカルに開発を行っている場合、この URL は *http://localhost/hello.php* または *http://127.0.0.1/hello.php* のようになります。 しかし、これは Web サーバの設定に依存します。 全てが正しく設定されている場合、 このファイルは PHP によりパースされ、以下の出力がブラウザに送信されます。

```
<html>
  <head>
    <title>PHP  Test</title>
  </head>
  <body>
  Hello  World<p>
  </body>
</html>
```

このプログラムは非常に簡単なので、実際には、このようなページを 作成するために PHP を使用する必要はありません。 *Hello World* を PHP の echo() 命令により出力しているだけです。 このファイルは、実行ファイルまたは特殊なファイルとする必要がないことに注意してください。 このファイルが拡張子 ".php" を有し、このファイルが PHP に渡される必要があると設定されているため、サーバは PHP により解釈されるファイルを見付けることができます。 このファイルは、多くの面白いことを可能にする特別なタグを利用できる、 通常の HTML ファイルと考えることができます。

この例を試しても何も出力されない場合、または、 ダウンロード用のプロンプトが表示されるか、 テキストとしてファイル全体が表示された場合、 利用しているサーバで PHP が利用できない可能性があります。本マニュアルの インストール の章により PHP を利用できるようにするよう管理者にきいてみてください。 ローカルに開発を行っている場合も、 インストールの章を読んで設定が全て正しく行われていることを確認してください。 解決しない問題がある場合は、多くの » PHP サポート の選択肢のどれかを利用してみてください。

この例の目的は、特殊な PHP タグ形式を示すことです。 この例では、*<?php* が PHP タグの開始を示しています。この後、PHP 命令を置き、終了タグ *?>* を記述することにより、PHP モードを抜けています。 このように任意の場所で PHP モードを抜けて HTML ファイルに移ることができます。詳細は、 基本的な構文のセクションを参照ください。

> **注意**: **改行に関する注意** HTML においては改行にはほとんど意味がありません。ただ、 HTML の見栄えをよくするためにも適宜改行を入れておくとよいでしょう。 *?>* の直後の改行は、PHP によって取り除かれます。 複数の PHP ブロックを使用している場合や、 何かを出力するのかどうかがわからないファイルを include する際などに、 この挙動は非常に便利です。と同時に少々混乱するかもしれません。 強制的に改行させるには、*?>* の後に空白を置くか、 あるいは PHP ブロック内の最後の echo/print で明示的に改行を出力します。
>
> **注意**: **テキストエディタに関する注意** PHP ファイルを作成、編集、監理する際に使用できる、 多くのテキストエディタや統合開発環境 (IDE) があります。 これらのツールのリストの一部は、» PHP エディタのリスト で整理されています。あるエディタを推薦したい場合、上記のページを訪れ、 ページの監理者にそのエディタをリストに加えてくれないかとたずねてみてください。
>
> **注意**: **ワードプロセッサに関する注意** StarOffice Writer, Microsoft Word および Abiword のようなワードプロセッサは、PHP ファイルの編集には向いていません。 これらのワープロ上でテストスクリプトを編集する場合は、 ファイルをプレーンテキストとして保存していることを確認してください。 さもないと、PHP はスクリプトを読み込んで実行できません。
>
> **注意**: **Windows のメモ帳に関する注意** Windows のメモ帳を使用して PHP スクリプトを書く場合には、ファイルに 拡張子 .php を付けて保存したかどうかを確認する必要があります (メモ帳は、以下の防止策のどらかを適用しない限り、拡張子 .txt を自動的に付加します)。 ファイルを保存する際に、 ファイル名を入力するプロンプトでファイル名を引用符で括ります (すなわち、"*hello.php*" とします)。 もしくは、保存ダイアログボックスにおいてドロップダウンメニュー "テキスト文書"をクリックし、"すべてのファイル" に設定を変更します。 これにより、引用符を付けずにファイル名を入力することができます。

さて、動作する簡単な PHP スクリプトを作成することができましたので、 最も有名な PHP スクリプトを作成してみましょう! phpinfo() 関数をコールすることにより、 定義済み変数、 ロードされている PHP モジュール、設定 等のシステムに関する多くの有用な情報を得ることができます。 この重要な情報を見てみてください。

Example#2 PHP からシステムに関する情報を取得する

```
<?php  phpinfo(); ?>
```

実用的な例

次に、より実用的なことをしてみましょう。 ページを見ているユーザが使用しているブラウザの種類を確認してみます。 これを行なうには、ブラウザが HTTP リクエストの一部として送信した user agent 文字列を調べます。 この情報は、変数 に保存されています。PHP では、変数名は常にドル記号で始まります。 ここで使用する変数は、*$_SERVER['HTTP_USER_AGENT']* です。

> **注意**: $_SERVER は、 Web サーバ関連情報を全て保持する PHP の特別な予約変数です。詳細は、 スーパーグローバル を参照してください。 これらの特別な変数は、» 4.1.0 で導入されました。これ以前は、 *$HTTP_SERVER_VARS* のような古い配列 *$HTTP_*_VARS* を代わりに使用していました。 古いとはいえ、これらの変数はまだ存在しています (古いコードに関する注記も 参照してください)。

この変数を表示するには、以下のようにします。

Example#1 変数を出力する (配列要素)

```
<?php
echo $_SERVER['HTTP_USER_AGENT'];
?>
```

このスクリプトの出力例は以下のようになります。

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

PHP で利用可能な変数の型 には多くの種類があります。上の例では、 配列 の要素を出力しています。配列は、非常に有用です。

$_SERVER は、PHP で自動的に利用可能な変数のひとつに過ぎません。マニュアルの 定義済の変数 のセクションでリストを参照することができます。 あるいは、完全なリストを取得するには、さきほどのセクションで使用した phpinfo() 関数の出力を確認します。

PHP タグの中に複数の PHP 命令を置くことができ、echo 文以上のことを行なうコードブロックを作成することができます。 例えば、インターネット・エクスプローラかどうかを調べたい場合は、 以下のようにします。

Example#2 制御構造 および 関数の使用例

```
<?php
if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== false) {
  echo 'あなたはInternet Explorerを使用しています<br />';
}
?>
```

このスクリプトの出力例は以下のようになります。

```
あなたはInternet Explorerを使用しています<br />
```

ここで、新しい概念をいくつか導入します。 if 文を使用しています。 C 言語の基本構文を知っているとしたら、理解できると思います。 C 言語や上記の構文を使用する他の言語をあまり知らない場合には、 PHP の入門書を手にとって最初の数章を読むか、このマニュアルの 言語リファレンスの部分を読むべきです。

二番目の新しい概念は、strpos() 関数のコールです。 strpos() は PHP に組み込まれた関数で、 文字列の中である文字列を探します。この場合、*$_SERVER['HTTP_USER_AGENT']* (いわゆる干し草の山 【haystack】) の中で *"MSIE"* (いわゆる針【needle】) を探しています。 この文字列が見つかった場合、 この関数はこの関数は文字列の相対的な位置を返し、 見つからなかった場合には **FALSE** を返します。 この関数が **FALSE** を返さなければ、 if 文は **TRUE** と評価し、その{波括弧}の中のコードが実行されます。 そうでない場合は、実行されません。 if, else と strtoupper() や strlen() のような他の関数で、似たような例を作ってみてください。 関連するマニュアルの各ページにも例がのっています。 関数の使用法に自信がない場合には、マニュアルの 関数定義の読み方および PHP関数のセクションの両方を 読んでみると良いでしょう。

この例を少し発展させて、PHP ブロックの中からでも PHP モードから出たり入ったりすることができることを以下に示します。

Example#3 HTML および PHP モードの両方を混在させる

```
<?php
if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== false) {
?>
<h3>strposが非falseを返しました</h3>
<center><b>あなたはInternet Explorerを使用しています</b></center>
<?php
} else {
?>
<h3>strposがfalseを返しました</h3>
<center><b>あなたはInternet Explorerを使用していません</b></center>
<?php
}
?>
```

この例の出力は以下のようになります。

```
<h3>strposが非falseを返しました</h3>
<center><b>あなたはInternet Explorerを使用しています</b></center>
```

何かを出力する際に PHP の echo 文を使用する代わりに、PHP モードを抜けて通常の HTML を送信しています。ここで注意すべき重要で強力な点は、 スクリプトの論理フローが損なわれないということです。 strpos() が **TRUE** または **FALSE** のどちらを返すか、言い換えると*MSIE* が見つかったかどうかに基づき、HTML ブロックだけが見る側に送信されることになります。

フォームの処理

PHP の最も強力な機能の一つは、HTML フォームを処理する手段です。 理解するべき重要な基本概念は、あるフォームの中の全てのフォーム要素が、自動的に PHP スクリプトで利用可能になるということです。 詳細は、マニュアルのセクション PHPの外部からくる変数 および PHP でフォームを使用する例を参照してください。以下に HTML フォームの例を示します。

Example#1 簡単な HTML フォーム

```
<form  action="action.php"  method="post">
  名前:  <input  type="text"  name="name"  />
  年齢:  <input  type="text"  name="age"  />
  <input  type="submit"  />
</form>
```

このフォームに関して特別なところはありません。これは通常の HTML フォームで特殊なタグは全く使用していません。 ユーザがこのフォームを記入し、投稿ボタンを押した時、 *action.php* ページがコールされます。 このファイルには、以下のようなコードを記述します。

Example#2 フォームからのデータを出力する

```
こんにちは、<?php  echo  htmlspecialchars($_POST['name']);  ?>さん。
あなたは、<?php  echo  (int)$_POST['age'];  ?>  歳です。
```

このスクリプトの出力例は次のようになります。

```
こんにちは、Joe  さん。あなたは、22  歳です。
```

htmlspecialchars() および *(int)* の部分以外は、何を行っているかは明らかでしょう。 htmlspecialchars() は、html での特殊な文字を適切にエンコードし、 HTML タグや Javascript をページ内に仕込めないようにします。 また、age フィールドには数値が入ることがわかっているので、これを integer 型に 変換 します。これにより、おかしな文字が入力されることを防ぎます。 これらの処理を PHP に自動的に行わせるためには、 filter 拡張モジュールを使用します。 変数 *$_POST['name']* と *$_POST['age']* は PHP により自動的に設定されます。 前の部分では、スーパーグローバル *$_SERVER* を使用しましたが、 ここでは、全ての POST データを保持するスーパーグローバル $_POST を導入しています。 フォームのメソッドが POST であることに注意してください。 *GET* メソッドを使用している場合、 フォームの情報は代わりにスーパーグローバル $_GET に代入されます。 リクエストデータの発信源に留意しない場合には、 スーパーグローバル変数 $_REQUEST を使用することもできます。この変数は、GET, POST, COOKIE, FILE データの混ざったものが含まれます。 import_request_variables() 関数も参照してください。

XForms の入力を PHP で扱うことも可能ですが、たいていの場合は HTML フォームのほうが快適に使用できるでしょう。 XForms は初心者向けのものではありませんが、気になるかたもいるかもしれません。 機能概要の節にある XForm から受信したデータの処理方法 を参照ください。

新しいバージョンの PHP で古いコードを使用する

今や PHP は有名なスクリプト言語となってきており、 各自のスクリプトで再利用可能なコードとして、 多くのリソースが公開されています。PHP 言語の開発者の大部分は、 過去のバージョンとの互換性を保とうとしており、 過去のバージョン用に書かれたスクリプトは (理想的には) より新しいバージョンの PHP で変更せずに動作するはずです。 しかし、実際には、通常いくつかの変更が必要となります。

古いコードに影響を与える最近の重要な二つの変更点を以下に示します。

- *$HTTP_*_VARS* 配列が過去のものとなったこと (これは、関数またはメソッドの中で使用する際にグローバル変数として 宣言を行なう必要がありました)。以下の スーパーグローバル 配列が » 4.1.0 で導入されました。これらを以下に示します。 *$_GET*, *$_POST*, *$_COOKIE*, *$_SERVER*, *$_ENV*, *$_REQUEST*, *$_SESSION*. 従来の $HTTP_POST_VARS のような配列 *$HTTP_*_VARS* もまだ存在し、PHP 3 以来維持されています。 PHP 5.0.0 以降、PHP の長い 定義済みの変数 配列は register_long_arrays ディレクティブにより無効にすることができます。
- 外部変数は、もはやデフォルトでグローバルスコープに登録されません。 言い換えると、PHP » 4.2.0 以降、PHP ディレクティブ register_globals は、 *php.ini* においてデフォルトで *off* となっています。 これらの値にアクセスする推奨される方法は、 上記のスーパーグローバル配列を使用する方法です。古いスクリプトや本、 チュートリアルはこのディレクティブに依存している可能性があります。 このディレクティブが on の場合、 例えば、URL *http://www.example.com/foo.php?id=42* から *$id* を使用することができます。 on、off によらず *$_GET['id']* は利用可能です。

これらの変更に関する詳細は、 定義済みの変数 のセクションおよびそこにあるリンクを参照してください。

次にするべきことは?

ここで得た知識により、マニュアルのほとんどの部分、 そしてサンプルのアーカイブにある多くのスクリプトの例を理解できるようになったはずです。他の例を Web サイト php.net にあるリンクセクション » http://www.php.net/links.php で探すこともできます。

この他 PHP でできる多くことを示すスライドプレゼンテーションを見るには、 PHP カンファレンスマテリアルサイト、 » http://conf.php.net/ と » http://talks.php.net/ を参照してください。

インストールと設定

目次

インストールにあたっての一般的な注意事項

インストールを行う前に、PHP を使用する用途を明確にしておく必要があります。 PHP を適用可能な分野としては、 「PHP にできることは？」 のセクションに記述されている通り、主に次の 3 つがあります。

- Web サイトや Web アプリケーション (サーバサイドのスクリプト)
- コマンドラインのスクリプト
- デスクトップ (GUI) アプリケーション

最初の用途がもっとも一般的で、この場合、PHP 本体、Web サーバ、Web ブラウザ の 3 つが必要となります。Web ブラウザは既にお持ちだと思います。 使用しているオペレーティングシステムのセットアップの状況によっては、 さらに Web サーバも稼働しているかもしれません（例、Linux 上の Apache や Windows 上の IIS）。また、ホスティング会社で Web 用のスペースを 借りることもできるでしょう。この場合は、自分自身でセットアップを行う必要はなく、 PHP スクリプトを作成し、借りているサーバにアップロードするだけで、 ブラウザを使って処理結果を見ることができます。

一方、Web サーバと PHP を自分でセットアップする場合、 サーバに PHP を組み込む方法が 2 種類あります。 多くのサーバに対して、各サーバ独自のモジュールインターフェイス (SAPI とも呼ばれます) を通じて、ダイレクトに PHP を動作させることができます。 Apache、Microsoft Internet Information Server、 Netscape、iPlanet サーバなどがサポートされています。 ISAPI と呼ばれるマイクロソフト互換のモジュールインターフェイスを 持つ Web サーバ（OmniHTTPd など）もサポートされます。 PHP がモジュールのサポートをしていない Web サーバに対しては、 CGI もしくは FastCGI プロセサとして PHP を使用することができます。 つまり、PHP ファイルへのリクエストの処理を、 PHP のコマンドライン版の実行ファイルを使って行うよう Web サーバを設定することができます。

コマンドラインでのスクリプト実行に PHP を使用する （たとえば、オフラインで画像を自動生成するスクリプトを書いたり、 指定した引数に応じてテキストファイルを処理したりといった） 場合は、コマンドライン版の実行ファイルが必要となります。 詳細な情報については、 「PHP をコマンドラインから使用する」 の章を参照してください。この場合、サーバとブラウザは不要です。

PHP-GTK 拡張モジュールを使って、PHP でクライアントサイドの GUI アプリケーションを作成することも可能です。この場合のアプローチは Web ページの 作成とは完全に異なり、HTML を出力するのではなく、 ウインドウやその中のオブジェクトの管理を行うことになります。PHP-GTK に関する より詳細な情報については、 » PHP-GTK 拡張モジュールのサイト を参照してください。PHP-GTK は、PHP の公式アーカイブには含まれていません。

以降、この節では Unix や Windows 上の Web サーバに サーバモジュールインターフェイスおよび CGI 実行ファイルとして PHP をセットアップする方法を説明します。 コマンドラインの実行ファイルについての情報も、 これ以降の節で得られるでしょう。

PHP のソースコードと Windows 用のバイナリアーカイブは、 » http://www.php.net/ にあります。 アーカイブをダウンロードする際には、最も近い » ミラーサイト を使用するようにしてください。

Unix システムへのインストール

目次

本章では、UNIX 系のシステムへの PHP のインストールと設定に関する 手引きを示します。使用するプラットフォームや Web サーバについての セクションを参照して、インストールを行ってください。

このマニュアルでは、 「インストールにあたっての一般的な注意事項」 の章で述べたように、Web 用のセットアップを主に扱います。加えて、 コマンドラインから PHP を使うためのセットアップについても扱います。

Unix プラットホームに PHP をインストールする方法はいくつかあり、 コンパイルして設定するやり方と（コンパイル済みの）パッケージを使う方法とに 別けられます。この手引きでは、コンパイルして設定する方法を主に取り上げます。 Unix 系システムには、パッケージを用いるインストールシステムを持つものも多く、 一般的なセットアップを行うには、パッケージが役に立つでしょう。ただし、 （セキュアサーバや様々なデータベースドライバなど）少々特殊な機能が必要な場合、 PHP や Web サーバをビルドする必要が有るかもしれません。ソフトウエアの ビルドに不慣れな場合は、必要な機能を含めてビルドされたパッケージを 誰か他の人が作成済みでないかを調べてみると良いでしょう。

コンパイルにあたって必要な知識とソフトウエアを以下に示します。

- UNIX に関する基本的な知識 ("make" および C コンパイラを使える程度)
- ANSI C コンパイラ
- flex: バージョン 2.5.4
- bison: バージョン 1.28 (推奨), 1.35, または 1.75
- Web サーバ
- (gd、pdf ライブラリ等の) モジュール用のコンポーネント

PHP の初期設定および設定プロセスは、**configure** スクリプトに与えられたコマンドラインオプションによりコントロールされます。 **./configure --help** とすると、 オプションの一覧と簡単な解説が表示されます。 本マニュアルでは、オプションの種類ごとに別けて解説されています。 PHP 本体のオプションの一覧は 付録 にまとめられています。 各拡張モジュール特有のオプションは、関数リファレンスのページに記述されています。

PHP の configure が完了していないと、 拡張モジュールや本体の実行ファイルのビルドができません。 **make** コマンドの実行にあたっては、注意してください。 configure がうまく行かず原因もよくわからない場合は、 問題が起きた場合 についての章を参照してください。

Apache 1.3.x (Unix システム用)

このセクションでは、PHP を Unix プラットフォームの Apache 1.3.x にインストールする際の 手引きと注意事項について説明します。Apache 2 に関する手引きと注意 は 別のセクションにあります。

以下の説明では、バージョン番号が意図的に省略されています。 'xxx' の部分を使用するファイルに対応する番号に置き換えてください。 また、手順 10 で **configure** に与える引数は、 configure のすべてのオプション から選択できます。

Example#1 PHP インストール 手順 (Apache 共有モジュール版)

```
1.   gunzip  apache_xxx.tar.gz
2.   tar  -xvf  apache_xxx.tar
3.   gunzip  php-xxx.tar.gz
4.   tar  -xvf  php-xxx.tar
5.   cd  apache_xxx
6.    ./configure  --prefix=/www  --enable-module=so
7.   make
8.   make  install
9.   cd  ../php-xxx

10. PHP の configure を行います。ここでは、様々なオプションを指定して、
    特定の拡張モジュールを有効にするといった、カスタマイズを行います。
     指定可能なオプションの一覧は、./configure --help を実行すると得られ
    ます。以下に、簡単な設定例を示します。Apache 1 と MySQL のサポートを
    有効にする例です。apxs のパスは、Apache のインストールパスによって
    異なる場合があります。

         ./configure  --with-mysql  --with-apxs=/www/bin/apxs

11.  make
12.  make  install

     configure オプションを変更して再インストールする場合は、最後の 3 つ
    の手順を繰り返します。共有モジュールとしてコンパイルされた PHP を
     有効にするには Apache を再起動するだけです。Apache の再コンパイルは
    必要ありません。

     特に指定がない限り、'make install' は、PEAR, phpize のような様々な
    関連ツール、CLI 版 PHP などもインストールすることに注意してください。

13. php.ini ファイルをセットアップ

        cp  php.ini-dist  /usr/local/lib/php.ini

    PHP の実行時設定を変更するには、.ini ファイルを編集します。
    このファイルを他の場所に置きたい場合は、手順 10 で、
     オプション --with-config-file-path=/path を使用します。

     php.ini-dist ではなく、php.ini-recommended を使用する場合は、PHP の
    動作が変化しますので、ファイル中に記載されている変更点の一覧を確認する
    ようにしてください。

14. httpd.conf を編集し、PHP の共有モジュールをロードするよう設定します。
     LoadModule 命令の右側に記述するパスは、システムの PHP 共有モジュール
     を指している必要があります。上記の make install により既にこの設定は
    追加されている場合もありますが、確認が必要です。

    PHP 4 の場合:

        LoadModule  php4_module  libexec/libphp4.so
```

PHP 5 の場合:

```
LoadModule php5_module libexec/libphp5.so
```

15. httpd.conf の AddModule セクションに以下を追加します。
ClearModuleList の下あたりに追加してください。

PHP 4 の場合:

```
AddModule mod_php4.c
```

PHP 5 の場合:

```
AddModule mod_php5.c
```

16. Apache が特定の拡張子のファイルを PHP としてパースするよう（httpd.conf を編集して）設定します。例えば、Apache が拡張子 .php のファイルを PHP としてパースするように設定します。複数の拡張子も、空白で区切って記述するだけで PHP としてパースさせることができます。以下の例は .php と .phtml とを指定した場合です。

```
AddType application/x-httpd-php .php .phtml
```

PHP のソースをハイライト表示させるために、拡張子 .phps を設定することもよく行われます。

```
AddType application/x-httpd-php-source .phps
```

17. Apache サーバを、通常の手順通り、起動させます（HUP または USR1 シグナルを 使用してリロードするのではなく、サーバを停止させてから再起動する必要があります）。

PHP を静的オブジェクトとしてインストールすることも可能です。

Example#2 PHP インストール手順 (Apache 静的モジュール)

```
1.  gunzip -c apache_1.3.x.tar.gz | tar xf -
2.  cd apache_1.3.x
3.  ./configure
4.  cd ..

5.  gunzip -c php-5.x.y.tar.gz | tar xf -
6.  cd php-5.x.y
7.   ./configure --with-mysql --with-apache=../apache_1.3.x
8.  make
9.  make install

10. cd ../apache_1.3.x

11.  ./configure --prefix=/www --activate-module=src/modules/php5/libphp5.a

    (上の行は間違いではありません。この段階で libphp5.a は存在していませんが
    この時点での存在は必須ではなく、後に作成されます。)

12. make

    (httpd バイナリが作成され、Apache バイナリディレクトリにコピーすることが
    できます。最初のインストールの場合は、この後 "make install" を行います。)

13. cd ../php-5.x.y
14. cp php.ini-dist /usr/local/lib/php.ini

15. /usr/local/lib/php.ini を編集すると、PHP の実行時設定を変更できます。
    httpd.conf もしくは srm.conf ファイルを編集し、以下を追記します。
     AddType application/x-httpd-php .php
```

注意: PHP 4 については、*php-5* を *php-4* へ、 *php5* を *php4* へ置き換えてください。

インストールされている Apache や UNIX の種類によりますが、サーバの停止・再起動の 方法はいくつもあります。いろいろな Apache/UNIX の組合せを想定して、 サーバを再起動する典型的な方法を以下に示します。 */path/to/* を使用するシステムのアプリケーション へのパスに置き換えてください。

Example#3 Apache を再起動するためのコマンドの例

```
1. Linux および System V 系
/etc/rc.d/init.d/httpd  restart

2. apachectl スクリプトを使用する方法
/path/to/apachectl  stop
/path/to/apachectl  start

3. (OpenSSL を使用している場合) httpdctl および httpsdctl を使用する方法
/path/to/httpsdctl  stop
/path/to/httpsdctl  start

4. mod_ssl や他の SSL サーバを使用している場合、手動で stop や start する
/path/to/apachectl  stop
/path/to/apachectl  startssl
```

apachectl および http(s)dctl の実行ファイルの位置は、システムにより 異なります。システムが *locate* もしくは *whereis*、*which* コマンドを サポートしているなら、これらサーバ制御用プログラムを見つけるために使用すると 便利でしょう。

PHP を Apache 用にコンパイルするには、いくつかの方法があります。以下に例を示します。

```
./configure  --with-apxs  --with-pgsql
```

この例では、Apache がロードする共有モジュールのライブラリ *libphp5.so* (あるいは PHP 4 では *libphp4.so*) が作成されます。この共有ライブラリの読み込みは、Apache の設定ファイル *httpd.conf* の LoadModule の行にて設定します。また、このライブラリには、 PostgreSQL サポートが埋め込まれます。

```
./configure  --with-apxs  --with-pgsql=shared
```

この例でも Apache 用 *libphp4.so* 共有ライブラリ が作成されます。加えて、（PHP 拡張モジュールの）共有ライブラリ *pgsql.so* も作成されます。この共有ライブラリ は、PHP 設定ファイル *php.ini* の extension ディレクティブにより、 もしくは PHP スクリプト内で明示的に dl() 関数により ロードされます。

```
./configure  --with-apache=/path/to/apache_source  --with-pgsql
```

この例では、*libmodphp5.a* ライブラリと *mod_php5.c* およびいくつかの付属ファイルが 作成され、Apache のソースツリーのディレクトリ *src/modules/php5* にコピーされます。続いて、 *--activate-module=src/modules/php5/libphp5.a* と指定して Apache をコンパイルしてください。Apache のビルドシステムにより、*libphp5.a* が作成され、 *httpd* バイナリに静的にリンクされます (PHP 4 に対しては、*php5* を *php4* へ 置き換えてください)。PostgreSQL サポートはこの *httpd* バイナリに 直接埋め込まれるため、最終的な結果としては、Apache 全体と PHP 全体を含む 単一の *httpd* バイナリが出来上がります。

```
./configure  --with-apache=/path/to/apache_source  --with-pgsql=shared
```

この例は、上と同様ですが、最終的な *httpd* バイナリに PostgreSQL サポートは直接埋め込まれません。共有ライブラリ *pgsql.so* が作成され、PHP 設定ファイル *php.ini*、もしくは dl() 関数により明示的に PHP にロードすることができます。

PHP のビルド方法を選択する際には、各方法の利点と欠点を考慮する必要 があります。共有モジュールのオブジェクトとしてビルドすると、Apache とは別に コンパイルすることができ、PHP を追加または変更する際に全体を 再コンパイルする必要がありません。PHP を Apache に（静的に）組み込むと、 PHP はより高速にロード・実行されます。詳細な情報については、 Apache の Web ページ 「» 動的共有オブジェクト (DSO) サポート」 を参照してください。

> **注意**: Apache のデフォルトの *httpd.conf* には、次のように記述されたセクションがあります。
>
> ```
> User  nobody
> Group  "#-1"
> ```
>
> これを "Group nogroup" （や "Group daemon"）等に変更しないと、 PHP はファイルをオープンすることができません。
>
> **注意**: --with-apxs=/path/to/apxs オプションを 指定する場合には実際にシステムにインストールされている apxs を指定してください。 Apache のソースディレクトリ内にある apxs を指定してはいけません。

Apache 2.0 (Unixシステム用)

このセクションでは、PHPを Unix システム上の Apache 2.0 にインストールする際の 手引きと注意事項について説明します。

警告

Apache2 の MPM マルチスレッドモードを実運用環境で使用することは推奨されません。代わりに prefork MPM または Apache1 を使用してください。その理由については、 マルチスレッド版 MPM の Apache2の FAQ エントリを参照してください。

» Apache ドキュメンテーション を参照し、Apache 2.0.x の基本的な事項について理解しておくことを 強く推奨します。

> **注意**: **PHP と Apache 2.0.x の互換性に関する注意** PHP の以下のバージョンは、Apache 2.0.x の最新版での動作が確認されています。
>
> - PHP 4.3.0 およびそれ以降 (» http://www.php.net/downloads.php で入手可能)
> - 最新の安定開発版。 ソースコード » http://snaps.php.net/php5-latest.tar.gz を入手、または Windows 用のバイナリ » http://snaps.php.net/win32/php5-win32-latest.zip をダウンロードしてください。
> - プレリリース版を » http://qa.php.net/ から ダウンロード可能です。
> - » anonymous CVS から PHP を 入手することも可能です。
>
> 以上のバージョンの PHPは、Apache 2.0.40 以降と互換性があります。
> Apache 2.0 *SAPI* のサポートは PHP 4.2.0 で開始されました。 PHP 4.2.3 は Apache 2.0.39 で動作します。PHP 4.2.3 を Apache の他のバージョンと 組み合わせて使用しないでください。 PHP 4.3.0 もしくはそれ以降のバージョンの PHP を 最新版の Apache2 と組み合わせて使用することが推奨されます。
> ここで挙げたバージョンの PHP は、Apache 1.3.x でも動作します。

最新バージョンの » Apache 2.0 を ダウンロードし、上述のいずれかのバージョンの PHP を用意してください。 この手引きでは Apache 2.0 で *PHP* を動作させるための 基本的な部分しかカバーしていません。さらに詳しい情報については、» Apache ドキュメンテーション を参照してください。 情

報が古く不正確になってしまうため、以下では詳細なバージョン番号は 記述されていません。'NN' という文字列をご使用のバージョンに適宜置き換えてください。

Example#1 インストール手順 (Apache 2 共有モジュール版)

```
1.   gzip  -d  httpd-2_0_NN.tar.gz
2.   tar  xvf  httpd-2_0_NN.tar
3.   gunzip  php-NN.tar.gz
4.   tar  -xvf  php-NN.tar
5.   cd  httpd-2_0_NN
6.   ./configure  --enable-so
7.   make
8.   make  install

     以上で Apache 2.0.NN が、モジュールの動的ロードとデフォルトの
     MPM（マルチプロセッシングモジュール）である prefork が有効になった
      状態で、/usr/local/apache2 にインストールされます。

     インストールが正常か調べるには、以下のようにします。
          /usr/local/apache2/bin/apachectl  start
     サーバの停止は、以下の通り。
          /usr/local/apache2/bin/apachectl  stop

     引き続き PHP のセットアップを行います。

9.   cd  ../php-NN
10.  PHP の configure を行います。ここでは、様々なオプションを指定し、特定の
     拡張モジュールを有効にするといったカスタマイズを行います。指定可能な
      オプションの一覧は、./configure --help を実行すると得られます。以下に、
     Apache 2 と MySQL のサポートを有効にする、簡単な設定例を示します。

          ./configure  --with-apxs2=/usr/local/apache2/bin/apxs  --with-mysql

11.  make
12.  make  install

     configure オプションを変更して再インストールする場合は、最後の 3 つ
     の手順を繰り返します。共有モジュールとしてコンパイルされた PHP を
      有効にするには Apache を再起動するだけです。Apache の再コンパイルは
     必要ありません。

      特に断りがない限り、'make install' は、PEAR、phpize のような様々な
     関連ツール、CLI 版 PHP などもインストールすることに注意してください。

13.  php.ini ファイルを設定する

         cp  php.ini-dist  /usr/local/lib/php.ini

     PHP の実行時設定を変更するには、.ini ファイルを編集します。
     このファイルを他の場所に置きたい場合は、手順 10 で、
       オプション --with-config-file-path=/path を使用します。

      php.ini-dist ではなく、php.ini-recommended を使用する場合は、PHP の
     動作が変化しますので、ファイル中に記載されている変更点の一覧を確認する
     ようにしてください。

14.  httpd.conf を編集し、PHP の共有モジュールをロードするよう設定します。
      LoadModule 命令の右側に記述するパスは、システムの PHP 共有モジュール
      を指している必要があります。上記の make install により既にこの設定は
     追加されている場合もありますが、確認が必要です。

     PHP 4 の場合:

         LoadModule  php4_module  modules/libphp4.so

     PHP 5 の場合:

         LoadModule  php5_module  modules/libphp5.so

15.  Apache が特定の拡張子のファイルを PHP としてパースするよう（httpd.conf
      を編集して）設定します。例えば、Apache が拡張子 .php のファイルを PHP
     としてパースするように設定します。複数の拡張子も、空白で区切って記述する
      だけで PHP としてパースさせることができます。以下の例は .php と .phtml
     とを指定した場合です。

         AddType  application/x-httpd-php  .php  .phtml

     PHP のソースをハイライト表示させるために、拡張子 .phps を設定することも
     よく行われています。

         AddType  application/x-httpd-php-source  .phps

16.  Apache サーバを、通常の手順通り、起動させます。
       例、/usr/local/apache2/bin/apachectl  start
```

上記の手順で、*SAPI* モジュールとして PHP を Apache 2.0 で動作させることができます。もちろん、Apacheと *PHP* の双方とも、もっと多くの configure オプションを指定することが出来ます。 詳しい情報を得るには、ソースツリーディレクトリで **./configure --help** を実行してください。また、マルチスレッド版の Apache 2.0を ビルドする場合は、デフォルトの MPM である *prefork* を *worker* もしくは *perchild* で 上書きする必要があります。このためには、上記の手順 6 で--with-mpm=worker もしくは--with-mpm=perchild を指定します。マルチスレッド版 については、その動作に関して十分注意してください。 詳細については Apacheドキュメントの » マルチプロセッシングモジュール (MPM) を参照してください。

注意: コンテントネゴシエーションを使用する場合には、 Apache の MultiViews オプションに関するFAQ を参照してください。

注意: マルチスレッド版の Apache をビルドする場合は、システムがスレッドを サポートしている必要があります。また、*PHP* は 実験的なステータスにある Zend Thread Safety (ZTS) でビルドされます。 そのため、使用できない拡張モジュールがあります。デフォルトの *prefork* MPM でのビルドが推奨されます。

Caudium サーバ

PHP 4 は Caudium Web サーバ用 Pike モジュールとしてビルド可能です。PHP 3 で はこの機能はサポートされていないことに注意してください。以下に PHP 4 を Caudium にインストールする手順を示します。

Example#1 Caudium へのインストール手順

```
1.  PHP 4 をインストールする前に Caudium のインストールを確認する

    PHP 4 を正しく動作させるためには、Pike 7.0.268 以降
     が必要です。この例では、Caudium が /opt/caudium/server/ に
    インストールされていることを仮定します。

2.  php-x.y.z ディレクトリ(x.y.z はバージョン番号)に移動する
3.   ./configure  --with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  実行中ならば、Caudium を再起動する
7.  GUI 設定画面にログインし、PHP 4 サポートを追加したい仮想サーバに移動する
8.  Add Module をクリックし、PHP 4 Script Support module を追加する
9.  'PHP 4 interpreter isn't available' と出力される場合は、サーバを
    再起動したかどうか確認する

    PHP4.so に関するエラーを確認するには、
     /opt/caudium/logs/debug/default.1 をチェックしてください。
      caudium/server/lib/[pike-version]/PHP4.so
    が存在することも確認してください。

10. 必要に応じて、PHP Script Support module を設定する
```

もちろん、PHP 4 で利用可能となった多くの PHP 拡張モジュールを有効にして Caudium モジュールをコンパイルすることもできます。 各拡張モジュール特有の設定オプションについてはリファレンスページを 参照してください。

注意: MySQL サポートを有効にして PHP 4 をコンパイルする場合、必ず 通常の MySQL クライアントのコードを使用するよう指定する必要があります。 そうでない場合、MySQL サポートを組み込み済みの Pike と衝突する 可能性があります。これを行うには、--with-mysql オプションにより MySQL インストールディレクトリを指定します。

---

fhttpd サーバ

PHP を fhttpd モジュールとして作成するには、"Build as an fhttpd module?" に対して、"yes" と答えてください (configure の オプション --with-fhttpd=DIR) 。 そして、fhttpd ソースのベースディレクトリを指定してください。 デフォルトディレクトリは、 */usr/local/src/fhttpd* です。 fhttpd を使用している場合には、PHP をモジュールとして作成した方が、 より優れた性能、より高度な制御/リモート実行機能を使用することができます。

注意: PHP4.3.0 をもって、fhttpd サポートは廃止されました。

Sun, iPlanet, Netscape サーバ（Sun Solaris 用）

このセクションでは、Sun Solaris 上の Sun Java System Web Server, Sun ONE Web Server, iPlanet and Netscape server に PHP をインストールする際の 手引きと注意事項について説明します。

PHP 4.3.3 より、NSAPI モジュール を使って 独自エラーページ および ファイル一覧表示ページの生成 が可能です。 Apache 互換の関数も追加されています。 また、これらの Web サーバについての サブリクエストに関する注意 も参照してください。

Netscape Enterprise Server (NES) への PHP のインストールに関しては » http://benoit.noss.free.fr/php/install-php4.html にも情報があります。

Sun JSWS/Sun ONE WS/iPlanet/Netscape Web サーバ用に PHP をビルド するには、--with-nsapi=[DIR] オプションに適切なインストールディレクトリを指定してください。 デフォルトのディレクトリは、通常、*/opt/netscape/suitespot/* です。
/php-xxx-version/sapi/nsapi/nsapi-readme.txt も参照してください。

1. 以下のパッケージを、» http://www.sunfreeware.com/ や他のダウンロードサイトから取得し、 インストールします。

    - *autoconf-2.13*

- *automake-1.4*
- *bison-1_25-sol26-sparc-local*
- *flex-2_5_4a-sol26-sparc-local*
- *gcc-2_95_2-sol26-sparc-local*
- *gzip-1.2.4-sol26-sparc-local*
- *m4-1_4-sol26-sparc-local*
- *make-3_76_1-sol26-sparc-local*
- *mysql-3.23.24-beta* (mysql サポートが必要な場合)
- *perl-5_005_03-sol26-sparc-local*
- *tar-1.13* (GNU tar)

2. パスを適切に設定します (*PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin*)。 そして、**export PATH** とし、パスを有効にします。
3. **gunzip php-x.x.x.tar.gz** (.gz 配布の場合のみ)
4. **tar xvf php-x.x.x.tar**
5. PHP を展開したディレクトリに移動します **cd ../php-x.x.x**
6. 以下のステップを実施します。 */opt/netscape/suitespot/* は netscape サーバがインストールされている場所です。 異なる場合は、適切なパスに変更してください。

```
./configure --with-mysql=/usr/local/mysql ¥
--with-nsapi=/opt/netscape/suitespot/ ¥
--enable-libgcc
```

7. **make** を実行し、その後 **make install** を実行します。

基本インストールを実行したら、適当な readme ファイルを参照してください。 いくつかの追加インストール手順を実行する必要があるかもしれません。

Sun/iPlanet/Netscape の設定手順

まず、共有ライブラリの探索のために、環境変数 LD_LIBRARY_PATH にパスをいくつか追加する必要があります。 Webサーバの開始スクリプトで行うのが最善でしょう。 開始スクリプトは、通常 */path/to/server/https-servername/start* にあります。 また、*/path/to/server/https-servername/config/* にある設定ファイルの編集も必要です。

1. *mime.types*に次の行を追加します (administration server で行えます。)

```
type=magnus-internal/x-httpd-php exts=php
```

2. *magnus.conf* (サーバ>= 6の場合) または *obj.conf* (サーバ< 6の場合) を編集し、 以下の行を追加します。 ここで、shlib はシステムにより異なります。 */opt/netscape/suitespot/bin/libphp4.so* 等となるでしょう。 *mime types init* の後に置いてください。

```
Init fn="load-modules" funcs="php4_init,php4_execute,php4_auth_trans" shlib="/opt/netscape/suitespot/bin/libphp4.so'
Init fn="php4_init" LateInit="yes" errorString="Failed to initialize PHP!" [php_ini="/path/to/php.ini"]
```

(PHP >= 4.3.3) *php_ini* パラメータはオプションですが、 これを指定することにより、Web サーバの設定ファイルがあるフォルダに *php.ini* を置くことが可能になります。

3. *obj.conf* のデフォルトオブジェクトを設定します （バージョン 6 以降の仮想サーバの場合は *vserver.obj.conf*。）

```
<Object name="default">
.
.
.
.#NOTE this next line should happen after all 'ObjectType' and before all 'AddLog' lines
Service fn="php4_execute" type="magnus-internal/x-httpd-php" [inikey=value inikey=value ...]
.
.
</Object>
```

(PHP >= 4.3.3) 追加のパラメータとして、いくつかの特別な *php.ini* 値を 追加することができます。例えば、コンテキスト *php4_execute* に対して *docroot="/path/to/docroot"* を設定するなどです。 また、論理値の場合、0/1 を値として使用してください。 *"On","Off",...* では正しく動作しません。 例えば、*zlib.output_compression="On"* ではなく、 *zlib.output_compression=1* とします。

4. 以下は、（*cgi-bin* ディレクトリ のように） PHP スクリプトだけが置かれるディレクトリを設定したい場合にのみ必要です。

```
<Object name="x-httpd-php">
ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
Service fn=php4_execute [inikey=value inikey=value ...]
</Object>
```

こうしておくと、Administration Server に特定のディレクトリを設定し、 これをスタイル *x-httpd-php* に割り付けることができます。 このディレクトリの中にあるすべてのファイルは PHP スクリプト として実行されます。 これは、ファイルの拡張子を *.html* に変更し、 PHP が使用されている事を隠したい場合に有用です。

5. 認証を設定します。PHP による認証は他の認証と併用することはできません。 すべての認証は、PHP スクリプトに渡されます。 サーバ全体に

対して PHP 認証を設定する場合は、以下の行を追加してください。

```
<Object  name="default">
AuthTrans  fn=php4_auth_trans
.
.
.
</Object>
```

6. 単一のディレクトリでのみ PHP による認証を行う場合は、次の行を追加します。

```
<Object  ppath="d:¥path¥to¥authenticated¥dir¥*">
AuthTrans  fn=php4_auth_trans
</Object>
```

注意: PHP が使用するスタックサイズは Web サーバの設定に依存します。 非常に大きい PHP スクリプトを実行させた際にクラッシュが発生する場合は、 Administration Server でスタックサイズ（"MAGNUS EDITOR"） を大きくすると良いでしょう。

CGI 環境変数と php.ini の変更

Sun JSWS/Sun ONE WS/iPlanet/Netscape がマルチスレッドの Web サーバだという事が PHP スクリプトを書く際に重要になります。すべてのリクエストは同一の （Web サーバ自体の）プロセス空間で実行され、そのプロセス空間は 1 つの環境変数しか 持っていません。PATH_INFO や HTTP_HOST などの CGI 変数を取得する場合、 PHP 3.x で行っていたような古い方法、つまり getenv() 関数を使用する方法や他の同等な方法 （グローバル変数の登録機能、 $_ENV 等）を使うのは正しい方法ではありません。 Web サーバの環境変数をただ単に取得するだけと、 正しい CGI 変数は得られないのです。

注意: なぜ正しくない CGI 変数が登録されているのでしょうか？
それは、Web サーバのプロセスを Administration Server から起動させる際、 Web サーバの起動スクリプトが CGI スクリプトとして実行されるためです。したがって、 起動された Web サーバの環境変数には CGI 変数も含まれることになります。 Administration Server 以外から Web サーバを起動してみればこのことをテストできるでしょう。 ルートユーザでコマンドラインを使って、手動で起動してみると、CGI 変数らしき 環境変数が登録されないことが確認できるでしょう。

PHP 4.x のスクリプトで CGI 変数を取得する場合は、スーパーグローバル *$_SERVER* を用いるのが正しい方法です。また、*$HTTP_HOST* などを使う古いスクリプトを使用する場合は、*php.ini* で *register_globals* をオンにし、変数のパースの順番 (*variables_order*) を変更してください （*"E"* を削除。環境変数を読み込む必要は無いため。）

```
variables_order  =  "GPCS"
register_globals  =  On
```

独自エラーページおよびファイル一覧表示ページ (PHP >= 4.3.3)

PHP を使って、*"404 Not Found"* などに対するエラーページを生成することが できます。オーバーライドしたいすべてのエラーページ対して、以下の行を *obj.conf* 中のオブジェクトに追加してください。

```
Error  fn="php4_execute"  code=XXX  script="/path/to/script.php"  [inikey=value  inikey=value...]
```

ここで、*XXX* は HTTP のエラーコードです。 追加したものと干渉する *Error* ディレクティブは削除してください。 発生するすべてのエラーに対応するページを設定したい場合は、 *code* パラメータを省略してください。スクリプトで HTTP ステータス コードを取得するには、*$_SERVER['ERROR_TYPE']* を使用します。

独自のファイル一覧表示ページを PHP を使って生成することも可能です。 ファイル一覧表示を行う PHP スクリプトを作成し、*obj.conf* の *type="magnus-internal/directory"* の行に 書かれているデフォルトのサービスを以下のように置き換えます。

```
Service  fn="php4_execute"  type="magnus-internal/directory"  script="/path/to/script.php"  [inikey=value  inikey=value...]
```

エラーページ、ファイル一覧表示ページのいずれでも、元の URI および 変換後の URI は、それぞれ、*$_SERVER['PATH_INFO']* および *$_SERVER['PATH_TRANSLATED']* に格納されます。

nsapi_virtual() および サブリクエストに関する注意 (PHP >= 4.3.3)

NSAPI モジュールは、現在、nsapi_virtual() 関数 (エイリアス: virtual()) をサポートしており、 Web サーバへサブリクエストを行い、結果を Web ページへ挿入することができます。 ただし、この関数は NSAPI ライブラリの文書化されていない機能を若干使用しています。 モジュールは、自動的に必要な関数群を探し、可能であればそれらの関数を 使用します。もし使用可能でなければ、nsapi_virtual() 関数は 使用不可となります。

注意: nsapi_virtual() サポートは「実験的」な機能です。

CGI およびコマンドライン

デフォルトでは、PHP は CGI プログラムとしてビルドされます。すなわち、 コマンドラインインタプリタが生成され、 CGI 処理や Web 以外での PHP スクリプトの実行に使用できます。 通常は、PHP のモジュール組込みをサポートしている Web サーバに対しては、 性能面からモジュール版の PHP を選択するべきです。 しかし、CGI 版を使用すると、ページに応じて異なるユーザ ID で PHP を実行することが可能となる利点があります。

警告

CGI としてセットアップすると、サーバは様々な攻撃を受ける可能性があります。これらの攻撃からサーバを守る方法については、 CGI セキュリティ のセクションを参照してください。

PHP4.3.0 において PHP に重要な追加がなされ、 CLI (Command Line Interface) と呼ばれる新しい SAPI が CGI バイナリと同じ名前で存在するようになりました。 configure のオプションにしたがって *{PREFIX}/bin/php* に インストールされます。詳細については、マニュアルの 「PHPをコマンドラインから使用する」を 参照してください。

テスト

PHP を CGI 版のプログラムとしてビルドした場合、**make test** とすることで、ビルドされたバイナリをテストすることが 可能です。常にビルド後のテストを行うことが推奨されます。これにより、 使用するプラットホームにおける PHP の問題を早期に見付けることが可能となり、 後になってその問題に苦しむことがなくなるでしょう。

ベンチマーク

PHP 3 を CGI 版プログラムとしてビルドした場合は、 **make bench** とすることにより、ベンチマークを 実行することができます。セーフモードがデフォルトで on の場合、30 秒以上かかるベンチマークは完了させることができません。これは、 セーフモードでは、set_time_limit() を使用することが できないためです。スクリプト用にこの時間を設定するには、設定 max_execution_time を使用してください。**make bench** は、設定ファイル を無視します。

> **注意**: **make bench** は PHP 3 でのみ利用可能です。

環境変数の使用

いくつかのサーバが提供する環境変数 は、現在の » CGI/1.1 規約 において定義されていません。以下の変数だけがこの規約で定義されています。 *AUTH_TYPE*, *CONTENT_LENGTH*, *CONTENT_TYPE*, *GATEWAY_INTERFACE*, *PATH_INFO*, *PATH_TRANSLATED*, *QUERY_STRING*, *REMOTE_ADDR*, *REMOTE_HOST*, *REMOTE_IDENT*, *REMOTE_USER*, *REQUEST_METHOD*, *SCRIPT_NAME*, *SERVER_NAME*, *SERVER_PORT*, *SERVER_PROTOCOL*, および *SERVER_SOFTWARE* 。 その他の環境変数は、「ベンダー拡張」として取り扱うべきです。

HP-UX へのインストール

このセクションでは、HP-UX へインストールする場合特有の注意とコツについて説明します

PHP を HP-UX にインストールするには、二通りの方法があります。 自分でコンパイルするか、あるいはコンパイル済みのバイナリを使用するかのどちらかです。

公式のコンパイル済みパッケージは、こちらにあります。 » http://software.hp.com/

このセクションをきちんと書き直すまで、PHP (および拡張モジュール) を HP-UX でコンパイルする方法についてのドキュメントをいったん削除します。 当面は、以下のドキュメントを参照ください。 » Building Apache and PHP on HP-UX 11.11

OpenBSD へのインストール

このセクションでは、PHP を » OpenBSD 3.6 に インストールする場合に固有の注意事項とヒントについて説明します。

バイナリパッケージの使用

OpenBSD に PHP をインストールするには、バイナリパッケージを使用することが もっとも簡単で、また推奨される方法です。 コアパッケージは他のモジュールと分けられており、別個に インストールしたり、削除したりすることができます。OpenBSD の CD や FTP サイトから 必要なファイルを見つけることができます。

インストールに必要なメインのパッケージは *php4-core-4.3.8.tgz* です。これには基本エンジン (と gettext と iconv) が含まれています。次に、 *php4-mysql-4.3.8.tgz* や *php4-imap-4.3.8.tgz* のようなモジュールパッケージを 探してください。これらのモジュールを *php.ini* 上で有効/無効にするには **phpxs** コマンドを使用する必要があります。

Example#1 OpenBSD パッケージインストールの例

```
#  pkg_add  php4-core-4.3.8.tgz
#  /usr/local/sbin/phpxs  -s
#  cp  /usr/local/share/doc/php4/php.ini-recommended  /var/www/conf/php.ini
   (add in mysql)
#  pkg_add  php4-mysql-4.3.8.tgz
#  /usr/local/sbin/phpxs  -a  mysql
   (add in imap)
#  pkg_add  php4-imap-4.3.8.tgz
#  /usr/local/sbin/phpxs  -a  imap
   (remove mysql as a test)
#  pkg_delete  php4-mysql-4.3.8
#  /usr/local/sbin/phpxs  -r  mysql
   (install the PEAR libraries)
```

```
# pkg_add php4-pear-4.3.8.tgz
```

OpenBSD のバイナリパッケージに関する詳細は、man ページの » packages(7) を 参照してください。

Ports の使用

» ports ツリー を使って PHP のソースを コンパイルすることもできます。ただし、この方法は OpenBSD に詳しいユーザにのみ 推奨されます。PHP 4 ポートは core, extensions の 2 つのサブディレクトリに 分割されています。extensions ディレクトリはサポートされているすべての PHP モジュールのサブパッケージを生成します。 これらのうちのいずれかのモジュールを生成したくない場合には、 **no_*** FLAVOR を使用してください。例えば、imap モジュール のビルドをスキップするには FLAVOR を **no_imap** に セットします。

一般的な問題点

- Apache は、デフォルトインストールでは、 » chroot(2) jail 内で実行されます。 これにより、PHP は、*/var/www* 以下のファイルにしか アクセスできないように制限されます。そのため、セッションを使用するには、 */var/www/tmp* ディレクトリを作成するか、他のセッションバックエンドを 用いる必要があります。また、データベースのソケットは jail 内に置かれるか、 *localhost* インタフェイスが listen できるようにする必要があります。 ネットワーク関数を使用する場合は、*/etc* 内のファイル、たとえば */etc/resolv.conf* や */etc/services* を、 */var/www/etc* に移動させる必要があります。 OpenBSD の PEAR パッケージは、適切な chroot 内のディレクトリに自動的にインストールされますので、 特に修正は必要ありません。OpenBSD における Apache についての詳細は » OpenBSD FAQ を参照してください。
- OpenBSD 3.6 においては、» gd 拡張モジュールのパッケージは XFree86 のインストールを必要とします。X11 が必要となるいくつかのフォント機能を使わない場合は、 代わりに *php4-gd-4.3.8-no_x11.tgz* パッケージをインストールしてください。

過去のリリース

OpenBSD の過去のリリースは、静的にリンクされた PHP をコンパイルするために FLAVORS システムを使用していました。 この方法でバイナリパッケージを作成することは困難なため、 この方法は現在は使用されていません。 古い安定版の ports ツリーを使用することもできますが、 これらは OpenBSD チームによりサポートされていません。 これに関するコメントがある場合、port の現在の管理者は Anil Madhavapeddy (avsm at openbsd dot org) です。

Solaris へのインストール

このセクションでは、Solaris に PHP をインストールする際の注意とコツを説明します。

必要なソフトウエア

Solaris には、C コンパイラおよび関連するツールがインストールされていない事が しばしばあります。GNU 版の各種ツールが必要となりますが、その理由については こちらの FAQ をお読みください。 必要なソフトウエアは次の通りです。

- gcc (推奨。他のC コンパイラも動作するかもしれません)
- make
- flex
- bison
- m4
- autoconf
- automake
- perl
- gzip
- tar
- GNU sed

加えて、(Oracle や MySQLなどの) 使用する設定に応じた追加ソフトウエアを インストール（おそらくはコンパイルも）する必要があります。

パッケージの使用

pkgadd を使用すれば、必要なコンポーネントのインストール処理を 簡単に行うことができます。

Debian GNU/Linux へのインストール

このセクションでは、» Debian GNU/Linux に PHP をインストールする際の注意事項とヒントについて説明します。

APT の使用

PHP のソースコードをダウンロードし自分でコンパイルすることも可能ですが、 Debian のパッケージシステムを使用することが、もっともシンプルでクリーンな インストール方法です。Linux でのソフトウエアのビルドに不慣れならば、 この方法を取るのが良いでしょう。

最初に決めなければいけないのは、Apache 1.3.x と Apache 2.x のどちらを インストールするかです。それぞれに対応するパッケージは、libapache-mod-php* および libapache2-mod-php* という名前です。 以下では、Apache 1.3.x を使用する場合について説明します。 また、現在のところ、PHP 5 の Debian 公式パッケージはありません。 ですので、PHP 4 のインストールについて解説します。

Debian では、PHP を CGI や CLI として使用することもできます。 それぞれ、php4-cgi と php4-cli という名前が付けられています。 これらが必要な場合は、下記のステップを対応する名前でもって行ってください。 また、特別なパッケージとしては、さらに、php4-pear があります。 これは、**pear** コマンドラインユーティリティ および PEAR の最小限の部分がインストールされるものです。

Debian の安定版のパッケージよりも新しいパッケージが必要な場合や、 Debian の公式レポジトリに含まれない PHP 拡張モジュールが必要な場合は、 » http://www.apt-get.org/ を探してみると 良いかもしれません。たとえば、» Dotdeb などが見つかるでしょう。この非公式レポジトリは » Guillaume Plessis が管理しており、最新版の PHP 4 および PHP 5 の Debian パッケージがアップされています。このリポジトリを使用するには、以下の 2 行を */etc/apt/sources.lists* に追加し、**apt-get update** を実行します。

Example#1 Dotdeb 関連の apt line

```
deb  http://packages.dotdeb.org  stable  all
deb-src  http://packages.dotdeb.org  stable  all
```

パッケージのリストが最新であるかにも注意を払う必要があります。 パッケージの更新をしばらく行っていないのならば、**apt-get update** をまず最初に実行する必要があります。こうすることにより、 Apache と PHP の最新の安定版パッケージを取得することができます。

すべての準備が完了したら、以下の例に従って Apache と PHP をインストールします。

Example#2 Debian への Apache 1.3 のインストール

```
#  apt-get  install  libapache-mod-php4
```

APT により、Apache 1.3 用の PHP 4 モジュールが自動的にインストールされます。 インストール中に Apache を再起動する旨が表示されなかった場合は、手動で 再起動する必要があります。

Example#3 PHP 4 インストール後に Apache を停止・起動させる

```
#  /etc/init.d/apache  stop
#  /etc/init.d/apache  start
```

追加モジュールと適切な設定

上記では、PHP のコアモジュールだけがインストールされています。 これだけでは不足の場合もあるでしょう。また将来、MySQL や cURL, GD といった より多くのモジュールを有効にすることが必要になるかも知れません。

PHP を自らコンパイルする場合は、有効にするモジュールを **configure** コマンドで指定します。APT では、追加のパッケージをインストールします。それら追加パッケージは、'php4-*'（サードパーティのリポジトリから PHP 5 をインストールする場合は 'php5-*'）といった名前が付けられています。

Example#4 追加のパッケージの名前を取得する

```
#  dpkg  -l  'php4-*'
```

上記の出力を見てわかるとおり、(php4-cgi や php4-cli, php4-pear といった 特別なパッケージのほかにも) さまざまなパッケージがあり、インストールが可能です。 詳細に参照して、必要なものを選択してください。選択したモジュールに必要な ライブラリがインストールされていない場合、APT により自動的にインストールされます。

MySQL, cURL および GD サポートを PHP に追加する場合のコマンドは 以下のようになります。

Example#5 MySQL, cURL および GD を PHP にインストールする

```
#  apt-get  install  php4-mysql  php4-curl  php4-gd
```

APT により、（システムで）用いられている *php.ini* (*/etc/php4/apache/php.ini*, */etc/php4/cgi/php.ini* など) に適切な設定が追加されます。

Example#6 MySQL, cURL および GD を有効にする設定

```
extension=mysql.so
extension=curl.so
extension=gd.so
```

追加したモジュールを有効にするには、前述したとおり、Apache を再起動させる必要があります。

よく起きる問題

- PHP スクリプトが実行された結果が出力されず、PHP のソースコードが表示される場合は、 APT によって、*/etc/apache/conf.d/php4* が読み込まれるように、 Apache 1.3 の設定ファイルが変更されるべきところが、 実際には変更されていないものと思われます。 以下の行を */etc/apache/httpd.conf* に追加し、 Apache を再起動させてください。

  Example#7 PHP 4 を Apache で有効にする

  ```
  #  Include  /etc/apache/conf.d/
  ```

- 追加の拡張モジュールをインストールしたにもかかわらず、 その機能がスクリプトから利用できない場合は、前述した適切な設定が *php.ini* に追加されているかどうか確認してください。debconf による設定で コンフリクトが起こると、追加モジュールのインストールに失敗する場合があります。

Mac OS X へのインストール

目次

本章では、PHP を Mac OS X 上にインストールする際の注意事項とコツを説明します。 クライアント版とサーバ版という、わずかに異なる 2 種類のバージョンの Mac OS X が 存在しますが、本マニュアルでは、両方のシステムへのインストールについて扱います。 PHP は、MacOS 9 およびそれ以前のバージョンでは使用できませんので、注意してください。

パッケージの使用

Mac OS X 用にコンパイルされた PHP パッケージがいくつか存在します。 一般的なセットアップを行うにあたって、利用することができます。ただし、 （セキュアサーバや様々なデータベースドライバなど）少々特殊な機能が必要な場合、 PHP を自分でビルドする必要が有るかもしれません。ソフトウエアの ビルドやコンパイルに不慣れな場合は、必要な機能を含めてビルドされたパッケージを 誰か他の人が作成済みでないかを調べてみると良いでしょう。

以下の場所に、簡単にインストールできる Mac OS 用のコンパイル済み PHP パッケージがあります。

- MacPorts: » http://www.macports.org/
- Entropy: » http://www.entropy.ch/software/macosx/php/
- Fink: » http://fink.sourceforge.net/

バンドルされている PHP の使用法

PHP は、OS X バージョン 10.0.0 以降の Mac に標準添付されています。 デフォルトのウェブサーバで PHP を有効にするには、Apache 設定ファイル *httpd.conf* で数行のコメントを解除する必要があります。 一方、CGI や CLI はデフォルトで有効になっています (ターミナルから簡単に利用できます)。

PHP を有効にするには以下の手順を使用してください。 これは、ローカルの開発環境を手早く設定する方法を示したものです。 常に PHP を最新バージョンに更新することを *強く推奨*します。 多くのソフトウェアでは、新しいバージョンでは多くのバグが修正されています。 また機能追加もされています。PHP も同様です。 詳細は、適切な MAC OS X インストールドキュメントを参照ください。 以下の手順は初心者を対象としたもので、 まずデフォルトの設定で動作させるための手順を示しています。 すべてのユーザが、新しいパッケージ版をコンパイルしてインストールすることを推奨します。

標準的なインストール方法は mod_php を使用するものです。Mac OS X 上の Apache web server (デフォルトのウェブサーバで、System Preferences からアクセスできます) 上に、mod_php を組み込むには次のようにします。

1. Apache の設定ファイルを開きます。デフォルトでは */etc/httpd/httpd.conf* にあります。 *Finder* あるいは *Spotlight* を使用してこれを見つけることは難しいでしょう。 このファイルはプライベート設定されており、その所有者は *root* ユーザだからです。

   > **注意**: Unix ベースのテキストエディタ、たとえば *nano* を用いて、ターミナルでこのファイルを開きます。このファイルの所有者は root なので、*sudo* コマンドを使用して (root として) 開く必要があります。つまり、ターミナル上で *sudo nano /etc/httpd/httpd.conf* と入力します (その後、パスワードを聞かれます)。 覚えておくべき nano コマンドは次のとおりです。*^w* (検索)、 *^o* (保存) そして *^x* (終了)。 ここで ^ は Ctrl キーを表します。

2. テキストエディタで、次のような行 (これらのふたつの行は並んでいないこともあります。 それぞれをファイル中で探してください) のコメントをはずします (# を削除します)。

   ```
   #  LoadModule  php4_module  libexec/httpd/libphp4.so
   #  AddModule  mod_php4.c
   ```

   パスに注意しましょう。将来 PHP をビルドする際には 上のファイルは移動するかコメントアウトする必要があります。

3. 指定した拡張子 (例: .php .html および .inc) が PHP でパースされるようにします。

以下のような行が *httpd.conf* にあれば (Mac Panther 以降にはあります)、 PHP を有効にするだけで *.php* ファイルが自動的に PHP で処理されます。

```
<IfModule  mod_php4.c>
     # If  php  is  turned  on,  we  respect  .php  and  .phps  files.
      AddType  application/x-httpd-php  .php
      AddType  application/x-httpd-php-source  .phps

     # Since  most  users  will  want  index.php  to  work  we
     # also  automatically  enable  index.php
     <IfModule  mod_dir.c>
          DirectoryIndex  index.html  index.php
     </IfModule>
</IfModule>
```

4. DirectoryIndex でデフォルトインデックスファイルが読み込まれるようにします。 これも *httpd.conf* で設定します。典型的なパターンは *index.php* および *index.html* でしょう。デフォルトでは *index.php* が有効になります。上で見たような設定がすでにあるからです。 これを適切に調整しましょう。
5. *php.ini* の場所を設定するか、デフォルトを使用します。 Mac OS X におけるデフォルトの場所は */usr/local/php/php.ini* で、 phpinfo() をコールするとこの情報を表示します。 *php.ini* を使用しない場合は、PHP はすべてデフォルト値を使用します。 php.ini ファイルはどこにおけばいいのですか? が、関連する FAQ です。
6. *DocumentRoot* を配置あるいは設定します。 これは、すべてのウェブファイルのルートディレクトリとなります。 このディレクトリ内のファイルはウェブサーバで処理されるようになるので、 PHP ファイルは PHP でパースしてからブラウザに出力されます。 典型的なデフォルトのパスは */Library/WebServer/Documents* ですが、これは *httpd.conf* で別の場所にすることができます。 また、各ユーザの DocumentRoot は */Users/yourusername/Sites* となります。
7. phpinfo() ファイルを作成します。

   phpinfo() 関数は、PHP についての情報を表示します。 DocumentRoot 内に、次のような PHP コードを含むファイルを作成してください。

   ```
   <?php  phpinfo(); ?>
   ```

8. Apache を再起動し、先ほど作成した PHP ファイルを読み込みます。 再起動するには、シェルで *sudo apachectl graceful* を実行するか、あるいは OS X System Preferences で "Personal Web Server" オプションを使用して停止/起動します。 デフォルトでは、ローカルファイルをブラウザで読み込むには *http://localhost/info.php* のような URL を指定します。 あるいは、各ユーザディレクトリの DocumentRoot の場合は *http://localhost/~yourusername/info.php* のようになります。

CLI (あるいは旧バージョンの CGI) は、*php* という名前で、おそらく */usr/bin/php* にあります。 ターミナルを開き、PHP マニュアルの Open up the terminal, read the PHP をコマンドラインから使用する を読んだうえで *php -v* を実行してみましょう。 これは、PHP バイナリのバージョンを表示します。 phpinfo() をコールしても、この情報を取得できます。

Mac OS X サーバ用にコンパイルする

Mac OS X サーバへのインストール

1. Apache と PHP の最新版を入手します。
2. tar ファイルを展開し、Apache の **configure** プログラムを以下のように実行します。

   ```
   ./configure  --exec-prefix=/usr  ¥
   --localstatedir=/var  ¥
   --mandir=/usr/share/man  ¥
   --libexecdir=/System/Library/Apache/Modules  ¥
   --iconsdir=/System/Library/Apache/Icons  ¥
   --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers  ¥
   --enable-shared=max  ¥
   --enable-module=most  ¥
   --target=apache
   ```

3. コンパイラに最適化を行わせたい場合には、次の行を追加します。

   ```
   setenv  OPTIM=-O2
   ```

4. 次に、PHP 4 のソースディレクトリで、configure を行います。

   ```
   ./configure  --prefix=/usr  ¥
        --sysconfdir=/etc  ¥
        --localstatedir=/var  ¥
        --mandir=/usr/share/man  ¥
       --with-xml  ¥
         --with-apache=/src/apache_1.3.12
   ```

   他に追加するもの (MySQL、GD 等) がある場合、必ずここでこれらを追加する ようにしてください。--with-apache 文字列に関しては、Apache ソースのディレクトリを */src/apache_1.3.12* のように指定してください。

5. **make** および **make install** を実行します。 これにより、Apache ソースディレクトリに *src/modules/php4*の下の ディレクトリが追加されます。

6. PHP4 を ビルドするよう Apache を 再設定します。

```
./configure  --exec-prefix=/usr  ¥
--localstatedir=/var  ¥
--mandir=/usr/share/man  ¥
--libexecdir=/System/Library/Apache/Modules  ¥
--iconsdir=/System/Library/Apache/Icons  ¥
--includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers  ¥
--enable-shared=max  ¥
--enable-module=most  ¥
--target=apache  ¥
--activate-module=src/modules/php4/libphp4.a
```

   libmodphp4.a が期限切れというメッセージが出力されるかもしれません。 この場合、Apache ソースディレクトリの配下の *src/modules/php4* ディレクトリに行き、以下のコマンドを実行します、 **ranlib libmodphp4.a** 。 次に Apache ソースディレクトリのルートに戻り、 上記の **configure** コマンドを再び実行します。 これにより、リンクテーブルが最新になります。再度、 **make** および **make install**を実行します。

7. *php.ini-dist* ファイルを PHP4 ソースディレクトリから *bin* ディレクトリにコピーし、 リネームします。 **cp php.ini-dist /usr/local/bin/php.ini** 、 または (local ディレクトリが無い場合) **cp php.ini-dist /usr/bin/php.ini** 。

Mac OS X クライアント用にコンパイルする

以下の手順は、PHP モジュールを Mac OS X に付属する Apache Web サーバ用に インストールする際の手引きとなるものです。本バージョンには、MySQL および PostgreSQL データベースのサポートが含まれます。以下の手引きは、 » Marc Liyanage 氏に提供して頂いたものです。

警告

以下の手順を行う際は、Apache Web サーバをダウンさせる可能性 があるので注意してください

以下のインストール手順を実行してください。

1. ターミナルウインドウをオープンします。
2. **wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz** とし、ダウンロード完了を待ちます。
3. **gunzip libphp4.so.gz** と入力します。
4. **sudo apxs -i -a -n php4 libphp4.so** と入力します。
5. 次に **sudo open -a TextEdit /etc/httpd/httpd.conf** と 入力すると、TextEdit により Web サーバの設定ファイルがオープンされます。ファイルの 末尾の方にある以下の 2 行を探してください（検索コマンドを使用しましょう。）

```
#AddType  application/x-httpd-php  .php
#AddType  application/x-httpd-php-source  .phps
```

   ハッシュ記号 (#) を削除して、ファイルを保存し、 TextEdit を終了させます。

6. 最後に、Web サーバを再起動させます。 *sudo apachectlgraceful* と入力してください。

この時点で PHP が動作しているはずです。*test.php* という 名前のファイルを *Sites* フォルダに 作成してください。このファイルを編集し、*<?php phpinfo() ?>* と書いてください。

次に、*127.0.0.1/~あなたのユーザ名/test.php* を Web ブラウザで オープンしてください。ステータステーブルが表示され、インストールした PHP モジュールに関する情報を見ることができるはずです。

Windows システムへのインストール

目次

本章は、Windows 98/Me および Windows NT/2000/XP に適用されます。 PHP は、Windows 3.1 などの 16 ビットのプラットフォームでは動作しません。 PHP がサポートする Windows プラットフォーム を Win32 と呼ぶことがあります。 Windows95 は PHP4.3.0 以降サポートされなくなりました。

PHP を Windows にインストールするには、 マニュアルインストール と インストーラによる方法 の 2 種類があります。

Microsoft Visual Studio を持っていれば、オリジナルのソースコードから PHP を ビルドする ことも可能です。

Windows システムに PHP をインストールした後、機能を追加するために 拡張モジュールのロード が必要な場合があります。

警告

インターネットでオールインワンのインストーラがいくつか見かけられます。 しかし、いずれも PHP.net により公認されたものではありません。 システムを最適化し、また、安全を保つためには、 » http://www.php.net/downloads.php にある公式 windows パッケージのいずれかを使用することがもっとも良い選択ではないかと我々は考えています。

Windows インストーラ (PHP 5.2 以降)

Windows 用 PHP インストーラの最新版は、MSI 形式になっています。これは Wix Toolkit (» http://wix.sourceforge.net/) で作成されています。 このインストーラは、インストールおよび PHP の設定、そして組み込みのモジュールと PECL 拡張モジュールの設定だけでなく、 IIS、Apache そして Xitami などといった多くのウェブサーバの設定も行います。

最初に、お好みの HTTP (ウェブ) サーバをシステムにインストールし、動作確認をします。 それから、次のうちのいずれかのインストール形式を選択します。

通常のインストール

MSI インストーラを実行し、インストールウィザードの指示に従います。 まず最初に設定するウェブサーバを選択し、 それに伴って必要となる設定を行います。

次に、さまざまな機能や拡張モジュールの中から インストールして有効にしたいものを選択します。 各項目のドロップダウンメニューで "Will be installed on local hard drive" を選択することで、その機能をインストールするかどうかを切り替えます。 "Entire feature will be installed on local hard drive" を選択すると、 その機能に関連するすべての機能がインストールされます (たとえば "PDO" に対してこのオプションを選択すると、 すべての PDO ドライバがインストールされます)。

警告

すべての項目をデフォルトでインストールしてしまうのはお勧めしません。 それらの多くは、適切に動作させるためには PHP 以外の外部の依存性を解決する必要があるからです。 その代わりに、いったんインストールした後で コントロールパネルの「プログラムの追加と削除」 で修復モードを使用して拡張モジュールを有効にするようにしましょう。

次に、インストーラは Windows で使用する PHP と *php.ini* ファイルを設定し、ウェブサーバで PHP を使用できるように設定します。 現在インストーラがサポートしているのは IIS、 Apache、Xitami および Sambar Server のみです。 それ以外のウェブサーバを使用する場合は、自分で設定する必要があります。

サイレントインストール

このインストーラはサイレンとモードもサポートしています。 このモードは、システム管理者が簡単に PHP 環境を構築する際に便利です。 サイレントモードは、次のようにして使用します。

```
msiexec.exe  /i  php-VERSION-win32-install.msi  /q
```

インストール先ディレクトリを設定するには、インストール時のパラメータとして指定します。 例えば、e:\php にインストールする場合は次のようになります。

```
msiexec.exe  /i  php-VERSION-win32-install.msi  /q  INSTALLDIR=e:¥php
```

同様にして、Apache の設定ディレクトリ (APACHEDIR) や Sambar Server のディレクトリ (SAMBARDIR)、 そして Xitami Server のディレクトリ (XITAMIDIR) も指定することが可能です。

また、インストールする機能を指定することもできます。例えば、 mysqli 拡張モジュールおよび CGI 実行ファイルをインストールするには次のようにします。

```
msiexec.exe  /i  php-VERSION-win32-install.msi  /q  ADDLOCAL=cgi,ext_php_mysqli
```

現在、インストールする機能としてサポートされている項目は次のとおりです。

```
MainExecutable - php.exe 実行ファイル
ScriptExecutable - php-win.exe 実行ファイル
ext_php_* - 各種拡張モジュール (例: MySQL の場合は ext_php_mysql)
apache13 - Apache 1.3 モジュール
apache20 - Apache 2.0 モジュール
apache22 - Apache 2,2 モジュール
apacheCGI - Apache CGI 実行ファイル
iis4ISAPI - IIS ISAPI モジュール
iis4CGI - IIS CGI 実行ファイル
NSAPI - Sun/iPlanet/Netscape サーバモジュール
Xitami - Xitami CGI 実行ファイル
Sambar - Sambar Server ISAPI モジュール
CGI - php-cgi.exe 実行ファイル
PEAR - PEAR インストーラ
Manual - CHM 形式の PHP マニュアル
```

MSI インストーラをコマンドラインから使用する方法の詳細については » http://msdn.microsoft.com/library/en-us/msi/setup/command_line_options.asp を参照ください。

インストーラによる PHP のアップグレード

アップグレードの際も、ダブルクリックあるいはコマンドラインから通常どおりインストーラを実行します。 インストーラが現在のインストールオプションを読み取り、 現在インストールされているバージョンを削除してから 同じオプションで PHP をインストールしなおします。 インストールディレクトリのファイルを手動で置き換えるのではなく、 この方法で PHP をアップグレードすることを推奨します。

Windows インストーラ (PHP 5.1.0 以前)

CGI 版の PHP をインストールする Windows 用 PHP インストーラが、 » http://www.php.net/downloads.php から取得可能です。 IIS、PWS、Xitami に対しては Web サーバの設定も同時に行われます。 このインストーラには、PHP 拡張モジュール (php_*.dll) が含まれていません。 拡張モジュールは、Zip パッケージもしくは PECL ダウンロード からのみ入手できます。

> **注意**: Windows インストーラは PHP を動作させるための簡便な方法で、 （拡張モジュールの自動設定ができないといった）制約がいくつかあります。 インストーラの利用は、PHP をインストールする最適な方法ではありません。

まずはじめに、使用する HTTP (Web) サーバを システムにインストールし、完全に動作するようにして下さい。

インストーラを実行し、インストールウイザードの指示に従って下さい。 2 種類のインストール方法がサポートされています。一つ目は standard で、 設定の選択肢についてデフォルト値が示されます。もう一つは advanced で、 選択肢について質問が行われます。

インストールウイザードは、*php.ini* ファイルを設定し、 PHP が使用可能となるように Web サーバを設定するために必要な情報を集めます。 ただし、Apache をはじめとするいくつかの web サーバでは、インストーラによる自動での 設定変更は行われず、手動で設定変更する必要があります。

インストールが完了すると、システムもしくはサーバを再起動する必要があるのか、 そのまま PHP の使用を開始すればよいのか表示されます。

警告

こうしてインストールされた PHP の設定は安全ではないことに注意して下さい。安全に PHP を設定したい場合は、マニュアルでインストールし、オプションを注意深く設定するべきです。 上記の自動設定により PHP のインストールを瞬時に行うことが可能となりますが、 インターネットに接続されたサーバで使用されうるものではありません。

マニュアルインストール

ここからは、Microsoft Windows に手動で PHP をインストールし、Web サーバを 設定する手順について説明します。

始めに、» http://www.php.net/downloads.php のダウンロードページから zip バイナリアーカイブをダウンロードしてください。

公式に配布されている Microsoft Window 向けの PHP インストーラを含め 多数のオールインワンのインストールキットが存在しますが、 まずは、幾ばくかの時間を割いて独力で PHP をセットアップしてみることをお勧めします。 そうすることで、システムをより理解することができ、PHP 拡張モジュールの インストールも必要に応じて容易にできるようになるでしょう。

> **注意**: **前バージョンから PHP をアップグレードする場合** 古いバージョンのマニュアルでは、ini ファイルおよび DLL ファイルをシステムフォルダ (*C:¥WINDOWS* など) へ移動させることを推奨していました。この移動により、インストール手順は簡単になりますが、 アップグレードは面倒になっていました。 新しいバージョンのインストールにあたっては、これら移動させたファイル（システムフォルダ内の *php.ini* や PHP 関連の DLLなど）をすべて削除することを推奨します。 この時システムを壊さないようにするために、バックアップを確実に行ってください。 古い *php.ini* は新しい PHP を設定するのに有用です。 本解説で推奨する PHP のインストール方法は、以下の解説に示すとおり、すべての PHP 関連の ファイルをひとつのフォルダにまとめ、システムパスにそのフォルダを登録する方法です。

> **注意**: **MDAC 要件** *Windows 98/NT4* を使用している場合には、 プラットフォームに合う Microsoft Data Access Components (MDAC) の最新版を 入手してください。» http://msdn.microsoft.com/data/ からダウンロードできます。MDAC が必要な理由は、ODBC が Windows バイナリにビルトインされているためです。

各サーバ特有の設定を行う前に、以下に示すステップを完了させてください。

まず、配布ファイルを適当なフォルダに展開します。PHP 4 の場合は、*C:¥* がよいでしょう。zip パッケージは *php-4.3.7-Win32* のようなフォルダ名で展開されます。 PHP 5 の場合は、PHP 4 のようにフォルダ内に展開されないので、*C:¥php* 内で展開してください。 他の位置に展開することもできますが、空白を含むパス (例:c:\program files\php) にすることはお勧めできません。Web サーバによってはクラッシュを引き起こします。

展開されたフォルダの構造は、PHP 4 と PHP 5 とで異なり、以下のようになります。

Example#1 PHP 4 パッケージ構造

```
c:¥php
    |
    +--cli
    |   |
    |   |-php.exe             -- CLI 実行ファイル - コマンドラインでのスクリプト実行専用
    |
    +--dlls                   -- 拡張モジュールの利用に必要な DLL
    |   |
    |   |-expat.dll
    |   |
    |   |-fdftk.dll
    |   |
    |   |-...
    |
    +--extensions             -- PHP 拡張モジュールの DLL
    |   |
    |   |-php_bz2.dll
    |   |
    |   |-php_cpdf.dll
    |   |
    |   |-..
    |
    +--mibs                   -- SNMP 用サポートファイル
    |
    +--openssl                -- Openssl 用サポートファイル
    |
    +--pdf-related            -- PDF 用サポートファイル
    |
    +--sapi                     -- SAPI (server module support) DLL
    |   |
    |   |-php4apache.dll
    |   |
    |   |-php4apache2.dll
    |   |
    |   |-..
    |
    +--PEAR                   -- PEAR の初期コピー
    |
    |
    |-go-pear.bat             -- PEAR セットアップ用スクリプト
    |
    |-..
    |
    |-php.exe                 -- CGI 実行ファイル
    |
    |-..
    |
    |-php.ini-dist            -- デフォルトの php.ini 設定
    |
    |-php.ini-recommended     -- 推奨される php.ini 設定
    |
    |-php4ts.dll              -- コア PHP DLL
    |
    |-...
```

Example#2 PHP 5 パッケージ構造

```
c:¥php
    |
    +--dev
    |   |
    |   |-php5ts.lib
    |
    +--ext                    -- PHP 拡張モジュールの DLL
    |   |
    |   |-php_bz2.dll
    |   |
    |   |-php_cpdf.dll
    |   |
    |   |-..
    |
    +--extras
    |   |
    |   +--mibs               -- SNMP 用サポートファイル
    |   |
    |   +--openssl            -- Openssl 用サポートファイル
    |   |
    |   +--pdf-related        -- PDF 用サポートファイル
```

```
|   |
|   |-mime.magic
|
+--pear                    -- PEAR の初期コピー
|
|
|-go-pear.bat              -- PEAR セットアップ用スクリプト
|
 |-fdftk.dll
|
|-..
|
|-php-cgi.exe              -- CGI 実行ファイル
|
|-php-win.exe              -- コマンドプロンプトを開かずにスクリプトを実行する
|
|-php.exe                  -- CLI 実行ファイル - コマンドラインでのスクリプト実行専用
|
|-..
|
|-php.ini-dist             -- デフォルトの php.ini 設定
|
 |-php.ini-recommended     -- 推奨される php.ini 設定
|
  |-php5activescript.dll
|
 |-php5apache.dll
|
 |-php5apache2.dll
|
|-..
|
|-php5ts.dll               -- コア PHP DLL
|
|-...
```

PHP 4 と PHP 5 とで、相違点もあり、共通点もあります。 双方とも、CGI 実行ファイル、CLI 実行ファイル、およびサーバモジュールがあります。しかし、フォルダや ファイル名が異なります。PHP 4 では、サーバモジュールは、 *sapi* フォルダ内にありますが、 PHP 5 ではそういったフォルダは無く、PHP を展開したフォルダのルートにあります。 PHP 5 の拡張モジュールが必要とする DLL も、別フォルダ内には納められていません。

> **注意**: PHP 4 では、*dll* フォルダおよび *sapi* フォルダ内のすべてのファイルを メインのフォルダ（*C:¥php* など）に 移動してください。

PHP 4 および PHP 5 で提供されているサーバモジュールは以下の通りです。

- *sapi/php4activescript.dll (php5activescript.dll)* - ActiveScript エンジン。Windows アプリケーションに PHP を埋め込むことが可能
- *sapi/php4apache.dll (php5apache.dll)* - Apache 1.3.x モジュール
- *sapi/php4apache2.dll (php5apache2.dll)* - Apache 2.0.x モジュール
- *sapi/php5apache2_2.dll* - Apache 2.2.x モジュール
- *sapi/php4isapi.dll (php5isapi.dll)* - ISAPI モジュール。 IIS 4.0/PWS 4.0 以降を始とした ISAPI 互換サーバ用
- *sapi/php4nsapi.dll (php5nsapi.dll)* - Sun/iPlanet/Netscape 用サーバモジュール
- *sapi/php4pi3web.dll (PHP 5 では提供なし)* - Pi3Web 用サーバモジュール

サーバモジュールは、CGI バイナリと比べ、パフォーマンスが非常に高く、 機能も追加されています。CLI 版は、PHP をコマンドライン用のスクリプトとして 使うためのものです。CLI 版について詳しくは PHP をコマンドラインから使用するの章を参照してください。

警告

SAPI モジュールは、バージョン 4.1 以降、顕著な改善が行われています。 一方、古いシステムにおいては、サーバエラーが発生したり、ASP 等のサーバモジュールが 落ちたりする可能性があります。

CGI バイナリ、CLI バイナリ、およびサーバモジュールのいずれも *php4ts.dll* (*php5ts.dll*) を必要とします。 PHP がこのファイルを見つけられるようにする必要があります。 探索順は、以下の通りです。

- *php.exe* のコール元のフォルダ。SAPI モジュールを 使用している場合、WEB サーバのフォルダ (例、 *C:¥Program Files¥Apache Group¥Apache2¥bin*)
- Windows の *PATH* 環境変数に登録されたフォルダ

php4ts.dll / *php5ts.dll* を有効に するには、[1] Windows のシステムフォルダにコピーする方法、 [2] WEB サーバのフォルダにコピーする方法、[3] PHP フォルダ (*C:¥php*) を *PATH* 環境変数に登録する方法の３つの選択肢があります。メンテナンスを考慮すると、 ３番目の環境変数に登録する方法をとるべきでしょう。こうすれば、将来の PHP の アップグレードが容易になります。PHP フォルダを環境変数に登録する方法に ついては、FAQ を参照して ください (また、コンピュータを再起動することを忘れないでください。 ログオフするだけでは不十分です)。

次のステップは、PHP の設定ファイル *php.ini* を正しく記述することです。 配布される zip ファイルには *php.ini-dist* と *php.ini-recommended* の二つの ini ファイルが含まれています。 パフォーマンスとセキュリティの観点から最適化された初期設定がなされているので、*php.ini-recommended* の使用を推奨します。 *php.ini-dist* から多くの点で変更が行われています。 ini ファイルには詳しくコメントが書かれている

ので、注意深く読んでみると良いでしょう。 たとえば、display_errors や magic_quotes_gpc が *off* となっていたりします。 さらに、設定ファイル のセクションもよく読んで 各要素を自ら設定してみてください。 PHP は、デフォルトの ini ファイルの設定できちんと動作するとはいえ、 最高のセキュリティを達成したいならば、自ら手を動かして設定することが最善の方法です。 選んだ ini ファイルを、PHP がアクセスできるフォルダにコピーし、*php.ini* にリネームしてください。PHP は、実行時設定 の節で述べられている場所から *php.ini* を探します。

Apache 2 を使用する場合、PHPIniDir ディレクティブを使うのが最も簡単でしょう (Apache 2 へのインストール ページ参照。) 他の場合、*PHPRC* 環境変数を設定するのが良いでしょう。 詳しい方法については、こちらの FAQ エントリ で解説されています。

> **注意**: Windows NT, 2000, XP または 2003 で NTFS を使用している場合、WEB サーバを 実行するユーザが、*php.ini* を読める権限があることを確認してください。

以下のステップは、必要に応じて行ってください。

- *php.ini* ファイルを編集し、 doc_root に Web サーバーの ドキュメントルートを設定します（ただし、OmniHTTPd を使用する場合は編集しないこと。） 例えば、以下のように設定します。

  ```
  doc_root = c:\inetpub          // IIS/PWS の場合
  doc_root = c:\apache\htdocs // Apache の場合
  ```

- PHP の起動時に、必要な拡張モジュールがロードされるように設定します。セットアップの方法や、 ビルトイン済みのモジュールについては Windows 用拡張モジュール のセクションを 参照してください。PHP の新規インストールの場合は、まず PHP が拡張モジュールなしで正しく動作することを 確認してから、php.ini を変更して拡張モジュールを有効にするほうが賢明です。
- PWS と IIS においては、browscap を 次のように指定することができます。 *c:¥windows¥system¥inetsrv¥browscap.ini* (Windows 9x/Me の場合)、 *c:¥winnt¥system32¥inetsrv¥browscap.ini* (NT/2000 の場合)、 *c:¥windows¥system32¥inetsrv¥browscap.ini* (XP の場合)。 *browscap.ini* の編集については、 FAQ を参照してください。

以上で、Windows への PHP のインストールが完了しました。ついで、 使用する WEB サーバ にあわせて、PHP を利用可能とするための設定を行います。 目次から使用する WEB サーバを選択し、該当するセクションを参照してください。

ActiveScript

このセクションでは、ActiveScript で PHP を使用する場合について説明します。

ActiveScript は Windows 独自の SAPI で、PHP スクリプトが Windows Script Host, ASP/ASP.NET, Windows Script Components や Microsoft Scriptlet control といった ActiveScript 互換ホストで実行可能になります。

PHP 5.0.1 の時点で、ActiveScript サポートは » PECL リポジトリに移動されました。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

> **注意**: まず始めに、マニュアルインストールの手順 をお読みください。

PHP をインストールした後、ActiveScript 用の DLL (*php5activescript.dll*) をダウンロードし、メイン PHP フォルダ (*C:¥php* 等) へ置いてください。

ファイルを設置できたら、システムに DLL を登録します。 （スタートメニューから）コマンドプロンプトを開き、 PHP のフォルダへ移動し（*cd C:¥php* 等とする）、 *regsvr32 php5activescript.dll* と打ち込むと DLL を登録できます。

ActiveScript が動作しているかテストするには、*test.wsf* という名前（拡張子名が重要）で新しいファイルを作成し、 以下のようにタイプしてください。

```
<job  xml:id="test">

  <script  language="PHPScript">
    $WScript->Echo("Hello  World!");
  </script>

</job>
```

保存した後、ファイルをダブルクリックして、ウインドウに 「Hello World!」 と表示されたら、上手く動作しています。

> **注意**: PHP 4では、エンジンは「ActivePHP」と命名されました。 したがって、PHP 4を使用している場合、「PHPScript」を上記の例における 「ActivePHP」に取り替えるべきです。

> **注意**: ActiveScript では、デフォルトの *php.ini* ファイルは使われません。 代わって、ロードする .exe ファイルと同じフォルダが探されます。 拡張モジュールを読み込みたい場合は、 *php-activescript.ini* ファイルを作成し、当該フォルダに 設置すると良いでしょう。

Microsoft IIS / PWS

この章では、IIS (Microsoft Internet Information Server) に関する 注意やヒントを取り上げます。

警告

CGI としてセットアップすると、サーバは様々な攻撃を受ける可能性があります。これらの攻撃からサーバを守る方法については、 CGI セキュリティ のセクションを参照してください。

IIS あるいは PWS で PHP を使用する際に、一般的に考慮すべき点

- まず最初に、 マニュアルインストール をお読みください。 ここには Windows 上に PHP をインストールするための重要な情報が含まれているので、 決して読み飛ばしてはいけません。
- CGI を利用する場合は、*php.ini* 内で cgi.force_redirect PHP ディレクティブを *0* に設定する必要があります。 cgi.force_redirect に関する FAQ に重要な情報がありますので お読みください。また、CGI を利用する場合は cgi.redirect_status_env ディレクティブを設定することもあるかもしれません。これらの ディレクティブを使用する際には、*php.ini* 内でその項目が コメントアウトされていないことを確認してください。
- PHP 4 では CGI は *php.exe* という名前ですが、 PHP 5 ではその名前は *php-cgi.exe* となります。 PHP 5 では *php.exe* は CLI であり、CGI ではありません。
- Windows の環境変数 *PATH* を変更し、PHP の ディレクトリを含めるようにしてください。こうすることによって PHP の DLL ファイルや PHP 実行ファイルを PHP ディレクトリの中に置いておくことが可能となり、Windows のシステムディレクトリを汚すことが避けられます。 詳細な情報は、FAQ の PATH を設定する方法 を参照ください。
- IIS ユーザ (通常は IUSR_MACHINENAME) に対しては、 *php.ini*、ドキュメントルートおよびセッションの一時ディレクトリなどの さまざまなファイルやディレクトリへの読み込み権限を与えておくことが 必要です。
- *php.ini* 内のディレクティブ extension_dir および doc_root は、必ず適切に設定するようにしましょう。 これらのディレクティブの内容は、PHP がインストールされている システムに依存します。PHP 4 では extension_dir は *extensions* となりますが、PHP 5 では *ext* です。そのため PHP 5 の extensions_dir の値は例えば *"c:¥php¥ext"* のようになり、 IIS の doc_root の値は例えば *"c:¥Inetpub¥wwwroot"* のようになります。
- *php_mysql.dll* や *php_curl.dll* のような PHP 拡張モジュールの DLL ファイルは、zip パッケージ版の PHP 配布物に含まれています (インストーラ版には含まれません)。 PHP 5 では多くの拡張モジュールが PECL に含まれるようになり、 "Collection of PECL modules" パッケージとしてダウンロード できるようになりました。この中には *php_zip.dll* や *php_ssh2.dll* などが含まれます。 » PHP はここからダウンロードできます。
- 実行ファイルを定義する際に「ファイルの存在を確認する」をチェックします。 少しパフォーマンス は落ちますが、IIS (もしくは PWS) が PHP を起動する前に、 そのスクリプトが存在し認証上の問題がないかをチェックするようになります。 PHP は CGI エラー時に空白の画面しか出力しませんが、こうすることで、 より解りやすい 404 エラーメッセージを出力させるようにすることができます。
- 配布されている PHP の実行ファイルは 32bit アプリケーションです。 64bit 版の Windows を使用している場合は、自分でバイナリを再ビルドするか、 あるいは 32bit の拡張モジュールを実行できるように IIS を設定します。 この設定は、通常は IIS の管理スクリプトで Cscript.exe adsutil.vbs SET W3SVC/AppPools/Enable32bitAppOnWin64 1 のようにします。

Windows NT/200x/XP 上の IIS 4 以降

PHP は、CGI バイナリあるいは ISAPI モジュールのいずれかの形式で インストールされていることでしょう。いずれにしても、まず 「マイクロソフト マネージメントコンソール」 (Windows NT 4.0 オプションパック環境では「インターネット サービスマネージャ」、Windows 2000/XP では コントロールパネル=>管理ツールにあります) を起動する必要があります。 次に、Web サーバノード (たいていは「既定の Web サイト」と表示されています) 上で右クリックし、「プロパティ」を選択します。

CGI バイナリを使う場合は、次のようにしてください。

- 「ホームディレクトリ」あるいは「仮想ディレクトリ」「ディレクトリ」タブで 以下のようにします。
- 実行アクセス許可を「スクリプトのみ」に変更します。
- 「構成」ボタンをクリックし、「マッピング」タブを選択します。 「追加」をクリックし、実行可能ファイルに適切な CGI ファイルを指定します。 たとえば PHP 5 では *C:¥php¥php-cgi.exe* となります。 「拡張子」に *.php* を指定し、「動詞」は空白のまま、 「スクリプトエンジン」チェックボックスをチェックしてください。 そして、「OK」ボタンを何度かクリックしてください。
- 適切なセキュリティを設定してください (これは インターネットサービスマネージャで行います)。もし NT サーバで NTFS ファイルシステムを使用しているなら、 *php.exe* / *php-cgi.exe* があるディレクトリへの実行権限を I_USR_ に追加してください。

ISAPI モジュールを使う場合、次のようにしてください。

- PHP を使用した HTTP 認証を実行しない場合は、この手順を飛ばしてください。 「ISAPI フィルタ」タブで新規フィルタを追加します。「フィルタ名」として PHP を使用し、「実行ファイル」には *php4isapi.dll* / *php5isapi.dll* へのパスを入力してください。
- 「ホームディレクトリ」あるいは「仮想ディレクトリ」「ディレクトリ」タブで 以下のようにします。
- 実行アクセス許可を「スクリプトのみ」に変更します。
- 「構成」ボタンをクリックし、「マッピング」タブを選択します。 「追加」をクリックし、実行可能ファイルに適切な ISAPI DLL を指定します。 たとえば PHP 5 では *C:¥php¥php5isapi.dll* となります。 「拡張子」に *.php* を指定し、「動詞」は空白のまま、 「スクリプトエンジン」チェックボックスをチェックしてください。 そして、「OK」ボタンを何度かクリックしてください。
- IIS を停止させます (NET STOP iisadmin)。
- IIS を再度起動します (NET START w3svc)。

IIS 6 (2003 Server) の場合は IIS マネージャを開き、「Web サービス拡張」に 移動し、「新しい Web サービス拡張を追加」を選択し、たとえば「PHP」などと 拡張名を入力し、「追加」ボタンを押して ISAPI ファイル (*php4isapi.dll* または *php5isapi.dll*) あるいは CGI (*php.exe* または *php-cgi.exe*) を選択し、「拡張の状態を許可済みに設定する」をチェックして「OK」ボタンを クリックします。

デフォルトのページとして *index.php* を使用するには 以下のようにします。 (訳注:「既定の Web サイト」のプロパティダイアログで) 「ドキュメント」タブを選択し、「追加」を選択します。そこで *index.php* と入力し、「OK」をクリックします。 「上に移動」や「下に移動」を使用して順番を調整します。 これは Apache での DirectoryIndex の設定と同じです。

上記の手順を PHP スクリプトに関連づけたい拡張子ごとに繰り返してください。 一般的な拡張子は *.php* ですが、 古いアプリケーションでは *.php3* が必要な場合があります。

CPU 使用率が 100% となる場合は、IIS の設定「*ISAPI アプリケーションをキャッシュ*」をオフにしてください。

Windows 上の PWS 4

PWS 4 は ISAPI をサポートしていません。PHP CGI のみが使用可能です。

- 配布ファイル内の *pws-php4cgi.reg* / *pws-php5cgi.reg* ファイル (PHP 4 の場合は SAPI フォルダ、 PHP 5 の場合メインフォルダを参照) を編集し、*php.exe* / *php-cgi.exe* の設置場所を反映させます。 バックスラッシュはエスケープする必要があります。例、 *[HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥w3svc¥parameters¥Script Map] ".php"="C:¥¥php¥¥php.exe"* (PHP 5 の場合は *C:¥¥php¥¥php-cgi.exe* とします)。 編集を終えたらダブルクリックして、レジストリファイルをシステムに反映させます。
- PWS マネージャで、PHP を実行させたいフォルダで右クリックし、 プロパティを選択します。「実行」チェックボックスをチェックし、確認を押します。

Windows 上の PWS/IIS 3

PWS/IIS 3 サーバを設定するには、配布ファイルに含まれる REG ファイル (SAPI フォルダの *pws-php4cgi.reg* が PHP 4 用、 メインフォルダの *pws-php5cgi.reg* が PHP 5 用) を使用することを推奨します。 このファイルを編集し、拡張子および PHP インストールディレクトリを自分用の設定に 調整することが可能です。さもなくば、以下の手順により、手動でこの設定を行うことも可能です。

警告

以下の手順では Windows レジストリに対して直接変更を加えます。 ひとつでも間違えると、システムが不安定になる可能性があります。 レジストリのバックアップをとることをを強く推奨します。 PHP 開発チームは、レジストリが損傷した場合の責任を負いません。

- Regedit を起動します。
- *HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap* へ移動します。
- 「編集」メニューから、*新規 -> 文字列値* を選択します。
- PHP スクリプトに使用したい拡張子を入力します。 たとえば *.php* となります。
- この新しく作成した文字列値をダブルクリックし、入力欄に *php.exe* までのパスを入力します。たとえば PHP 4 では *C:¥php¥php.exe "%s" %s*、 PHP 5 では *C:¥php¥php-cgi.exe "%s" %s* となります。
- PHP スクリプトに関連付けたいすべての拡張子に対して、 上記の手順を繰り返します。

以下の手順は、Web サーバのインストールに影響を与えるものではなく、 PHP スクリプトをコマンドライン (例:「ファイル名を指定して実行」で *C:¥myscripts¥test.php* を入力するなど) やダブルクリックで実行させたい場合にのみ行ってください。 ダブルクリックした際にテキストエディタに読み込まれるままとしたいのなら、 以下の手順はスキップしてください。

- *HKEY_CLASSES_ROOT* へ移動します。
- 編集メニューから、*新規* -> キー を選択します。
- キーを先に入力した拡張子と同じ名前にします。例、*.php*
- 作成した新しいキーを選択した状態にし、右の欄で 「(既定)」 をダブルクリックして *phpfile* と入力します。
- 先に登録したすべての拡張子に対して繰り返します。
- *HKEY_CLASSES_ROOT* で、再度 *新規->キー* とし、*phpfile* という名前にします。
- 新しいキー *phpfile* を選択した状態にし、 右の欄で 「(既定)」 をダブルクリックして *PHP Script* と入力します。
- *phpfile* を右クリックし、*新規->キー* を選択し、新しくできたキーを *Shell* という名前にします。
- *Shell* を右クリックし、*新規->キー* を選択し、新しくできたキーを *open* という名前にします。
- *open* を右クリックし、*新規->キー* を選択し、新しくできたキーを *command* という名前にします。
- 新しくできた *command* を選択した状態にし、 右の欄で 「(既定)」 をダブルクリックして *php.exe* への パスを入力します。例、 *c:¥php¥php.exe -q %1* （*%1* を忘れずに入力すること）。
- Regedit を終了します。
- Windows 上で PWS を使用している場合、レジストリを再ロードするために 再起動します。

ここまでの作業で、PWSおよび IIS 3用のシステムインストールは完了しました。 IIS 3 用の良くできた » 設定ツール が Steven Genusa により 配布されており、これを使用してスクリプトマッピングを設定することができます。

Apache 1.3.x （Microsoft Windows 用）

このセクションでは、Microsoft Windows 上の Apache 1.3.x で PHP を使用する場合について説明します。 Apache 2 で PHP を使用する場合 については別に記載されています。

> **注意**: まず始めに、マニュアルインストールの手順 をお読みください。

PHP を Windows 上の Apache 1.3.x で動作させるには、2種類の方法が あります。一つは、CGI バイナリ (PHP 4 の場合 *php.exe*、 PHP 5 の場合 *php-cgi.exe*) を使用する方法、もう一つ は Apache モジュール DLL を使用する方法です。どちらの場合も *httpd.conf* を編集して Apache が PHP を利用できるようにした後、 Apache サーバを再起動する必要があります。

Windwos 環境向けの SAPI モジュールはかなり安定してきているため、 透過性と安全性の面からも CGI バイナリより SAPI モジュールの 使用を推奨します。

Apache で PHP を使うように設定する手順にはいくつかのバリエーションがありますが、 いずれも入門者にもできるほど簡単です。設定ディレクティブに関する詳細については、 Apache のドキュメントも参照してください。

設定ファイルを変更した後、サーバの再起動を忘れずに行ってください。 Apache を Windows サービスとして実行しているなら、**NET STOP APACHE** とした後 **NET START APACHE** とします。 もしくは、スタートメニューのショートカットからも再起動できる場合もあります。

> **注意**: Windows 上で Apache 設定ファイルにパスの値を追加する際、例えば *c:¥directory¥file.ext* に含まれるすべてのバックスラッシュは *c:/directory/file.ext* のように前向きスラッシュに変換する必要があります。また、ディレクトリを表す際には最後にスラッシュをつけなければなりません。

Apache モジュールの使用

以下の行を Apache の *httpd.conf* ファイルに追加してください。

Example#1 Apache 1.3.x でモジュール版の PHP を使用する場合の設定

以下では、PHP は *c:¥php* にインストールされていると仮定します。 そうでない場合はパスを適当に修正してください。

PHP 4 の場合

```
# LoadModule セクションの最後に追加
# sapi ディレクトリからこのファイルをコピーするのを忘れないこと!
LoadModule  php4_module  "C:/php/php4apache.dll"

# AddModule セクションの最後に追加
AddModule  mod_php4.c
```

PHP 5 の場合

```
# LoadModule セクションの最後に追加
LoadModule  php5_module  "C:/php/php5apache.dll"

# AddModule セクションの最後に追加
AddModule  mod_php5.c
```

共通

```
# <IfModule mod_mime.c> 条件節の内部に追加
AddType  application/x-httpd-php  .php

# .phps ファイルを構文ハイライト表示する場合に追加
AddType  application/x-httpd-php-source  .phps
```

CGI バイナリの使用

マニュアルインストールの手順 のセクションにある通り、PHP パッケージを *C:¥php¥* に展開したならば、以下を Apache の設定ファイルに追加すれば CGI バイナリを利用可能にできます。

Example#2 Apache 1.3.x で CGI 版の PHP を使用する場合の設定

```
ScriptAlias  /php/  "c:/php/"
AddType  application/x-httpd-php  .php

# PHP 4 の場合
Action  application/x-httpd-php  "/php/php.exe"

# PHP 5 の場合
Action  application/x-httpd-php  "/php/php-cgi.exe"

# php.ini の場所を設定
SetEnv  PHPRC  C:/php
```

上記の 2 行目は、コメントアウトされた状態で *httpd.conf* に記載されている場合があります。また、*c:/php/* は、実際のパスにあわせて修正してください。

警告

CGI としてセットアップすると、サーバは様々な攻撃を受ける可能性があります。これらの攻撃からサーバを守る方法については、 CGI セキュリティ のセクションを参照してください。

PHP ソースの構文ハイライト表示については、モジュール版にあるような便利な オプションはありません。Apache で CGI 版の PHP を使用している場合、 highlight_file() 関数を使用してください。 普通に PHP スクリプトを作成し、次のようにコードを記述すれば、構文ハイライト表示が可能です。 *<?php highlight_file('ハイライト表示するファイル'); ?>*

Apache 2.0.x （Microsoft Windows 用）

このセクションでは、Microsoft Windows 上の Apache 2.0.x で PHP を使用する場合について説明します。 Apache 1.3 で PHP を使用する場合 については別に記載されています。

> **注意**: まず始めに、マニュアルインストールの手順 をお読みください。

> **注意**: **Apache 2.2.x のサポート** Apache 2.2.x を利用しているかたも以下のドキュメントを使用できますが、 DLL ファイルの名前を *php5apache2_2.dll* に読みかえてください。このファイルは PHP 5.2.0 以降にしか含まれません。 » http://snaps.php.net/ も参照ください。

警告

Apache2 の MPM マルチスレッドモードを実運用環境で使用することは推奨されません。代わりに prefork MPM または Apache1 を使用してください。その理由については、 マルチスレッド版 MPM の Apache2の FAQ エントリを参照してください。

» Apache ドキュメンテーション を参照し、Apache 2.0.x の基本的な事項について理解しておくことを 強く推奨します。また、以下の解説を読む前に、Apache 2.0.x に関する » Windows 固有の情報 についても参照すると良いでしょう。

> **注意**: **PHP と Apache 2.0.x の互換性に関する注意** PHP の以下のバージョンは、Apache 2.0.x の最新版での動作が確認されています。
>
> - PHP 4.3.0 およびそれ以降 (» http://www.php.net/downloads.php で入手可能)
> - 最新の安定開発版。 ソースコード » http://snaps.php.net/php5-latest.tar.gz を入手、または Windows 用のバイナリ » http://snaps.php.net/win32/php5-win32-latest.zip をダウンロードしてください。
> - プレリリース版を » http://qa.php.net/ から ダウンロード可能です。
> - » anonymous CVS から PHP を 入手することも可能です。
>
> 以上のバージョンの PHPは、Apache 2.0.40 以降と互換性があります。
> Apache 2.0 *SAPI* のサポートは PHP 4.2.0 で開始されました。 PHP 4.2.3 は Apache 2.0.39 で動作します。PHP 4.2.3 を Apache の他のバージョンと 組み合わせて使用しないでください。 PHP 4.3.0 もしくはそれ以降のバージョンの PHP を 最新版の Apache2 と組み合わせて使用することが推奨されます。
> ここで挙げたバージョンの PHP は、Apache 1.3.x でも動作します。

警告

Apache 2.0.x は Windows NT 4.0, Windows 2000 および Windows XP で動作するように設計されています。現時点では、Windows 9x のサポートは 不完全です。

最新の » Apache 2.0.x と、対応する バージョンの PHP をダウンロードしてください。 マニュアルインストールの手順 を実施したら、引き続き以下のとおり PHP と Apache の設定を行ってください。

PHP を Windows 上の Apache 2.0.x で動作させるには、2種類の方法が あります。一つは、CGI バイナリを使用する方法、もう一つ は Apache モジュール DLL を使用する方法です。どちらの場合も *httpd.conf* を編集して Apache が PHP を利用できるようにした後、 Apache サーバを再起動する必要があります。

> **注意**: Windows 上で Apache 設定ファイルにパスの値を追加する際、例えば *c:¥directory¥file.ext* に含まれるすべてのバックスラッシュは *c:/directory/file.ext* のように前向きスラッシュに変換する必要があります。また、ディレクトリを表す際には最後にスラッシュをつけなければなりません。

CGI バイナリの使用

CGI 版のバイナリを使用する場合は、以下の行を Apache 設定ファイル *httpd.conf* へ追加してください。

Example#1 Apache 2.0 で CGI 版の PHP を使用する場合の設定

```
ScriptAlias  /php/  "c:/php/"
AddType  application/x-httpd-php  .php

# PHP 4 の場合
Action  application/x-httpd-php  "/php/php.exe"

# PHP 5 の場合
Action  application/x-httpd-php  "/php/php-cgi.exe"
```

警告

CGI としてセットアップすると、サーバは様々な攻撃を受ける可能性があります。これらの攻撃からサーバを守る方法については、 CGI セキュリティ のセクションを参照してください。

Apache モジュールの使用

Apache 2.0 でモジュール版の PHP を使用するには、 以下の行を Apache 設定ファイル *httpd.conf* に追加してください。

Example#2 Apache 2.0 でモジュール版の PHP を使用する場合の設定

```
# PHP 4 の場合
LoadModule  php4_module  "c:/php/php4apache2.dll"
# sapi ディレクトリから php4apache2.dll をコピーするのを忘れないこと!
AddType  application/x-httpd-php  .php

# PHP 5 の場合
LoadModule  php5_module  "c:/php/php5apache2.dll"
AddType  application/x-httpd-php  .php

# php.ini の場所を設定
PHPIniDir  "C:/php"
```

> **注意**: *c:/php/* は実際のパスにあわせて修正してください。 LoadModule ディレクティブでは必ず *php4apache2.dll* または *php5apache2.dll* を指定します。 *php4apache.dll* および *php5apache.dll* は Apache 1.3.x 用です。
>
> **注意**: コンテントネゴシエーションを使用する場合は、 関連の FAQ を参照してください。

警告

バージョンの異なる PHP の DLL ファイルを混ぜて使わないでください。 DLL と PHP 拡張モジュールは、同一バージョンのファイルだけが 組み合わせて使用可能です。

Sun, iPlanet, Netscape サーバ（Microsoft Windows 用）

このセクションでは、Windows 上の Sun Java System Web Server, Sun ONE Web Server, iPlanet and Netscape server で PHP を使用する場合について説明します。

PHP 4.3.3 より、NSAPI モジュール を使って 独自エラーページ およびファイル一覧表示ページの生成 が可能です。 Apache 互換の関数も追加されています。 また、これらの WEB サーバ専用の機能については、 「サブリクエストに関する注意」 をお読みください。

Sun, iPlanet, Netscape サーバ で CGI 版の PHP を使用する

CGI 版の PHP を使用する場合は、以下のようにしてください。

- *php4ts.dll* をシステムルート （Windows がインストールされているフォルダ）にコピーします。
- コマンドラインからファイルの関連付けを行います。 次の 2 行をタイプしてください。

  ```
  assoc  .php=PHPScript
  ftype  PHPScript=c:¥php¥php.exe  %1  %*
  ```

- Netscape Enterprise Administration Server で、ダミーの shellcgi ディレクトリを作成し、その後すぐに削除します （このステップにより 5 つの重要な行が obj.conf に作成され、 Web サーバが shellcgi スクリプトを扱えるようになります。）
- Netscape Enterprise Administration Server で 新しい mime type を作成します。(Category: type, Content-Type: magnus-internal/shellcgi, File Suffix:php)
- PHP を実行するすべての Web サーバインスタンスで上記を実行してください。

CGI 版の PHP を使用する場合の詳細な説明は » http://benoit.noss.free.fr/php/install-php.html を参照してください。

Sun, iPlanet, Netscape サーバ で NSAPI 版の PHP を使用する

NSAPI 版の PHP を使用する場合は、以下のようにしてください。

- *php4ts.dll* をシステムルート （Windows がインストールされているディレクトリ）にコピーする。
- コマンドラインからファイルの関連付けを行います。 次の 2 行をタイプしてください。

  ```
  assoc  .php=PHPScript
  ftype  PHPScript=c:¥php¥php.exe  %1  %*
  ```

- Netscape Enterprise Administration Server において 新しい mime type を作成します。 (Category: type, Content-Type: magnus-internal/shellcgi, File Suffix:php)
- *magnus.conf* (サーバ>= 6の場合) または *obj.conf* (サーバ< 6の場合)を編集し、 以下の行を追加します。 この行は *mime types init* の後に記述する必要があります。

  ```
  Init  fn="load-modules"  funcs="php4_init,php4_execute,php4_auth_trans"  shlib="c:/php/sapi/php4nsapi.dll"
  Init  fn="php4_init"  LateInit="yes"  errorString="Failed  to  initialise  PHP!"  [php_ini="c:/path/to/php.ini"]
  ```

  (PHP >= 4.3.3) *php_ini* パラメータはオプションですが、 これを指定することにより、Web サーバの設定ファイルがあるフォルダに *php.ini* を置くことが可能になります。

- *obj.conf* のデフォルトオブジェクトを設定します （仮想サーバの場合、*vserver.obj.conf* のクラス [SunONE 6.0]。） *< Object name="default" >* セクションに 次の行を追加してください。この行は、 'ObjectType' の後、'AddLog' の前に記述してください。

```
Service  fn="php4_execute"  type="magnus-internal/x-httpd-php"  [inikey=value  inikey=value  ...]
```

(PHP >= 4.3.3) 追加のパラメータとして、いくつかの特別な *php.ini* 値を 追加することができます。例えば、コンテキスト *php4_execute* をコールする時に *docroot="/path/to/docroot"* を設定することができます。 論理値の場合、 *"On","Off",...*（これは正 しく動作しません）ではなく、0/1 を値として使用してください。 例えば、 *zlib.output_compression="On"* ではなく、 *zlib.output_compression=1* とします。

- 以下は、（*cgi-bin* ディレクトリ のように）PHP スクリプトのみからなるディレクトリを設定したい場合にだけ必要です。

```
<Object  name="x-httpd-php">
ObjectType  fn="force-type"  type="magnus-internal/x-httpd-php"
Service  fn=php4_execute  [inikey=value  inikey=value  ...]
</Object>
```

これにより、Administration Server に特定のディレクトリを設定し、 これをスタイル *x-httpd-php* に割り付けることができ ます。このディレクトリの中にあるすべてのファイルは PHP スクリプト として実行されます。 これは、ファイルの名前を *.html* に変更し、PHP が使用されている事を隠したい場合に有用です。

- Web サービスを再起動して設定を反映させます。
- PHP を実行するすべての Web サーバインスタンスで上記を行ってください。

注意: NSAPI 版の PHP の使用についてのより詳細な説明は » http://benoit.noss.free.fr/php/install-php4.html をご覧ください。

注意: PHP が使用するスタックサイズは WEB サーバの設定に依存します。 非常に大きい PHP スクリプトを実行させた際にクラッシュする場合は、 Administration Server でスタックサイズ（"MAGNUS EDITOR"） を大きくすると良いでしょう。

CGI 環境変数と php.ini の変更

Sun JSWS/Sun ONE WS/iPlanet/Netscape がマルチスレッドの WEB サーバだという事が PHP スクリプトを書く際に重要になります。すべてのリクエストは同一の （WEB サーバ自体の）プロセス空間で実行されます。*PATH_INFO* や *HTTP_HOST* などの CGI 変数を取得する場合、 PHP 3.x で行っていたような古い方法、つまり getenv() 関数を使用する方法や他の同等な方法（グローバル変数の登録機能、 *$_ENV* 等）を使うのは正しい方法ではありません。WEB サーバの環境変数をただ単に取得すると、 正しい CGI 変数は得られません。

> **注意**: なぜ正しくない CGI 変数が登録されているのでしょうか？
> それは、WEB サーバのプロセスを Administration Server から起動させる際、 WEB サーバの起動スクリプトが CGI スクリプトとして実行されるためです。したがって、 起動された WEB サーバの環境変数には CGI 変数も含まれることになります。 Administration Server 以外から WEB サーバを起動してみればこのことをテストできるでしょう。 ルートユーザでコマンドラインを使って、手動で起動してみると、CGI 変数らしき 環境変数が登録されないことが確認できると思います。

PHP 4.x のスクリプトで CGI 変数を取得する場合は、スーパーグローバル *$_SERVER* を用いるのが正しい方法です。また、*$HTTP_HOST* などを使う古いスクリプトを使用する場合は、*php.ini* で *register_globals* をオンにし、変数のパースの順番 (*variables_order*) を変更してください （*"E"* を削除。環境変数を読み込む必要は無いため。）

```
variables_order  =  "GPCS"
register_globals  =  On
```

独自エラーページおよびファイル一覧表示ページ (PHP >= 4.3.3)

PHP を使って、*"404 Not Found"* などに対するエラーページを生成することが できます。オーバーライドしたいエラーページすべてに対して、以下の行を *obj.conf* 中のオブジェクトに追加してください。

```
Error  fn="php4_execute"  code=XXX  script="/path/to/script.php"  [inikey=value  inikey=value...]
```

ここで、*XXX* は HTTP のエラーコードです。 追加したものと干渉する *Error* ディレクティブは削除してください。 発生するすべてのエラーに対応するページを設定したい場合は、 *code* パラメータを省略してください。スクリプトで HTTP ステータス コードを取得するには、*$_SERVER['ERROR_TYPE']* を使用します。

独自のファイル一覧表示ページを PHP を使って生成することも可能です。 ファイル一覧表示を行う PHP スクリプトを作成し、*obj.conf* の *type="magnus-internal/directory"* の行に 書かれているデフォルトのサービスを以下のように置き換えます。

```
Service  fn="php4_execute"  type="magnus-internal/directory"  script="/path/to/script.php"  [inikey=value  inikey=value...]
```

エラーページ、ファイル一覧表示ページのどちらでも、元の URI および 変換後の URI は、それぞれ、*$_SERVER['PATH_INFO']* および *$_SERVER['PATH_TRANSLATED']* に格納されています。

nsapi_virtual() およびサブリクエストに関する注意 (PHP >= 4.3.3)

NSAPI モジュールは、現在、nsapi_virtual() 関数 (エイリアス: virtual()) をサポートしており、 WEB サーバへサブリクエストを行い、結果を WEB ページへ挿入することができます。 問題としては、この関数は 文書化されていない NSAPI ライブラリの機能を使用して いることにあります。

Unix では、モジュールは自動的に必要な関数群を探し、可能であればそれらの関数を 使用するため、特に問題はありません。もし使用可能でなければ、nsapi_virtual() は 使用不可となります。

Windows では、DLL の扱いに制限があるため、自動認識の使用には 最新の *ns-httpdXX.dll* ファイルが必要です。 バージョン 6.1 までテストが行われています。もし、より新しい Sun サーバを使う場合は、 自動認識が動作せず、nsapi_virtual() が使用不可となる可能性が あります。

もしそういった事になった場合は、 *magnus.conf/obj.conf* の *php4_init* へ以下のパラメータを追加してください。

```
Init  fn=php4_init  ...  server_lib="ns-httpdXX.dll"
```

ここで、*XX* は接続する DLL のバージョン番号です。 番号を調べるには、サーバのルートで、対応する名前のファイルを探してください。 おそらく、最もファイルサイズの大きい DLL が探しているファイルでしょう。

ステータスは phpinfo() 関数を使って確認できます。

> **注意**: nsapi_virtual() サポートは「実験的」な機能です。

OmniHTTPd サーバ

このセクションでは、Windows 上の » OmniHTTPd で PHP を使用する場合について説明します。

> **注意**: まず始めに、マニュアルインストールの手順 をお読みください。

警告

CGI としてセットアップすると、サーバは様々な攻撃を受ける可能性があります。これらの攻撃からサーバを守る方法については、 CGI セキュリティ のセクションを参照してください。

PHP を OmniHTTPd で動作するように設定するには以下の手順を行う必要があります。 以下は CGI 版の設定です。OmniHTTPd では SAPI もサポートされていますが、 ISPAI モジュール版の PHP を使用すると不安定なことが テストの結果明らかになっています。

> **注意**: **CGI 版 PHP を使用する場合** cgi.force_redirectに関するFAQ に重要な情報がありますのでお読みください。このディレクティブが *0* にセットされている必要があります。

1. OmniHTTPd サーバをインストールします。
2. システムトレイにある青い OmniHTTPd アイコンを 右クリックし、*Properties*を選択します。
3. *Web Server Global Settings* をクリックします
4. *virtual = .php ¦ actual = c:¥path-to-php-dir¥php.exe* と *External* タブに入力します。 入力したら、*Add* ボタンを押してください。
5. *virtual = wwwserver/stdcgi ¦ actual = .php* と *Mime* タブに入力します。 入力したら、*Add* ボタンを押してください。
6. *OK* をクリックします。

PHP に関連付けたい拡張子すべてに対して手順 2〜6 を繰り返してください。

> **注意**: PHP サポートを有効にしてビルドされている OmniHTTPd パッケージがあります。 その場合、セットアップの時にカスタムセットアップを選択し PHP コンポーネントのチェックをはずしてください。 PHP4 のベータ版を含んでいる OmniHTTPdパッケージもありますので、 PHP のビルトインサポートを選択すべきではありません。 自分で PHP を別途インストールすべきです。サーバが既にマシン上に インストールされている場合には、手順 4,5 において リプレースボタンを押して、正しい情報を新しくセットしてください。

Sambar サーバ（Microsoft Windows 用）

このセクションでは、Windows 上の » Sambar Server で PHP を使用する場合について説明します。

> **注意**: まず始めに、マニュアルインストールの手順 をお読みください。

以下の手順は、Windows上 の Sambar サーバで ISAPI モジュール版の PHP を使うように 設定する方法を解説するものです。

- Sambar をインストールしたフォルダ (の config フォルダ)にある *mappings.ini* という名前のファイルを探します。
- *mappings.ini* を開き、以下の行を *[ISAPI]* の下に追加します。

  Example#1 Sambar での ISAPI の設定

  ```
  # PHP 4 用
  *.php  =  c:¥php¥php4isapi.dll

  # PHP 5 用
  *.php  =  c:¥php¥php5isapi.dll
  ```

  (PHP が *c:¥php* にインストールされていると 仮定しています。)

- 変更を有効にするために Sambar サーバを再起動します。

  > **注意**: PHP で ネットワーク上の他のコンピュータにあるリソースと通信したい場合は、 Sambar Server サービスが使用するアカウント

を変更する必要があります。 Sambar Server サービスが使用しているデフォルトのアカウントは LocalSystem で、これはリモートのリソースにアクセスできません。 アカウントを変更するには、コントロールパネルの管理ツールから 「サービス」を使用します。

Xitami（Microsoft Windows 用）

このセクションでは、Windows 上の » Xitami で PHP を使用する場合について説明します。

> **注意**: まず始めに、マニュアルインストールの手順 をお読みください。

以下の手順は、Windows 上の Xitami でPHP の CGI 版バイナリを動作させる際の 設定方法です。

> **注意**: **CGI 版 PHP を使用する場合** cgi.force_redirectに関するFAQ に重要な情報がありますのでお読みください。このディレクティブが *0* にセットされている必要があります。 *$_SERVER['PHP_SELF']* を使用する場合は、 cgi.fix_pathinfo をオンにする必要があります。

警告

CGI としてセットアップすると、サーバは様々な攻撃を受ける可能性があります。これらの攻撃からサーバを守る方法については、 CGI セキュリティ のセクションを参照してください。

- Web サーバが正常に動作していることを確認し、 ブラウザで Xitami 管理用コンソール（通常は *http://127.0.0.1/admin*）を参照して、「Configuration」を クリックします。
- 「Filters」を選択し、php にパースさせるファイルの拡張子 (.php など) を 「File extensions (.xxx) 」フィールドに入力します。
- 「Filter command or script」に、CGI 版 PHP バイナリのパスと名前 (たとえば、*c:¥php¥php.exe*) を入力します。
- 「Save」アイコンを押します。
- 変更点を反映するためにサーバを再起動します。

ソースからのビルド

この章では、Windows 上でマイクロソフトのツールを用いて PHP をソースから コンパイルする方法を説明します。PHP を cygwin でコンパイルする場合は Unix システムへのインストール を参照ください。

Build Environment

PHP のコンパイルとビルドにはマイクロソフトの開発環境が必要です。 以下の環境をサポートしています。

- Microsoft Visual C++ 6.0 (公式)
- Microsoft Visual C++ .NET
- Microsoft Visual C++ 2005, Windows Platform SDK および .NET Framework SDK (現在)

公式の Windows 版は VC6 (Microsoft Visual C++ 6.0) で作成していますが、これは現在 Microsoft のウェブサイトからはダウンロードできません。 フリーで Windows 版の PHP をビルドしたい方は、Microsoft Visual C++ 2005 Express Edition と補助コンポーネントを使用してください。

Microsoft Visual C++ 2005 Express の設定

> **注意**: これらのコンポーネントは非常に大きなものであり、 すべてあわせると 1 ギガバイト以上のディスク容量が必要となります。

Microsoft Visual C++ 2005 Express の設定は少し複雑で、 3 つの別々のパッケージをインストールする必要があり、 互換性にかかわる変更もあります。 これらの 3 つのプログラムがインストールされた場所を覚えておくようにしましょう。 以下のプログラムをダウンロードしてインストールします。

- » Microsoft Visual C++ 2005 Express
- » Microsoft Windows Server 2005 Platform SDK
- » .NET Framework 2.0 Software Development Kit

インストールしたら、その後にいくつかの作業があります。

- » MSVC 2005 Express は、 Windows Platform SDK を使用するように設定する必要があります。 PHP でグラフィカルなユーザインターフェイスを使用しないのなら、 次のステップを実行する必要はありません。
- Windows Platform SDK には *WinResrc.h* というファイルが含まれています。通常、このファイルは SDK をインストールしたディレクトリの下の *Include* フォルダにあります。 これをコピーして *winres.h* という名前に変更します。PHP はこの名前のファイルを使用します。

最後に、MSVC 2005 Express をコマンドラインから使用する場合は、 いくつか環境変数を設定する必要があります。 *vsvars32.bat* というファイルが、通常は *C:¥Program Files¥Microsoft Visual Studio 8¥Common7¥Tools* にあります (見つからなければ検索してください)。 このファイルには、*PATH* や *INCLUDE* そして *LIB* などの環境変数の宣言が書かれており、 それぞれ SDK のインストールディレクトリ配下の *bin*、*include* そして

lib ディレクトリを含めるようになっています。

> **注意**: .NET SDK のパスは既に *vsvars32.bat* に含まれているでしょう。 というのも、この SDK は自身のインストール先を Microsoft Visual C++ 2005 Express と同じディレクトリにするからです。

ライブラリ

ダウンロードしたファイルを展開するには、ZIP 展開ユーティリティが 必要です。Windows XP 以降のバージョンでは、ZIP 展開機能が すでに組み込まれています。

始める前に、ダウンロードしなければならないものがいくつかあります。

- » http://www.php.net/extra/win32build.zip から win32 ビルドツール
- » http://www.php.net/extra/bindlib_w32.zip から PHP が使用する DNS ネームリゾルバのソース。 これは *win32build.zip* に含まれる *resolv.lib* の代用です。
- PHP を Apache モジュールとしてコンパイルする場合には » Apache のソースも 必要です。

最後に、PHP 自体のソースが必要となります。最新の開発版を » anonymous CVS から、または » スナップショット、あるいは 最新のリリース版の » ソース tar ボールをダウンロードします。

ファイルの配置

必要なパッケージをすべてダウンロードしたら、 ファイルを適切な位置に展開しなければなりません。

- すべてのファイルを展開するための作業ディレクトリを作成します。 例えば *C:¥work*
- 作業ディレクトリ (*C:¥work*) 配下に *win32build* ディレクトリを作成し、そこに *win32build.zip* を展開します。
- 作業ディレクトリ (*C:¥work*) 配下に *bindlib_w32* ディレクトリを作成し、そこに *bindlib_w32.zip* を展開します。
- 作業ディレクトリ (*C:¥work*) 配下に PHP のソースコードを展開します。
- 必要となるライブラリをビルド (あるいはもしバイナリが入手可能なら それをダウンロード) し、ヘッダおよびライブラリをそれぞれ *C:¥work¥win32build¥include* および*C:¥work¥win32build¥lib* ディレクトリに配置します。
- cygwin 版の bison および flex をインストールしていない場合は、 configure スクリプトがこれらのツールを見つけられるようにするために *C:¥work¥win32build¥bin* ディレクトリを PATH に追加する必要があります。

上記の手順を行えば、ディレクトリ構造は以下のようになっているはずです。

```
+--C:¥work
|   |
|    +--bindlib_w32
|   |   |
|   |   +--arpa
|   |   |
|   |   +--conf
|   |   |
|   |   +--...
|   |
|    +--php-5.x.x
|   |   |
|   |   +--build
|   |   |
|   |   +--...
|   |   |
|   |   +--win32
|   |   |
|   |   +--...
|   |
|    +--win32build
|   |   |
|   |   +--bin
|   |   |
|   |   +--include
|   |   |
|   |   +--lib
```

» Cygwin を使用していない場合は、 *C:¥usr¥local¥lib* ディレクトリを作成した上で *C:¥work¥win32build¥bin* にある *bison.simple* を *C:¥usr¥local¥lib* にコピーする必要があります。

> **注意**: PEAR およびそのコマンドラインインストーラを使用したい場合は、 CLI-SAPI が必須となります。PEAR およびそのインストーラについての 詳細な情報は、» PEAR web サイトのドキュメントを参照ください。

resolv.lib のビルド

resolv.lib ライブラリをビルドしなければなりません。 デバッグシンボルを有効にするか (bindlib - Win32 Debug) 否か (bindlib - Win32 Release) を決定してください。このとき、 どちらを選択したかを覚えておいてください。というのは、デバッグモードを 有効にした場合は、PHP もデバッグモードでビルドしないとリンクできなくなる からです。 適切な設定を用いてビルドします。

- GUI ユーザの場合、*C:¥work¥bindlib_w32¥bindlib.dsw* をダブルクリックして VC++ を起動します。 そして Build=>Rebuild All を選択し

ます。
- コマンドラインユーザの場合、C++ 環境変数が設定されているか、あるいは **vcvars.bat** を既に実行したかを確認してください。 そして、以下のコマンドを実行します。
  - **msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"**
  - **msdev bindlib.dsp /MAKE "bindlib - Win32 Release"**

この時点で、使用可能な *resolv.lib* が *C:¥work¥bindlib_w32¥Debug* または *Release* サブディレクトリのどちらかに存在します。 このファイルを *C:¥work¥win32build¥lib* ディレクトリの同名のファイルに上書きコピーしてください。

新しいビルドシステムを使用して PHP をビルドする [PHP >=5 のみ]

この章では、新しいビルドシステムを使用して PHP >=5 をコンパイルする 方法を説明します。これは CLI ベースのものであり、Unix での PHP のビルドシステムに非常に似ています。

> **注意**: PHP 4 ではこの方式を使用することはできません。代わりに ソースからのビルド を参照ください。

はじめる前に、ソースからのビルド をよく読んだ上で » Libxml や » ICU (これは PHP >= 6 で必要です) のような必要なライブラリをすべてビルドしておいてください。

まず最初に、スタートメニューから Visual Studio コマンドプロンプトを 開きます。通常のコマンドプロンプトではうまく動作しません。 おそらく、必要な環境変数が設定されていないからでしょう。 次に、たとえば **cd C:¥work¥php-5.x.x** などのように 入力して PHP のソースディレクトリに移動します。 いよいよ PHP の設定が始まります。

次に、バッチファイル **buildconf** を実行します。 これは、*config.w32* ファイルの内容をもとにして 設定スクリプトを作成します。デフォルトでは、このコマンドは 以下のディレクトリから *config.w32* を探します。 *pecl; ..¥pecl; pecl¥rpc; ..¥pecl¥rpc*。 PHP 5.1.0 以降では、引数 *--add-modules-dir* を指定することで、この振る舞いを変更可能です (例えば **cscript /nologo win32/build/buildconf.js --add-modules-dir=../php-gtk2 --add-modules-dir=../pecl**)。

次のステップは、出来上がった設定スクリプトの実行です。 使用可能な設定オプションの一覧を見るには、 **cscript /nologo configure.js –help** と入力します。それらのオプションを有効/無効にすることを決めたら、 たとえば **cscript /nologo configure.js –disable-foo –enable-fun-ext** のように指定します。 *--enable-foo=shared* のようにすると、'foo' を共有モジュールとしてビルドし、動的に読み込まれるようにします。

最後に、コンパイルです。これは、ただ単に **nmake** というコマンドを実行するだけです。できあがったファイル (たとえば .exe や .dll) は、*Release_TS* あるいは *Debug_TS* ディレクトリに 配置されます (スレッドセーフ環境でビルドされた場合)。 それ以外の場合は *Release* あるいは *Debug* ディレクトリに配置されます。

オプションとして、PHP のテストスイートを実行することもできます。 この場合は **nmake test** と入力します。 特定のテストだけを実行させたいのなら、変数 'TESTS' を指定します (例 **nmake /D TESTS=ext/sqlite/tests test** - これは sqlite のテストのみを行います)。 コンパイル時に作成されたファイルを削除するには、 **nmake clean** コマンドを使用します。

スナップショットをビルドする際に非常に有用な設定オプションが、 *--enable-snapshot-build* です。 これは、新たなコンパイルモード (**nmake build-snap**) を作成します。このモードは、使用可能なすべての拡張モジュールを (デフォルトでは共有モジュールとして) ビルドしようとしますが、個々の拡張モジュールや SAPI の構築の際のエラーは無視されます。

DSW ファイルを使用して PHP をビルドする [PHP 4]

DSW ファイルを使用して PHP をコンパイルする方法は、PHP 5 では サポートされていません。 より柔軟なシステムを 使用することができます。 今でもここで説明されている方法を使用することは可能です。しかし 今後はメンテナンスの頻度が下がるため、コンパイル時に問題が 発生する可能性があることを覚えておきましょう。 Windows で PHP 4 をコンパイルする場合は、ここで説明されている方法が 唯一のものです。

MVC ++ の設定

最初のステップは、MVC++ でコンパイルするための準備です。 Microsoft Visual C++ を起動し、メニューから ツール => オプション を選択します。 ダイアログで、ディレクトリタブを選択します。 実行ファイル、インクルードファイル、ライブラリファイル の設定を順に変更し、以下のようにします。

- 実行ファイル: *C:¥work¥win32build¥bin*, Cygwin ユーザの場合: *C:¥cygwin¥bin*
- インクルードファイル: *C:¥work¥win32build¥include*
- ライブラリファイル: *C:¥work¥win32build¥lib*

コンパイル

まず始めは、標準的な CGI バージョンをビルドしてみると良いでしょう。

- GUI ユーザの場合、VC++ を起動し、ファイル => ワークスペースを 開くを選び、*C:¥work¥php-4.x.x¥win32¥php4ts.dsw* を選択してください。続いて、構築=>アクティブな 構成を設定 を選び、*php4ts - Win32 Debug_TS* あるいは *php4ts - Win32 Release_TS* から好きなほうを 選択します。最後に、構築=>すべて構築 を選択します。
- コマンドライン版ユーザの場合、C++ 用の環境変数が定義されている かどうか、もしくは、**vcvars.bat** を実行済みか どうか確認してから、*C:¥work¥php-4.x.x¥win32* ディレクトリから次の内のいずれかを実行してください。

- **msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"**
- **msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"**
- ここまでの手順で、サブディレクトリ *C:¥work¥php-4.x.x¥Debug_TS* または *Release_TS* に利用可能な *php.exe* が作成されているはずです。

main/config.win32.h ファイルを編集すると ビルドプロセスでの細かい設定を行えます。 例えば *php.ini* のデフォルトの位置を変えたり、組み込む拡張モジュールや 拡張モジュールのデフォルトの位置を変えることができます。

次に、PHP をコマンドラインから使用する ための CLI バージョンを構築しましょう。 *php4ts_cli - Win32 Debug_TS* または *php4ts_cli - Win32 Release_TS* を選択すること以外は CGI バージョンのビルド手順と同様です。コンパイルが成功すると *Release_TS¥cli¥* または *Debug_TS¥cli¥* ディレクトリの どちらかに *php.exe* が作成されます。

Microsoft IIS 上で PHP アプリを実行するための SAPI モジュール (*php4isapi.dll*) をビルドするには、 アクティブな設定を *php4isapi-whatever-config* にし、お望みの dll をビルドしてください。

Windows 用 PHP 拡張モジュール

Windows に PHP を インストールし、Web サーバの設定ができたら、 次は PHP 拡張モジュールを使うための設定です。 *php.ini* を使って PHP が起動時にロードする拡張モジュールを設定することができます。 もしくは、スクリプトの中で dl() 関数を使用することにより、 拡張モジュールを動的にロードすることも可能です。

PHP 拡張モジュールの DLL には、ファイル名の前に 'php_' が付いています。

Windows 版の PHP には、多くの拡張モジュールが *組み込まれています*。これらの関数を使用する際には、 追加の DLL ファイルや extension ディレクティブの設定は *不要です*。 追加の DLL が必要となる (あるいはかつて必要だった) 拡張モジュールについては、 Windows 版 PHP 拡張モジュール の表にまとめてあります。以下にあげられている拡張モジュールは、すでに PHP に組み込まれています。

PHP 4 (PHP 4.3.11 時点): BCMath、 Caledar、 COM、 Ctype、 FTP、 MySQL、 ODBC、 Overload、 PCRE、 Session、 Tokenizer、 WDDX、 XML そして Zlib

PHP 5 (PHP 5.0.4 時点) では、さらに以下が組み込まれています。 DOM、 LibXML、 Iconv、 SimpleXML、 SPL そして SQLite。 また、以下は組み込まれなくなりました。 MySQL および Overload。

PHP が拡張モジュールを探すデフォルトの場所は PHP 4 の場合 *C:¥php4¥extensions*、 PHP 5 の場合 *C:¥php5* です。 変更するには *php.ini* ファイルを編集してください。

- extension_dir を拡張モジュール があるフォルダに変更する必要があります。 *php_*.dll*ファイルをそこに置いてください。 例えば次のようになります。

  ```
  extension_dir  =  c:¥php¥extensions
  ```

- *php.ini* には、多くの拡張モジュール名がコメントアウトされた状態で記載済みです。 それらの拡張モジュールを有効にするには、*php.ini* 上の *extension=php_*.dll* の行をアンコメント（行頭の ; を削除する）してください。

  Example#1 Windows 版の PHP で Bzip2 拡張モジュールを有効にする

  ```
  // この行を以下から
  ;extension=php_bz2.dll

  // このように変更する
  extension=php_bz2.dll
  ```

- 拡張モジュールによっては、その動作に外部 DLL が必要な場合があります。 配布パッケージには、一部の外部 DLL がバンドルされています。 PHP 4 の場合は *C:¥php¥dlls¥*を、 PHP 5 の場合は親フォルダを参照ください。ただし、必要な外部 DLL がバンドルされていないモジュールもあり、たとえば、Oracle モジュール (*php_oci8.dll*) は、非バンドルの DLL を必要とします。 PHP 4 をインストールする場合、バンドルされた DLL を *C:¥php¥dlls* からメインのフォルダ *C:¥php* へコピーしてください。 また、忘れずに *C:¥php* をシステムパスに 追加してください（その方法は、別途 FAQ に記載されています。）

- これらの DLLs の中には、PHP の配布ファイルに含まれていないものもあります。 詳細は、それぞれの拡張モジュールのドキュメントを参照ください。 また、PECL についての詳細は、マニュアルの PECL 拡張モジュールのインストール という節を参照ください。多くの PHP 拡張モジュールが PECL に移行しつつあり、これらは 個別にダウンロード しなければなりません。

  注意: PHP をサーバモジュールとして実行している場合は、Webサーバを再起動しないと、 *php.ini* の設定が反映されません。Web サーバの再起動を忘れずに行ってください。

以下の表は、使用可能な拡張モジュールと それらの実行に別途必要な DLL のリストです。

PHP 拡張モジュール

拡張モジュール	説明	備考
php_bz2.dll	bzip2 圧縮関数	
php_calendar.dll	カレンダー 関数	PHP 4.0.3 以降ビルトイン
php_cpdf.dll	ClibPDF 関数	
php_crack.dll	Crack 関数	
php_ctype.dll	文字型(ctype) 関数	PHP 4.3.0 以降ビルトイン
php_curl.dll	CURL, Client URL Library 関数	*libeay32.dll* および *ssleay32.dll* が必要 （バンドル）
php_cybercash.dll	Cybercash 支払関数	PHP <= 4.2.0
php_db.dll	DBM 関数	非推奨の関数。DBA を代わりに使用のこと (*php_dba.dll*)
php_dba.dll	DBA: (dbm 型の) データベース・アブストラクション レイヤー関数	
php_dbase.dll	dBase 関数	
php_dbx.dll	dbx 関数	
php_domxml.dll	DOM XML 関数	PHP <= 4.2.0 では *libxml2.dll* が必要 （バンドル）, PHP >= 4.3.0 では *iconv.dll* が必要 （バンドル）
php_dotnet.dll	.NET 関数	PHP <= 4.1.1
php_exif.dll	EXIF 関数	php_mbstring.dll。*php.ini* で *php_exif.dll* は *php_mbstring.dll* の後で読み込まれる必要がある。
php_fbsql.dll	FrontBase 関数	PHP <= 4.2.0
php_fdf.dll	FDF: Forms Data Format 関数	*fdftk.dll* が必要 （バンドル）
php_filepro.dll	filePro 関数	読み込みのみ
php_ftp.dll	FTP 関数	PHP 4.0.3 以降ビルトイン
php_gd.dll	イメージ 関数 (GD ライブラリ)	PHP 4.3.2以降で削除。トゥルーカラー関数は GD1 では使用できない。 代わりに *php_gd2.dll* を使用のこと。
php_gd2.dll	イメージ 関数 (GD2 ライブラリ)	GD2
php_gettext.dll	Gettext 関数	PHP <= 4.2.0 では *gnu_gettext.dll* が必要 （バンドル）, PHP >= 4.2.3 では *libintl-1.dll* および *iconv.dll* が必要（バンドル）
php_hyperwave.dll	HyperWave 関数	
php_iconv.dll	ICONV 関数	*iconv-1.3.dll* が必要（バンドル）、 PHP >=4.2.1 *iconv.dll*
php_ifx.dll	Informix 関数	Informix ライブラリが必要
php_iisfunc.dll	IIS management 関数	
php_imap.dll	IMAP,POP3,NNTP 関数	
php_ingres.dll	Ingres II 関数	Ingres II ライブラリが必要
php_interbase.dll	InterBase 関数	*gds32.dll* が必要（バンドル）
php_java.dll	Java 関数	PHP <= 4.0.6 で *jvm.dll* が必要（バンドル）
php_ldap.dll	LDAP 関数	PHP <= 4.2.0 では *libsasl.dll* が必要（バンドル）, PHP >= 4.3.0 では *libeay32.dll* と *ssleay32.dll* が必要（バンドル）
php_mbstring.dll	マルチバイト文字列 関数	
php_mcrypt.dll	Mcrypt 暗号化 関数	*libmcrypt.dll* が必要
php_mhash.dll	Mhash 関数	PHP >= 4.3.0 で *libmhash.dll* が必要（バンドル）
php_mime_magic.dll	Mimetype 関数	*magic.mime* が必要（バンドル）
php_ming.dll	Ming 関数 (Flash 用)	
php_msql.dll	mSQL 関数	*msql.dll* が必要（バンドル）
php_mssql.dll	MSSQL 関数	*ntwdblib.dll* が必要（バンドル）
php_mysql.dll	MySQL 関数	PHP >= 5.0.0。*libmysql.dll* が必要（バンドル）
php_mysqli.dll	MySQLi 関数	PHP >= 5.0.0。*libmysql.dll* (PHP <= 5.0.2 では *libmysqli.dll*) が必要（バンドル）
php_oci8.dll	Oracle 8 関数	Oracle 8.1+ クライアントライブラリが必要
php_openssl.dll	OpenSSL 関数	*libeay32.dll* が必要（バンドル）
php_oracle.dll	Oracle 関数	Oracle 7 クライアントライブラリが必要
php_overload.dll	オブジェクトオーバーロード 関数	PHP 4.3.0 以降ビルトイン

拡張モジュール	説明	備考

PECL 拡張モジュールのインストール

目次

PECL インストール入門

» PECL は PHP 拡張モジュールのリポジトリで、 » PEAR パッケージシステムを経由して使用可能です。 ここでは PECL 拡張モジュールを取得してインストールする方法を解説します。

以下に示す手順では、PHP のソース配布物へのパスが */your/phpsrcdir/* であり、PECL 拡張モジュールの名前が *extname* であると仮定しています。適切に変更してください。また、 » pear コマンド についても理解していることとします。 PEAR マニュアルにある *pear* コマンドについての情報は、そのまま *pecl* コマンドにもあてはまります。

便利な機能を使用するには、拡張モジュールをビルドし、 インストールして読み込まなければなりません。 拡張モジュールのビルドとインストールについては以下でさまざまな方法を説明しますが、 モジュールの読み込みは自動的には行われません。モジュールを読み込むには、 *php.ini* ファイルに extension ディレクティブを追加するか、dl() 関数を使用します。

拡張モジュールのビルドにあたっては、適切なバージョンのツール (autoconf, automake, libtool など) を使用することが重要です。 必要なツールとそのバージョンについては、 » Anonymous CVS の手順 を参照してください。

PECL 拡張モジュールをダウンロードする

PECL 拡張モジュールをダウンロードするには、以下に示す通り、いくつかの方法があります。

- » http://pecl.php.net/ PECL のウェブサイトでは、PHP 開発チームが提供する さまざまな拡張モジュールについての情報が公開されています。 ChangeLog やリリース情報、必要な要件、リビジョンといった情報が参照可能です。
- *pecl download extname* PECL のウェブサイトで公開されている PECL 拡張モジュールは、 » pecl コマンド を使ってソースファイルをダウンロードすることもできます。 特定のバージョンを指定可能です。
- CVS 大半の PECL 拡張モジュールは CVS にも収められています。 » http://cvs.php.net/pecl/ で、 ウェブから参照することができます。 CVS から直接ダウンロードする場合は、 以下の一連のコマンドを使用します。 ユーザ *cvsread* のパスワードは *phpfi* であることに注意しましょう。

```
$ cvs -d:pserver:cvsread@cvs.php.net:/repository login
$ cvs -d:pserver:cvsread@cvs.php.net:/repository co pecl/extname
```

- Windows ダウンロード Windows 用にコンパイルされた PECL バイナリを » PHP Downloads ページの *Collection of PECL modules* のところからダウンロードできます。あるいは » PECL のスナップショット からも取得できます。また、拡張モジュールの DLL は、 » PECL4WIN で取得できます。 Windows での PHP のコンパイル方法については、 こちらの章 により適切な説明があります。

Windows で PECL を使用する

他の PHP 拡張モジュールの DLL と同様に、 PECL 拡張モジュールの DLL も *php.ini* の extension_dir フォルダにコピーし、*php.ini* の設定を行ってください。例を以下に示します。

```
extension=php_extname.dll
```

完了したら、ウェブサービスを再起動してください。

共有 PECL 拡張モジュールを、pecl コマンドを用いてコンパイルする

PECL を用いると、共有 PECL 拡張モジュールを容易に作成することができます。 以下のように » pecl コマンド を用います。

```
$ pecl install extname
```

extname のソースがダウンロードされ、 コンパイルおよび extension_dir への *extname.so* のインストールが行われます。 *extname.so* は、*php.ini* 経由で読み込まれます。

デフォルトでは、*pecl* コマンドは 状態が *alpha* あるいは *beta* のパッケージをインストールしません。*stable* なパッケージが存在しない場合は、以下のコマンドを使用して *beta* パッケージをインストールします。

```
$ pecl install extname-beta
```

特定のバージョンをインストールするには、次のような変化形を使用します。

```
$ pecl install extname-0.1
```

> **注意**: 拡張モジュールを *php.ini* で有効にしたら、 ウェブサービスを再起動させないとそれは反映されません。

---

phpize で共有 PECL 拡張モジュールをコンパイルする方法

時には *pecl* インストーラを使用するという選択肢が使えない場合もあります。 たとえばファイアウォールの内部で作業をしている場合がそうですし、まだリリースされていない CVS 版のように PECL パッケージ形式になっていないものをインストールする場合も それにあてはまります。このようなモジュールをビルドする必要がある場合は、 より低レベルなビルドツールを使用して手動でビルドします。

PHP 拡張モジュールのビルド環境を準備するために、 *phpize* コマンドを使用します。以下の例では、 拡張モジュールのソースが *extname* というディレクトリにあると仮定します。

```
$ cd extname
$ phpize
$ ./configure
$ make
# make install
```

上手くいけば、*extname.so* が作成され、 それが PHP の 拡張モジュールディレクトリ に置かれます。 この拡張モジュールを使用する前に、*php.ini* に *extension=extname.so* という行を追加する必要があります。

コンパイル済みのパッケージ (RPM など) を使用している場合などで、もし *phpize* コマンドが見つからない場合は、適切な *devel* 版の PHP パッケージをインストールしましょう。 PHP や拡張モジュールをビルドするために必要なヘッダファイルや *phpize* コマンドは、このパッケージに含まれます。

使用法についての詳細な情報を表示するには、 *phpize --help* を実行します。

PECL 拡張モジュールを PHP に静的に組み込む

PECL 拡張モジュールを PHP に静的に組み込みたいと思うこともあるでしょう。 そのためには、拡張モジュールのソースを *php-src/ext/* ディレクトリに置き、PHP にその設定スクリプトを生成させる必要があります。

```
$ cd /your/phpsrcdir/ext
$ pecl download extname
$ gzip -d < extname.tgz | tar -xvf -
$ mv extname-x.x.x extname
```

上記を行うと、以下のディレクトリが作成されます。

```
/your/phpsrcdir/ext/extname
```

これ以降、PHP に configure スクリプトを再実行させ、通常通りに PHP をビルドします。

```
$ cd /your/phpsrcdir
$ rm configure
$ ./buildconf --force
$ ./configure --help
$ ./configure --with-extname --enable-someotherext --with-foobar
$ make
$ make install
```

> **注意**: 'buildconf' スクリプトを実行するためには、autoconf 2.13 と automake 1.4+ が必要です (新しいバージョンの autoconf では動作するかも知れませんが、 サポートされていません)。

拡張モジュールによって、*--enable-extname* もしくは *--with-extname* オプションを指定します。 外部ライブラリを使用しない拡張モジュールにつ

いては、 *--enable* が使われるのが一般的です。 buildconf の後で、以下を行うと確認できます。

```
$ ./configure --help | grep extname
```

問題が起きた場合

目次

FAQ を読む

よくある問題とそうでない問題とがあります。 よくある問題（とその答え）の内、代表的なものが » PHP FAQ に記載されています。

FAQ に無い問題

FAQ を参照しても問題が解決しない場合には、PHP インストール用メーリングリストの誰かが 助けてくれるかもしれません。最初に、アーカイブをチェックして、過去に誰かが 自分と同じ問題に関する質問を行い、既に解答を受けていないかをチェックしてください。 アーカイブは、» http://www.php.net/ のサポートページで 公開されています。PHP インストール用メーリングリストに加入するには、空のメールを » php-install-subscribe@lists.php.net に送信してください。メーリングリストのアドレスは、*php-install@lists.php.net* です。

メーリングリストにおいて質問をする場合は、正確さを心がけ、 解答に必要な使用環境に関する事項（オペレーティングシステムの種類、 PHP のバージョン、Web サーバの種類、PHP を CGI として使用しているのか サーバモジュールとして使用しているのか、セーフモード 等）を明らかにするように してください。他の人が問題を再現し、テストできるのに充分なコードを 示すことが望まれます。

バグ報告

PHP のバグを発見した場合は、報告してください。あなたがそのバグを報告しない限り、 PHP の開発者がそのバグを知ることは恐らくなく、修正されることもないでしょう。 バグの報告は、» http://bugs.php.net/ にあるバグ追跡システムを使用して行います。 バグレポートをメーリングリストや個人宛てのメールで送らないでください。

» バグ報告の仕方 を読み、 それにしたがってバグ報告を投稿してください。

実行時設定

目次

設定ファイル

設定ファイル (PHP 3 では *php3.ini*、 PHP 4 以降では *php.ini*) は PHP の起動時に 読み込まれます。 PHP のサーバモジュール版では、Web サーバの起動時に 一度だけ読み込まれます。CGI 版と CLI 版では、スクリプトが呼び出される度に読み込まれます。

php.ini のデフォルトの場所は、コンパイル時のオプションにより決定されます (FAQ のエントリを参照)。 しかし、CGI 版および CLI 版の場合、コマンドラインスイッチ *-c* により、読み込む設定ファイルを変更することができます。 コマンドラインからの PHP の使用 に関する章を参照してください。環境変数 *PHPRC* を使用して、 *php.ini* を探すパスを追加することもできます。 *php.ini* を探す場所は、次の場所 (順番に) です。

- SAPI モジュール特有の場所 (Apache 2 における *PHPIniDir* ディレクティブ、 CGI/CLI 版における *-c* コマンドラインオプション、 NSAPI における *php_ini* パラメータ、 THTTPD における *PHP_INI_PATH* 環境変数)
- *PHPRC* 環境変数。PHP 5.2.0 より前では、 これは、次に挙げるレジストリキーの後にチェックされていました。
- PHP 5.2.0 では、レジストリの以下の箇所を順に探します。 *HKEY_LOCAL_MACHINE¥SOFTWARE¥PHP¥x.y.z¥IniFilePath*、 *HKEY_LOCAL_MACHINE¥SOFTWARE¥PHP¥x.y¥IniFilePath* および *HKEY_LOCAL_MACHINE¥SOFTWARE¥PHP¥x¥IniFilePath*。 ここで x、y および z はそれぞれ PHP のメジャー、マイナー、 リリース番号を表します。

- *HKEY_LOCAL_MACHINE¥SOFTWARE¥PHP¥IniFilePath* (Windows レジストリの場所)
- 現在の作業ディレクトリ (CLI を除く)
- Web サーバのディレクトリ (SAPI モジュールの場合)、 もしくは PHP ディレクトリ (そうでない Windows の場合)
- Windows ディレクトリ (*C:¥windows* もしくは *C:¥winnt*)) (Windows の場合) 、もしくはコンパイル時のオプション *--with-config-file-path*

php-SAPI.ini (ここで SAPI は使用する SAPI 名。 たとえば *php-cli.ini* や *php-apache.ini*) が存在する場合、 *php.ini* の代わりに使用されます。 SAPI 名は php_sapi_name() によって決定されます。

> **注意**: Apache web サーバは、スタート時にディレクトリをルート に変更するので、ファイルシステムのルートに *php.ini* が存在する場合、PHP はそれを読もうとします。

拡張モジュールに対する *php.ini* ディレクティブは、 各拡張モジュールのドキュメントで解説されています。 コア ディレクティブ （PHP 本体に対するディレクティブ）のリストは付録にまとめられてます。 ただし、（更新の都合上）すべての PHP ディレクティブが 本マニュアル中で解説されている訳ではありません。 使っているバージョンの PHP で指定可能なすべてのディレクティブについては、 *php.ini* ファイル内に詳細なコメントが記されていますので、参照してください。 もしくは、CVS から入手可能な » 最新の *php.ini* も有用でしょう。

Example#1 *php.ini* の例

```
; 引用符をつけないセミコロン(;)の後のテキストは、すべて無視されます
[php] ; セクションマーカ（角括弧の中のテキスト）は無視されます
; 論理値は、次のいずれかで指定します
;     true, on, yes
; または false, off, no, none
register_globals = off
magic_quotes_gpc = yes

; 文字列を二重引用符で括ることも可能です
include_path = ".:/usr/local/lib/php"

; バックスラッシュは他の文字と同様に処理されます
include_path = ".;c:¥php¥lib"
```

PHP 5.1.0 以降、ini ファイル内で既存の ini 変数を参照することが可能です。 例: *open_basedir = ${open_basedir} ":/new/dir"*.

設定を変更するには

Apache モジュールとして PHP を実行している場合

PHP を Apache モジュールとして使用している場合、Apache 設定ファイル (例、*httpd.conf*) もしくは *.htaccess* ファイルにディレクティブを記述することで、PHP の設定の変更を行うことが 可能です。このようにして設定変更を行うには、"AllowOverride Options" もしくは "AllowOverride All" 権限が必要です。

Apache 設定ファイルから PHP の設定を変更するには、 PHP 4 および PHP 5 に対しては、以下に示す Apache ディレクティブを使用します。 各設定オプションの変更の可否（**PHP_INI_ALL**, **PHP_INI_PERDIR**, または **PHP_INI_SYSTEM**）については、付録 php.ini ディレクティブのリスト を参照してください。

> **注意**: PHP 3 においては、*php3.ini* の各設定オプションに 対応した Apache 用ディレクティブが存在します。それらのディレクティブ名は "php3_" から始まります。

`php_value` *`name value`*

> 指定した設定オプションに値を設定します。変更の可否が、 **PHP_INI_ALL** もしくは **PHP_INI_PERDIR** である設定オプションに対し利用できます。 セット済みの値をクリアしたい場合は、*none* を 値として使用してください。
>
> > **注意**: 論理値を設定する場合には `php_value` を使用しないでください。代わりに、`php_flag` （下記参照）を使用する必要があります。

`php_flag` *`name on|off`*

> 設定オプションに論理値を設定するために使用します。変更の可否が、 **PHP_INI_ALL** もしくは **PHP_INI_PERDIR** である設定オプションで利用できます。

`php_admin_value` *`name value`*

> 指定した設定オプションに値を設定します。このディレクティブは、*.htaccess* ファイルでは利用できません。また、 `php_admin_value` で設定された設定オプションの値は、*.htaccess* では上書きできません。 セット済みの値をクリアしたい場合は、*none* を 値として使用してください。

`php_admin_flag` *`name on|off`*

> 設定オプションに論理値を設定するために使用します。 このディレクティブは、*.htaccess* ファイルでは利用できません。 php_admin_value で設定された設定オプションの値は、*.htaccess* では上書きできません。

Example#1 Apache 設定の例

```
<IfModule  mod_php5.c>
    php_value  include_path  ".:/usr/local/lib/php"
   php_admin_flag  safe_mode  on
</IfModule>
<IfModule  mod_php4.c>
    php_value  include_path  ".:/usr/local/lib/php"
   php_admin_flag  safe_mode  on
</IfModule>
<IfModule  mod_php3.c>
    php3_include_path  ".:/usr/local/lib/php"
   php3_safe_mode  on
</IfModule>
```

警告

PHP 定数は PHP 以外では使用できません。たとえば、 *httpd.conf* の中で error_reporting オプションを設定しようとして **E_ALL** や **E_NOTICE** のような PHP 定数を使用することは できません。これらは意味を有さないため、 *0* と評価されてしまいます。 代わりに、対応するビットマスク値を使用してください。 *php.ini* の中では、これらの PHP 定数を使用することができます。

Windows レジストリによる PHP の設定の変更

Windows 上で PHP を実行している場合、Windows レジストリを使用して設定値を ディレクトリ毎に変更することができます。 設定値は、レジストリキー *HKLM¥SOFTWARE¥PHP¥Per Directory Values* に保存され、そのサブキーがパス名となります。例えば、ディレクトリ *c:¥inetpub¥wwwroot* に対する設定値は、 キー *HKLM¥SOFTWARE¥PHP¥Per Directory Values¥c¥inetpub¥wwwroot* に保存されます。ディレクトリに対する設定は、そのディレクトリ、 およびそのサブディレクトリで実行されるすべてのスクリプトで有効となります。 PHP 設定オプションのディレクティブを名前とする文字列値をキーに登録してください。 また、値のデータに PHP 定数を記述しても解釈されませんので、注意してください。 しかし、**PHP_INI_USER** で変更可能な設定値はこの方法で設定することが可能ですが、 **PHP_INI_PERDIR** な値は設定できません。

他の方法

どのように PHP を実行しているかに係わらず、ini_set() 関数を 用いて、スクリプトの実行時に一部のオプションの設定値を変更することができます。詳細は、 ini_set() 関数のリファレンスを参照してください。

使用しているシステムにおける現在のオプション設定値の完全なリストを得たい場合は、 phpinfo() 関数を実行し、出力された結果を参照してください。 また、ini_get() 関数または get_cfg_var() 関数を用いて、個々のオプションの設定値にアクセスすることも可能です。

言語リファレンス

目次

基本的な構文

目次

HTML からの脱出

PHP はファイルを解析して開始タグと終了タグを探します。タグが 見つかると、PHP はコードの実行を開始したり終了したりします。 このような仕組みにより、PHP を他のあらゆる形式のドキュメント中に 埋め込むことができるのです。つまり、開始タグと終了タグで囲まれている 箇所以外のすべての部分は、PHP パーサに無視されます。 たいていの場合、PHP は HTML ドキュメントの中に埋め込まれます。 たとえば以下のようにです。

```
<p>この部分は無視されます。</p>
<?php echo '一方、この部分はパースされます。'; ?>
<p>この部分も無視されます。</p>
```

もっと複雑な構造を用いることもできます:

Example#1 高度なエスケープ処理

```
<?php
if ($expression) {
    ?>
     <strong>真です。</strong>
    <?php
} else {
    ?>
     <strong>偽です。</strong>
    <?php
}
?>
```

これは期待通りに動作します。なぜなら、PHP は ?> 終了タグを見つけると それ以降新たに開始タグを見つけるまでの内容を何でも出力するからです (終了タグの直後の改行は別です。 命令の分離 を参照ください)。 確かにこの例には少し無理があります。しかし、大量のテキストを出力する際に echo() や print() を用いることを 考えると、このように一度 PHP のパースモードを抜けるほうが効率的です。

PHP で用いられるタグは 4 種類あります。 これらのうちの 2 つ、<?php ?> と <script language="php"> </script> は常に使用することができます。 残りの 2 つは短縮型のタグと ASP スタイルの タグで、これらは *php.ini* ファイルによって有効か無効かを切り替えられます。 中には短縮型のタグや ASP スタイルのタグを 便利に感じる人がいるかも知れませんが、長いタグに比べると移植性に欠けます。 また一般的には推奨されていません。

> **注意**: さらに注意しなければならないことがあります。PHP コードを XML や XHTML に 埋め込む場合には、標準規格に従うために <?php ?> タグを使用する 必要があるでしょう。

Example#2 PHP の開始タグと終了タグ

```
1.  <?php echo 'XHTMLまたはXMLドキュメントを処理したい場合は、この方法が良いでしょう'; ?>

2.  <script  language="php">
        echo '(FrontPageのような) いくつかのエディタ  は処理命令を好み
         ません';
    </script>

3.  <? echo 'これは、SGML を処理する最もシンプルな方法です'; ?>
     <?= expression ?> This is a shortcut for "<? echo expression ?>"

4.  <% echo 'オプションでASP形式のタグを使用可能です'; %>
     <%= $variable; # これは、"<%echo .." のショートカットです。%>
```

例の 1. と 2. のタグは常に利用可能です。中でも 1. のタグは最も一般的で 推奨される方法です。

短縮型のタグ（例 3.）が有効なのは、*php.ini* 設定ファイルのディレクティブ short_open_tag が 有効になっている場合か PHP が --enable-short-tags オプションつきで configure されている場合のみです。

> **注意**: PHP 3 では、**short_tags()** 関数を用いて 短縮型のタグを有効にすることもできます。*これは PHP 3 でのみ有効な方法です!*

ASP 型のタグ（例 4.）が有効なのは、 *php.ini* 設定ファイルのディレクティブ asp_tags が有効になっている場合のみです。

> **注意**: ASP 型のタグのサポートは、3.0.4 で追加されました。

> **注意**: 再利用されるか、または、自分の制御下にないPHPサーバで運用される アプリケーションまたはライブラリを開発する場合、短縮型のタグの 使用は避けるべきです。これは、短縮型のタグがターゲットサーバー でサポートされていない可能性があるためです。 可搬性のある、再配布可能なコードでは、短縮型のタグを使用しない ようにしてください。

命令の分離

C や Perl と同様に、PHP でもステートメントを区切りにはセミコロンが必要と なります。PHP コードブロックの終了タグには自動的にセミコロンが含まれていると 認識されます。 従って PHP コードの最終行にはセミコロンを記述する必要はありません。 ブロックの終了タグは、直後に改行がある場合、それを含んだものになります。

```
<?php
    echo 'テストです';
?>

<?php echo 'テストです' ?>

<?php echo '終了タグを省略しました';
```

> **注意**: ファイル終端における PHP ブロックの終了タグはオプション（任意）です。 include() や require() を利用する際には、 終了タグを省略する方が無難です。というのは、そうすることでファイルの最後に 予期せぬ空白文字があらわれてしまうことを防げますし、後でレス

ポンスに ヘッダを付加することも可能となるからです。また、出力バッファリングを 使用しており、include したファイルの生成する部分の最後に余計な空白を つけたくない場合などにも便利です。

コメント

PHP は、'C', 'C++' および Unix シェル型（Perl 型）のコメントをサポートします。 例えば、

```
<?php
echo 'テストです'; // C++型の単一行用のコメント
/* 複数行用のコメント
   もう一行分のコメント */
echo 'もうひとつのテストです';
echo '最後のテストです'; # シェル型の単一行用のコメント
?>
```

"一行"コメントは、改行または PHP コードのブロックの終わり のうちどちらか最初にくる方までです。 つまり、// ... ?> あるいは # ... ?> の後に続く HTML コードは 表示されるということです。?> により PHP モードを終了して HTML モードに戻ると、そこでは // あるいは # は何の影響も 及ぼしません。asp_tags 設定ディレクティブが有効になっている場合、// %> および # %> でも同じような動作になります。しかし、一行コメントの中の </script> では PHP モードを終了することは ありません。

```
<h1>これは  <?php # echo 'シンプルな';?>  例です。</h1>
<p>上の見出しは 'これは  例です。' となります。
```

'C' 型のコメントは、最初に */ が現れた時点で終了します。 'C' 型のコメントがネストしないように注意する必要があります。 大きなブロックをコメントアウトしようとする際に、この間違いを犯しがちです。

```
<?php
 /*
    echo 'テストです'; /* このコメントが問題を生じます */
 */
?>
```

型

目次

導入

PHP は、8 種類の基本型をサポートします。

4 種類のスカラー型:

- 論理値 (boolean)
- 整数 (integer)
- float (浮動小数点数, 'double' も同じ)
- 文字列 (string)

2 種類の複合型:

- 配列 (array)
- オブジェクト (object)

そして、最後に 2 種類の特別な型:

- リソース (resource)
- ヌル (NULL)

本マニュアルでは、可読性を向上させるため、以下のような擬似的な型も使用します。

- mixed
- number
- callback

そして擬似変数 `$...` 。

いくつかの場所で "double" 型を使用していることに気付くかもしれません。 double は float と同じものだと考えてください。2 種類の名前が存在するのは、 歴史的な理由によるものです。

変数の型は、基本的にプログラマが設定するものではありません。 その変数が使用される文脈に応じ、PHP が実行時に決定します。

> **注意**: もし式の型と値を正確に知りたい場合は、 var_dump() 関数を使用してください。 デバッグのために、単純に人間が読みやすい形で型を表示したい場合には gettype() を使用してください。型をチェックする場合には gettype() を使用してはいけません 。*is_type* 関数を使用してください。いくつかの例を以下に示します。

```
<?php
$a_bool  = TRUE;   // a boolean
$a_str   = "foo";  // a string
$a_str2  = 'foo';  // a string
$an_int  = 12;     // an integer

echo gettype($a_bool); // prints out:   boolean
echo gettype($a_str);  // prints out:   string

// 数値であれば、4を足す
if (is_int($an_int)) {
    $an_int  += 4;
}

// $bool が文字列であれば, それをprintする
// (そうでなければ何も出力されない)
if (is_string($a_bool)) {
     echo "String:  $a_bool";
}
?>
```

ある変数の型を強制的に他の型に変換したい場合、変数を キャスト するか、 settype() 関数を使用します。

変数は、その型に依存して異なった動作をする場合があることに注意してください。 詳細な情報については、 型の変換 のセクションを参照ください。 またPHP 型の比較表 もご覧ください。さまざまな型の変数の比較に関する例があります。

論理型 (boolean)

論理型は、最も簡単な型です。boolean は、真偽の値を表します。 この値は、**TRUE** または **FALSE** のどちらかになります。

> **注意**: 論理型は、PHP 4 で導入されました。

構文

boolean リテラルを指定するには、キーワード **TRUE** または **FALSE** を指定してください。 両方とも大文字小文字に依存しません。

```
<?php
$foo = True; //  値TRUEを$fooに代入する
?>
```

通常、boolean 型の値を返す演算子を使用してから、 制御構造にその結果を渡します。

```
<?php
// == は、boolean型を返す演算子
if ($action  == "show_version") {
    echo "バージョンは1.23です。";
}

// これは冗長
if ($show_separators  == TRUE) {
    echo "<hr>¥n";
}

// 上の例は次のように簡単に書くことができます。
if ($show_separators) {
    echo "<hr>¥n";
}
?>
```

boolean への変換

boolean に明示的に変換を行うには、キャスト *(bool)* または *(boolean)* を使用します。しかし、演算子、関数、制御構造が boolean 型の引数を必要とする場合には、値は自動的に変換されるため、 多くの場合はキャストは不要です。

型の相互変換 も参照ください。

boolean に変換する場合、次の値は **FALSE** とみなされます。

- boolean の **FALSE**
- integer の 0 (ゼロ)
- float の 0.0 (ゼロ)
- 空の文字列、 および文字列の "0"
- 要素の数がゼロである 配列
- ゼロを要素とする オブジェクト (PHP 4のみ)
- 特別な値 NULL (値がセットされていない変数を含む)
- 空のタグから作成された SimpleXML オブジェクト

その他の値は全て **TRUE** とみなされます (全ての resourceを含みます)。

警告

-1 は、他のゼロでない数と同様に (正負によらず) **TRUE** とみなされます。

```
<?php
var_dump((bool) "");        //  bool(false)
var_dump((bool) 1);         //  bool(true)
var_dump((bool) -2);        //  bool(true)
var_dump((bool) "foo");     //  bool(true)
var_dump((bool) 2.3e5);     //  bool(true)
var_dump((bool) array(12)); //  bool(true)
var_dump((bool) array());   //  bool(false)
var_dump((bool) "false");   //  bool(true)
?>
```

整数

integer は、Z = {..., -2, -1, 0, 1, 2, ...}という集合 です。

任意精度整数 / GMP および float、 任意精度整数 / BCMath も参照ください。

構文

整数 (integer) は、10 進数(基数 10)、16 進数 (基数 16)、8 進数 (基数 8) 表記で指定可能です。オプションで、符号(-または+)を前に付けることが可能です。

8 進数表記を使用する場合、数の前に *0* (ゼロ) を付ける必要があります。また、16 進数表記を使用するには、数の前に *0x* を付ける必要があります。

Example#1 整数リテラル

```
<?php
$a = 1234; //  10進整数
$a = -123; //  負の数
$a = 0123; //  8進数 (10進数の83と等価)
$a = 0x1A; //  16進数 (10進数の26と等価)
?>
```

使用可能な整数リテラルの形式は以下のように定義されています。

```
decimal      : [1-9][0-9]*
             | 0

hexadecimal  :  0[xX][0-9a-fA-F]+

octal        : 0[0-7]+

integer      : [+-]?decimal
             | [+-]?hexadecimal
             | [+-]?octal
```

整数のサイズはプラットフォームに依存しますが、 約 20 億 (32 ビット符号付) が一般的な値です。 PHP は符号無し整数をサポートしていません。 整数のサイズは **PHP_INT_SIZE** で決まります。 最大値は、PHP 4.4.0 から PHP 5.0.5 までは **PHP_INT_MAX** でした。

警告

8 進数の整数値として不正な数字（例: 8 または 9）が渡された場合、 数値の残りの部分は無視されます。

Example#2 おかしな 8 進数

```
<?php
var_dump(01090); // 010 (8 進数) = 8 (10 進数)
?>
```

整数のオーバーフロー

integer型の範囲外の数を指定した場合、かわりに floatとして解釈されます。また、結果が integer型の範囲外の数となるような計算を行うと floatが代わりに返されます。

```
<?php
$large_number =  2147483647;
var_dump($large_number);
//  出力:  int(2147483647)

$large_number =  2147483648;
var_dump($large_number);
//  出力:  float(2147483648)

// 2^31 から 2^32-1 までの値については、指定した 16 進表現整数を出力できる
var_dump( 0xffffffff  );
//  output:  float(4294967295)

// 2^32-1 を超える値については、指定した 16 進表現整数を出力できない
var_dump( 0x100000000  );
//  出力:  int(2147483647)

$million = 1000000;
$large_number =  50000 * $million;
var_dump($large_number);
//  出力:  float(50000000000)
?>
```

警告

不幸にして、過去のスクリプトエンジンにはバグがあり、 負の数が含まれている場合に、常に正しく動作するわけではありませんでした。例えば、 *-50000 * $million* を実行した場合、結果は、 *-429496728* となりました。 しかし、オペランドが共に正の場合は問題ありませんでした。

この問題は PHP 4.1.0 で解決されました。

PHP には整数の割り算はありません。*1/2* は float 型の *0.5* になります。 下方向の整数値に値を丸めるためにキャストを使用することができ、 また、 round() 関数を使用することもできます。

```
<?php
var_dump(25/7);          //  float(3.5714285714286)
var_dump((int)  (25/7)); //  int(3)
var_dump(round(25/7));   //  float(4)
?>
```

整数への変換

integer に値を明示的に変換するには、キャスト *(int)* または *(integer)* のどちらかを使用してください。しかし、多くの場合、演算子、関数、制御構造が integer 引数を必要とする場合、値は自動的に変換されるため、 キャストを使用する必要はありません。 関数 intval() を用いて値を整数に変換することも可能です。

型の相互変換 を参照ください。

booleans から

FALSE は、*0* (ゼロ) となり、 **TRUE** は、*1* となります。

浮動小数点数から

float から整数に変換する場合、その数はゼロの方に丸められます。

float が整数の範囲 (通常は *+/- 2.15e+9 = 2^31*) を越える場合、結果は undefined となります。これは、 その float が正しい整数の結果を得るために十分な精度を得られなかったからです。 この場合、警告も通知も発生しません!

警告

未知の端数を integer にキャストしないでください。 この場合、予期しない結果となることがあります。

```
<?php
echo (int) ( (0.1+0.7) * 10 ); //  7が出力されます!
?>
```

より詳細な情報については、 float の精度に関する注意を参照ください。

文字列から

文字列変換 を参照ください。

他の型から

警告

整数への変換の動作は、他の型については定義されません。 現在の動作は、その値がまず 論理値に変換された 場合と同じです。しかし、この動作は予告なく変更されることがありえるので、 これを前提にしていはいけません。

浮動小数点数

("float", "double", "実数" のような) 浮動小数点数は、次の構文により指定できます。

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

規約:

```
LNUM            [0-9]+
DNUM              ([0-9]*[¥.]{LNUM}) | ({LNUM}[¥.][0-9]*)
EXPONENT_DNUM  (  ({LNUM} | {DNUM}) [eE][+-]?  {LNUM})
```

float の大きさはプラットフォーム依存です。ただし、通常はおよそ 10 進数で 14 桁の精度があり、最大値は ~1.8e308 (これは 64ビット IEEE フォーマットです) となります。

警告

浮動小数点数の精度

0.1 や *0.7* のようなシンプルな小数であっても、 それを内部的な二進数表現に変換する際には、どうしても多少精度が落ちてしまいます。 その結果、不思議な結果を引き起こすことがあります。たとえば、 *floor((0.1+0.7)*10)* の結果はたいてい *7* となるでしょう。おそらくは *8* を想定していらっしゃるでしょうが、そのようにはなりません。 これは、(この計算結果の) 内部的な値が *7.9999999999...* のようになっているからです。

こうなる理由のひとつとして、「有限小数に変換できない分数がある」 という事実があります。たとえば *1/3* を小数で表そうとすると *0.3333333...* となります。

よって、小数の最後の桁を信用してはいけませんし、 小数が等しいという比較を行ってはいけません。より高い精度が必要な場合には、 任意精度数学関数または gmp 関数を代わりに使用してください。

float への変換

文字列型がどのようにして浮動小数点数に変換されるかに関する詳細な情報は、 文字列の数値型への変換 のセクションをご覧ください。 そのほかの型の浮動小数点数への変換については、整数型への変換と同様です。 詳細は整数型への変換 のセクションをご覧ください。 PHP 5 以降、オブジェクトを不動小数点数に変換しようとした場合には、 通知がスローされます。

文字列

string は、文字が連結されたものです。PHP では、 文字は 1 バイトと同じです。つまり、256 個の異なる文字を使用可能です。 これは、PHP が Unicode をネイティブにサポートしていないことも意味します。 いくつかの Unicode サポートについてはutf8_encode() および utf8_decode() を参照してください。

> **注意**: 文字列が非常に大きくなっても問題ありません。 PHP に課せられる文字列のサイズの実用上の制限はありません。 このため、長い文字列に関して恐れる必要は全くありません。

構文

文字列リテラルは、3 つの異なる方法で指定することが可能です。

- 引用符
- 二重引用符
- ヒアドキュメント構文

引用符

文字列を指定する最も簡単な方法は、引用符 (文字 ') で括ることです。

引用符をリテラルとして指定するには、多くの他の言語と同様にバックスラッシュ (¥) でエスケープする必要があります。 バックスラッシュを引用符の前または文字列の最後に置きたい場合は、 二重にする必要があります。この他の文字をエスケープする場合には、 バックスラッシュも出力されることに注意してください! このため、 通常はバックスラッシュ自体をエスケープする必要はありません。

> **注意**: PHP 3 では、この場合、*E_NOTICE* レベルの警告が出力されます。

> **注意**: 他の二つの構文と異なり、 変数と特殊文字のエスケープシーケンスは、 引用符 (シングルクオート) で括られた文字列にある場合には *展開されません*。

```
<?php
echo 'this is a simple string';

echo 'You can also have embedded newlines in
strings this way as it is
okay to do';

// 出力: Arnold once said: "I'll be back"
```

```
echo 'Arnold once said: "I¥'ll be back"';

// 出力: You deleted C:\*.*?
echo 'You deleted C:¥¥*.*?';

// 出力: You deleted C:\*.*?
echo 'You deleted C:¥*.*?';

// 出力: This will not expand: \n a newline
echo 'This will not expand: ¥n a newline';

// 出力: Variables do not $expand $either
echo 'Variables do not $expand $either';
?>
```

二重引用符

文字列が二重引用符 (") で括られた場合、 PHP は、より多くの特殊文字のエスケープシーケンスを理解します。

エスケープされた文字

記述	意味
¥n	ラインフィード (LF またはアスキーの 0x0A (10))
¥r	キャリッジリターン (CR またはアスキーの 0x0D (13))
¥t	水平タブ (HT またはアスキーの 0x09 (9))
¥v	垂直タブ (VT またはアスキーの 0x0B (11)) (PHP 5.2.5 以降)
¥f	フォームフィード (FF またはアスキーの 0x0C (12)) (PHP 5.2.5 以降)
¥¥	バックスラッシュ
¥$	ドル記号
¥"	二重引用符
¥[0-7]{1,3}	正規表現にマッチする文字シーケンスは、8 進数表記の 1 文字です。
¥x[0-9A-Fa-f]{1,2}	正規表現にマッチする文字シーケンスは、16 進数表記の 1 文字です。

繰り返しますが、この他の文字をエスケープしようとした場合には、 バックスラッシュも出力されます! PHP 5.1.1 より前のバージョンでは、*¥{$var}* のバックスラッシュは出力されません。

しかし、二重引用符で括られた文字列で最も重要なのは、 変数名が展開されるところです。詳細は、文字列のパースを参照ください。

ヒアドキュメント

文字列を区切る別の方法としてヒアドキュメント構文 ("<<<") があります。この場合、ある ID (と、それに続けて改行文字) を <<< の後に指定し、文字列を置いた後で、同じ ID を括りを閉じるために置きます。

終端 ID は、その行の最初のカラムから始める必要があります。 使用するラベルは、PHP の他のラベルと同様の命名規則に従う必要があります。 つまり、英数字およびアンダースコアのみを含み、 数字でない文字またはアンダースコアで始まる必要があります。

警告

非常に重要なことですが、終端 ID がある行には、セミコロン (;) 以外の他の文字が含まれていてはならないことに注意しましょう。 これは、特に ID はインデントしてはならないということ、 セミコロンの前に空白やタブを付けてはいけないことを意味します。 終端 ID の前の最初の文字は、使用するオペレーティングシステムで定義された 改行である必要があることにも注意を要します。 これは、例えば、Macintoshでは *¥r* となります。 最後の区切り文字 (たいていはその後にセミコロンが続きます) の後にもまた、改行を入れる必要があります。

この規則が破られて終端 ID が "clean" でない場合、 終端 ID と認識されず、PHP はさらに終端 ID を探し続けます。 適当な終了 ID がみつからない場合、 スクリプトの最終行でパースエラーが発生します。

ヒアドキュメント構文を、クラスのメンバの初期化に用いることはできません。 他の文字列構文を利用してください。

Example#1 間違った例

```
<?php
class foo {
    public $bar = <<<EOT
bar
EOT;
}
?>
```

ヒアドキュメントは二重引用符を使用しませんが、 二重引用符で括られた文字列と全く同様に動作します。 しかし、この場合でも上記のリストでエスケープされたコードを使用することも可能です。 変数は展開されますが、文字列の場合と同様に ヒアドキュメントの内部で複雑な変数を表わす場合には注意が必要です。

Example#2 ヒアドキュメントで文字列を括る例

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* 変数を使用するより複雑な例 */
class foo
{
    var $foo;
    var $bar;

    function foo() {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$name = 'MyName';

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

注意: ヒアドキュメントは PHP 4 で追加されました。

変数のパース

スクリプトが二重引用符で括られるかヒアドキュメントで指定された場合、 その中の変数はパースされます。

構文の型には、単純な構文と 複雑な 構文の 2 種類があります。簡単な構文は、最も一般的で便利です。 この構文では、変数、配列値やオブジェクトのプロパティをパースすることが可能です。

複雑な構文は、PHP 4 で導入されました。 この構文は、式を波括弧で括ることにより認識されます。

簡単な構文

ドル記号 ($) を見付けると、 パーサは、有効な変数名を形成することが可能な最長のトークンを取得します。 変数名の終りを明示的に指定したい場合は、変数名を波括弧で括ってください。

```
$beer = 'Heineken';
echo "$beer's taste is great"; // 動作します。"'" は変数名として無効な文字です。
echo "He drunk some $beers"; // 動作しません。's' は、変数名として有効な文字です。
echo "He drunk some ${beer}s"; // 動作します。
echo "He drank some {$beer}s"; // 動作します。
```

同様に、配列添字とオブジェクトのプロパティをパースすることも可能です。 配列添字の場合、閉じ角括弧 (']') は添字の終りを意味し、 オブジェクトのプロパティの場合、同じ規則が簡単な変数として適用されます。 しかし、オブジェクトプロパティには、変数の場合のような手法はありません。

```
<?php
// これらの例は、文字列の内部で配列を使用する際のものです。
// 文字列の外部で使用する場合は、配列の文字列キーは常にクオート
// しましょう。また、{波括弧} も使用しないようにしましょう。

// すべてのエラーを表示するようにします。
error_reporting(E_ALL);

$fruits = array('strawberry' => 'red', 'banana' => 'yellow');

// シングルクオートの外では動作が異なることに注意してください。
echo "A banana is $fruits[banana].";

// 動作します。
echo "A banana is {$fruits['banana']}.";

// 動作しますが、以下に説明するように
// PHP はまず banana という名前の定数を探します。
echo "A banana is {$fruits[banana]}.";

// 動作しません。波括弧を使用しましょう。これはパースエラーとなります。
echo "A banana is $fruits['banana'].";

// 動作します。
echo "A banana is " . $fruits['banana'] . ".";

// 動作します。
echo "This square is $square->width meters broad.";

// 動作しません。解決策については、複雑な構文を参照ください。
echo "This square is $square->width00 centimeters broad.";
?>
```

より複雑な場合は、複雑な構文を使用する必要があります。

複雑な (波括弧) 構文

この構文が「複雑(complex)な構文」と呼ばれているのは、 構文が複雑であるからではなく、 この方法では複雑な式を含めることができるからです。

事実、この構文により、文字列の中に名前空間内のあらゆる値を含めることが可能です。 文字列の外側に置く場合と同様に式を書き、これを { と } の間に含めてください。'{' はエスケープすることができないため、 この構文は $ が { のすぐ後に続く場合にのみ認識されます (リテラル "{$" を指定するには、"{\$" を使用してください)。 以下のいくつかの例を見ると理解しやすくなるでしょう。

```
<?php
// すべてのエラーを表示します
error_reporting(E_ALL);

$great = 'fantastic';

// うまく動作しません。出力: This is { fantastic}
echo "This is { $great}";

// うまく動作します。出力: This is fantastic
echo "This is {$great}";
echo "This is ${great}";

// 動作します
echo "This square is {$square->width}00 centimeters broad.";

// 動作します
echo "This works: {$arr[4][3]}";

// これが動作しない理由は、文字列の外で $foo[bar]
// が動作しない理由と同じです。
// 言い換えると、これは動作するともいえます。しかし、
// PHP はまず最初に foo という名前の定数を探すため、
// E_NOTICE レベルのエラー(未定義の定数) となります。
echo "This is wrong: {$arr[foo][3]}";

// 動作します。多次元配列を使用する際は、
// 文字列の中では必ず配列を波括弧で囲むようにします。
echo "This works: {$arr['foo'][3]}";

// 動作します
echo "This works: " . $arr['foo'][3];

echo "You can even write {$obj->values[3]->name}";

echo "This is the value of the var named $name: {${$name}}";
?>
```

> **注意**: 関数とメソッドコールは PHP 5 から動作します。

> **注意**: 文字列内での変数のパースは、文字列の連結に比べてよりメモリを消費します。 メモリの使用量をできるだけ抑えた PHP スクリプトを書きたいのなら、 変数のパースを用いるのではなく、連結演算子 (.) を使用しましょう。

文字列への文字単位のアクセスと修正

$str[42] のように、 角括弧を使用してゼロから始まるオフセットを指定すると、 文字列内の任意の文字にアクセスし、修正することが可能です。 つまり、文字列を文字の配列として考えるわけです。 波括弧の後に任意の文字をゼロから始まるオフセットで指定することにより、 文字列内の文字にアクセス/修正することが可能です。

> **注意**: *$str{42}* のように波括弧を使用してアクセスすることも可能です。 しかし、角括弧を使用する方法のほうが推奨されます。 なぜなら、{波括弧} 形式は PHP 6 で廃止される予定だからです。

Example#3 文字列の例

```
<?php
// 文字列の最初の文字を取得します
$str = 'This is a test.';
$first = $str[0];

// 文字列の 3 番目の文字を取得します
$third = $str[2];

// 文字列の最後の文字を取得します
$str = 'This is still a test.';
$last = $str[strlen($str)-1];

// 文字列の最後の文字を変更します
$str = 'Look at the sea';
$str[strlen($str)-1] = 'e';

// {} を使用した、もうひとつの方法 (PHP 6 で廃止予定) です
$third = $str{2};
?>
```

> **注意**: その他の型の変数に対して *[]* や *{}* でアクセスすると、何もメッセージを出さずに単に **NULL** を返します。

便利な関数および演算子

文字列は、'.' (ドット) 結合演算子で結合することが可能です。'+' (付加) 演算子はこの例では出てこないことに注意してください。詳細については 文字列

演算子 を参照ください。

文字列の修正を行う場合には、便利な関数がたくさん用意されています。

一般的な関数については、文字列関数の節 を参照ください。高度な検索/置換を行う正規表現関数については Perl および POSIX 拡張 の 2 種類がありますが、 それぞれの節を参照ください。

URL 文字列用関数や文字列の暗号化/ 復号用の関数 (mcrypt および mhash) もあります。

最後に、探しているものがまだ見付からない場合には、 文字型の関数も参照ください。

文字列への変換

(string) キャストや strval() 関数を使って変数を文字列へ変換することができます。 文字列型を必要とする式のスコープにおいて、文字列への変換は自動的に行われます。 echo() や print() 関数を使うとき、 あるいは可変変数を文字列を比較するときにこの自動変換が行われます。 マニュアルの型 と 型の相互変換 の項を読むとわかりやすいでしょう。 settype()も参照してください。

boolean の **TRUE** は文字列の *"1"* に、 **FALSE** は *""* (空文字列) に変換されます。 これにより boolean と文字列の値を相互に変換することができます。

integer (整数) や浮動小数点数 (float) は その数値の数字として文字列に変換されます (指数の表記や浮動小数点数を含めて)。 浮動小数点数は、指数表記 (*4.1E+6*) を使用して変換されます。

> **注意**: 小数点を表す文字は、スクリプトのロケール (LC_NUMERIC カテゴリ) によって決まります。 setlocale() を参照ください。

配列は常に *"Array"* という文字列に変換されるので、 array の中を見るために echo() や print() を使ってダンプさせることはできません。 一つの要素を見るためには、*echo $arr['foo']* のようにしてください。内容の全てをダンプ/見るためには以降の TIP をご覧ください。

PHP 4 のオブジェクトは、常に *"Object"* という文字列に変換されます。 デバッグ等のために object の内部の変数を出力するような場合には、 以下をご覧ください。オブジェクトがなんという名前のクラスの インスタンスなのかを知るには get_class() をご覧ください。

リソースは常に *"Resource id #1"* という文字列に変換されます。*1* は実行中の PHP によって割り当てられる resource のユニークな番号です。 リソースの型を知るためには get_resource_type() を使用してください。

NULL は常に空文字列に変換されます。

以上に述べたように、配列、オブジェクト、リソースをプリントアウトしても その値に関する有益な情報を得られるわけではありません。 デバッグのために値を出力するのによりよい方法が知りたければ、 print_r() や var_dump() を参照ください。

PHP 変数を恒久的に保存するための文字列に変換することもできます。 この方法はシリアライゼーションと呼ばれ、 serialize() 関数によって実現できます。 WDDX サポートを有効にして PHP をセットアップすれば、PHP 変数を XML 構造にシリアライズすることもできます。

文字列の変換

数値として文字列が評価された時、結果の値と型は次のように定義されます。

文字列は、'.'、'e'、'E' のどれかが含まれている場合は float、それ以外は整数として評価されます。

文字列の最初の部分により値が決まります。文字列が、 有効な数値データから始まる場合、この値が使用されます。その他の場合、 値は 0 (ゼロ) となります。有効な数値データは符号(オプション)の後に、 1 つ以上の数字 (オプションとして小数点を 1 つ含む)、 オプションとして指数部が続きます。指数部は 'e' または 'E' の後に 1 つ以上の数字が続く形式です。

最初の式が文字列の場合、変数の型は 2 番目の式に依存します。

```
<?php
$foo = 1 + "10.5";                // $foo は float です (11.5)
$foo = 1 + "-1.3e3";              // $foo は float です (-1299)
$foo = 1 + "bob-1.3e3";           // $foo は integer です (1)
$foo = 1 + "bob3";                // $foo は integer です (1)
$foo = 1 + "10 Small Pigs";       // $foo は integer です (11)
$foo = 1 + "10 Little Piggies"; // $foo は integer です (11)
$foo = "10.0 pigs " + 1;          // $foo は integer です (11)
$foo = "10.0 pigs " + 1.0;        // $foo は float です (11)
?>
```

この変換に関する詳細は、Unix のマニュアルページで strtod(3) を参照ください。

本節の例を試したい場合、その例をカットアンドペーストしてから 動作を確認するために次の行を挿入してください。

```
<?php
echo "¥$foo==$foo; type is " . gettype ($foo) . "<br>¥n";
?>
```

(C 言語で行われるように) 数値に変換することで 一つの文字のコードを取得できると期待してはいけません。 文字と文字コードを相互に変換するには ord() および chr() 関数を使用してください。

配列

PHP の配列は、実際には順番付けられたマップです。マップは型の一種で、 *値*をキーに関連付けます。 この型は、いくつかの手法で最適化されます。このため、 実際の配列またはリスト (ベクトル)、(あるマップの実装である) ハッシュテーブル、ディレクトリ、コレクション、スタック、 キュー等として使用することが可能です。 PHP の配列には他の PHP 配列を値として保持することができるため、 非常に簡単にツリー構造を表現することが可能です。

これらのデータ構造に関する説明は本マニュアルの範囲外ですが、 これらの構造に各々に関する例を少なくとも一つ見付けることが可能です。 この分野は広範囲にまたがるので、 より詳細な情報については他の書籍を参照ください。

構文

array() で指定

配列 は、言語に組み込まれた array() で作成することが可能です。この構造は、 特定の数のカンマで区切られた *key* => *value* の組を引数とします。

```
array(   key =>   value
      , ...
      )
// key は、文字列または
// 非負の整数です。
// value に制約はありません.
```

```
<?php
$arr = array("foo" => "bar", 12 => true);

echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

key は、整数 または 文字列です。 あるキーが、整数の標準的な表現形式である場合、 そのように解釈されます (つまり、*"8"* は *8* として解釈されます。一方、 *"08"* は *"08"* として解釈されます)。 *key* に浮動小数点数値を指定すると、 その値は integer に切り詰められます。 PHP においては添字配列と連想配列の間に違いはなく、配列型は 1 つだけで、 整数または文字列のインデックスを使用することができます。

値には、PHP の全ての型を使用することができます。

```
<?php
$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));

echo $arr["somearray"][6];    // 5
echo $arr["somearray"][13];   // 9
echo $arr["somearray"]["a"];  // 42
?>
```

キーを省略した場合、整数添字の最大値が使用され、 新しいキーはその最大値 +1 となります。整数値は負の数とすることができ、 負の添字についても同様となります。例えば、最高時の添字が *-6* の場合、次のキーは *-5* となります。 整数添字がまだ存在しない場合、キーは *0* (ゼロ) となります。 値が既に代入されているキーを指定した場合、元の値は上書きされます。

```
<?php
// この配列は以下の配列と同じです ...
array(5 => 43, 32, 56, "b" => 12);

// この配列は上の配列と同じです
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

警告

PHP 4.3.0 以降、上記のような添字生成動作は変更されました。 現在では、配列に追加する際に、その配列の最大添字が負である場合は 次の添字はゼロ (*0*) となります。 以前は、正の添字の場合と同様に新しい添字は最大添字に +1 したものにセットされていました。

キーとして **TRUE** を使用した場合、 整数型の *1* がキーとして解釈されます。 キーとして **FALSE** を使用した場合、 整数型の *0* がキーとして解釈されます。 キーとして *NULL* を使用した場合、 空の文字列として評価されます。キーとして空の文字列を使用すると、 空の文字列のキーとその値を作成 (または上書き) します。 空の括弧を用いた場合と同じではありません。

配列またはオブジェクトをキーとして使用することはできません。 これを行なうと、warning: *Illegal offset type* を発生します。

角括弧構文で作成/修正

明示的に値を設定することにより、既存の配列を修正することも可能です。

これは、角括弧の中にキーを指定し、配列に値を代入することにより行います。 キーを省略することも可能です。この場合、空の角括弧 (*"[]"*) の変数名として追加してください。

```
$arr[key] = value;
$arr[] = value;
// key は 文字列 または
// 非負の整数のどちらかです。

// value は何でもかまいません
```

$arr がまだ存在しない場合、作成されます。 配列を指定する別の手段でもあります。ある値を変更するには、 新しい値に値を代入します。特定のキー/値の組を削除したい場合には、 unset() を使用する必要があります。

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // このスクリプトのこの位置に記述した場合、
                // $arr[13] = 56; と同じです

$arr["x"] = 42; // キー"x"の新しい要素を配列に追加します

unset($arr[5]); // 配列から要素を削除します

unset($arr);    // 配列全体を削除します
?>
```

注意: 上記のように、キーを省略して新規要素を追加する場合、 追加される数値添字は、使用されている添字の最大値 +1 になります。 既に値が割り当てられているキーを指定した場合には、 値は上書きされます。

警告

PHP 4.3.0 以降、上記のような添字生成動作は変更されました。 現在では、配列に追加する際に、 その配列の最大添字が負である場合は次の添え字はゼロ (*0*) となります。 以前は、正の添字の場合と同様に新しい添字は最大添字に +1 したものがセットされました。

次のキー生成において、オフセットとして使われる整数値 (添字の最大値) に対応するエントリーが、 必ずしも配列内に存在するわけではないことに注意してください。 しかし、その値は、多くの場合、 配列にある整数のキー値の最大値と等しいはずです。以下に例を示します。

```
<?php
// 簡単な配列を生成します。
$array = array(1, 2, 3, 4, 5);
print_r($array);

// 全てのアイテムを削除しますが、配列自体は削除しないでおきます。
foreach ($array as $i => $value) {
    unset($array[$i]);
}
print_r($array);

// アイテムを追加します(新しい添え字は0ではなく
// 5となることに注意)
$array[] = 6;
print_r($array);

// 添え字を振りなおします。
$array = array_values($array);
$array[] = 7;
print_r($array);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 5
)
Array
(
)
Array
(
    [5] => 6
)
Array
(
    [0] => 6
    [1] => 7
)
```

有用な関数

配列で使用する便利な関数がたくさんあります。 配列関数 の節を参照ください。

注意: unset()関数は配列のキーを削除することが出来ます。 ただし、これによってインデックスの再構築が行われるわけではないことに注意してください。 "通常の整数添字" (0 から始まり、1 つずつ増加) のみを使用している場合、 array_values() を用いてインデックスを再構築することができます。

```
<?php
$a = array(1 => 'one', 2 => 'two', 3 => 'three');
unset($a[2]);
/* これにより配列は以下の様に定義されます。
   $a = array( 1 => 'one', 3 => 'three');
```

```
   以下ではありません：
    $a = array( 1 => 'one', 2 => 'three');
*/

$b = array_values($a);
// bは、array(0 => 'one', 1 =>'three')となります
?>
```

配列専用の制御構造としてforeach があります。 この構造は、配列の要素に簡単に連続的にアクセスする手段を提供します。

配列ですべきこととしてはならないこと

なぜ、*$foo[bar]* は使用できないのか?

連想配列の添字の前後は常に引用符で括る必要があります。 例えば、$foo[bar] ではなく $foo['bar'] を使用してください。 しかし、$foo[bar] はなぜ誤りなのでしょうか? 古いスクリプトで次のような構文を見たことがあるかもしれません。

```
<?php
$foo[bar] = 'enemy';
echo $foo[bar];
// etc
?>
```

これは間違っていますが、動作します。では、なぜ間違っているのでしょう? その理由は、このコードには文字列 ('bar' - 引用符で括られている) ではなく未定義の定数 (bar) が使用されており、PHP が同じ名前の定数を不幸にして同じコードの中に定義する可能性があるためです。 下位互換性の維持のため、未定義の定数は同じ名前の文字列に自動的に変換されます。 そのため、このコードは動作します。 例えば、**bar** という名前の定義されていない定数があるとすると、 PHP は *'bar'* という文字列でそれを置換して使用します。

> **注意**: これは、添字を*常に*クォートするという意味ではありません。 定数や 変数 を添字として使う際には、クォートしてしまうと PHP はそれを解釈できなくなってしまいます。

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
ini_set('html_errors', false);
// 単純な配列
$array = array(1, 2);
$count = count($array);
for ($i = 0; $i < $count; $i++) {
    echo "¥nChecking $i: ¥n";
    echo "Bad: " . $array['$i'] . "¥n";
    echo "Good: " . $array[$i] . "¥n";
    echo "Bad: {$array['$i']}¥n";
    echo "Good: {$array[$i]}¥n";
}
?>
```

上の例の出力は以下となります。

```
Checking 0:
Notice: Undefined index: $i in /path/to/script.html on line 9
Bad:
Good: 1
Notice: Undefined index: $i in /path/to/script.html on line 11
Bad:
Good: 1

Checking 1:
Notice: Undefined index: $i in /path/to/script.html on line 9
Bad:
Good: 2
Notice: Undefined index: $i in /path/to/script.html on line 11
Bad:
Good: 2
```

この具体例を以下に示します。

```
<?php
// エラーを全て表示するよう設定
error_reporting(E_ALL);

$arr = array('fruit' => 'apple', 'veggie' => 'carrot');

// 正しい
print $arr['fruit'];  // apple
print $arr['veggie']; // carrot

// 間違い。これは動作しますが、未定義の定数fruitを使用しているため、
// 同時にE_NOTICEレベルのPHPエラーを発生します
//
// Notice: Use of undefined constant fruit - assumed 'fruit' in...
print $arr[fruit];    // apple

// 検証のため、定数を定義してみましょう。
// fruitという名前の定数に値'veggie'を代入します。
define('fruit','veggie');

// ここでは、出力が異なることに注意してください。
print $arr['fruit'];  // apple
print $arr[fruit];    // carrot
```

```
// 以下は文字列の中であるためOKです。定数は、文字列の中では解釈されな
// いため、E_NOTICEエラーはここでは発生しません。
print "Hello $arr[fruit]";      // Hello apple

// 例外が 1 つあり、文字列の中で波括弧で配列を括った場合には、
// 定数が解釈されます
print "Hello {$arr[fruit]}";    // Hello carrot
print "Hello {$arr['fruit']}";  // Hello apple

// これは動作せず、以下のようなパースエラーを発生します:
// Parse error: parse error, expecting `T_STRING' or `T_VARIABLE' or `T_NUM_STRING'
// 文字列の中でスーパーグローバルを使用した場合も無論同様です。
print "Hello $arr['fruit']";
print "Hello $_GET['foo']";

// 文字列結合で同じことをすることもできます。
print "Hello " . $arr['fruit']; // Hello apple
?>
```

error_reporting() で (**E_ALL** を指定する等により) *E_NOTICE* レベルのエラー出力を有効にした場合、 上記のエラーが出力されます。 デフォルトでは、 error_reporting はこれらを表示しない設定になっています。

構文の節に記述したように、角括弧 ('*[*' および '*]*') の間には、式がなければなりません。これは、 次のように書くことが可能であることを意味します。

```
<?php
echo $arr[somefunc($bar)];
?>
```

これは、関数の戻り値を配列の添字として使用する例です。PHP は定数についても認識します。以下のような *E_** の使用例を見たことがあるかもしれません。

```
<?php
$error_descriptions[E_ERROR]   = "A fatal error has occured";
$error_descriptions[E_WARNING] = "PHP issued a warning";
$error_descriptions[E_NOTICE]  = "This is just an informal notice";
?>
```

最初の例の *bar* と全く同様に *E_ERROR* も有効な添字であることに注意してください。 しかし、実際には最後の例は次のように書くことと同じです。

```
<?php
$error_descriptions[1] = "A fatal error has occured";
$error_descriptions[2] = "PHP issued a warning";
$error_descriptions[8] = "This is just an informal notice";
?>
```

これは、*E_ERROR* が *1* と等しいこと等によります。

では、なぜ *$foo[bar]* は動作することが可能なのでしょう? それは、*bar* が定数式であることを 期待される構文で使用されているためです。しかし、この場合、 *bar* という名前の定数は存在しません。PHP は、 この場合、あなたが文字列*"bar"* のようにリテラル *bar* を指定したが引用符を忘れたと仮定します。

では、なぜ間違っているのでしょう?

将来的に、PHP は他の定数またはキーワードを追加したいと思うかもしれず、 問題となる可能性があります。例えば、現在でも、 単語 *empty* および *default*を使用することはできません。 これは、これらが特別な 予約済みのキーワードであるためです。

> **注意**: 二重引用符で括られた文字列の中では 引用符で配列の添字を括らないことができ、このため、 *"$foo[bar]"* は有効です。 この理由の詳細については、上記の例や 文字列中での変数のパースを参照してください。

> **注意**: 二重引用符で括られた string の中で他の構文が有効です。 より詳細な情報については、文字列の中の変数 を参照ください。

配列への変換

integer, float, string, boolean, resourceのいずれの型においても、 array に変換する場合、 最初のスカラー値が割り当てられている一つの要素 (添字は 0) を持つ配列を得ることになります。

objectを配列にする場合には、配列の要素として オブジェクトの属性 (メンバ変数) を持つ配列を得ることになります。 添字はメンバ変数名となりますが、いくつか注意すべき例外があります。 private 変数の場合、変数名の頭にクラス名がつきます。また、 protected 変数の場合は、変数名の頭に '*' がつきます。 このとき、頭に追加される値の前後に null バイトがついてきます。 その結果、予期せぬ振る舞いをすることがあります。

```
<?php

class A {
    private $A; // これは '\0A\0A' となります
}

class B extends A {
    private $A; // これは '\0B\0A' となります
    public $AA; // これは 'AA' となります
}

var_dump((array) new B());
?>
```

上の例では 'AA' というキーがふたつあるように見えますが、 そのうちひとつは、実際は '\0A\0A' ということになります。

NULL を配列に変換すると、空の配列を得ます。

比較

array_diff() と 配列演算子 を用いると、配列を比較することができます。

例

PHP の配列型は、いろいろな使い方ができます。配列の強力な機能を示すため、 ここでいくつかの例を紹介します。

```
<?php
// this
$a = array( 'color' => 'red',
            'taste' => 'sweet',
            'shape' => 'round',
            'name'  => 'apple',
                       4        //  キーは0になります
          );

// これは以下と完全に同じです。
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name'] = 'apple';
$a[]        = 4;        //  キーは0になります

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// この結果は配列 array(0 => 'a' , 1 => 'b' , 2 => 'c'),
// または単に array('a', 'b', 'c') となります
?>
```

Example#1 array() の使用例

```
// マップを行う配列
$map = array( 'version'     => 4
            , 'OS'          => 'Linux'
            , 'lang'        => 'english'
            , 'short_tags' => true
            );

// 数値キーのみを有する
$array = array( 7
              , 8
              , 0
              , 156
              , -10
              );
// これは、array( 0 => 7, 1 => 8, ...) と同じです

$switching = array(           10 // key = 0
                   , 5    =>  6
                   , 3    =>  7
                   , 'a'  =>  4
                   ,          11 // key = 6 (最大の添字は5です)
                   , '8'  =>  2 // key = 8 (整数!)
                   , '02' => 77 // key = '02'
                   , 0    => 12 // 値10は12で上書きされます
                   );

// empty array
$empty = array();
```

Example#2 コレクション

```
<?php
$colors = array('red','blue','green','yellow');

foreach ( $colors as $color ) {
    echo "Do you like $color?¥n";
}

?>
```

上の例の出力は以下となります。

```
Do you like red?
Do you like blue?
Do you like green?
Do you like yellow?
```

PHP 5 以降では、配列を参照渡しすることでその値を直接変更できるようになりました。 それ以前のバージョンでは、以下のような回避策が必要です。

Example#3 コレクション

```
<?php
// PHP 5
foreach ($colors as &$color) {
    $color = strtoupper($color);
}
unset($color); /* これ以降の $color への書き込みが
```

```
配列の要素を書き換えてしまわないことを保証する */

// 旧バージョンでの回避策
foreach ( $colors as $key => $color ) {
    $colors[$key] = strtoupper($color);
}

print_r($colors);
?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => RED
     [1] => BLUE
     [2] => GREEN
     [3] => YELLOW
)
```

この例は、1 から始まる配列を作成します。

Example#4 1 から始まる添字

```
<?php
$firstquarter  = array(1 => 'January', 'February', 'March');
print_r($firstquarter);
?>
```

上の例の出力は以下となります。

```
Array
(
     [1] => 'January'
     [2] => 'February'
     [3] => 'March'
)
```

Example#5 配列に代入する

```
<?php
// ディレクトリから全てのアイテムを配列に代入する
$handle = opendir('.');
while ($file = readdir($handle))
{
    $files[] = $file;
}
closedir($handle);
?>
```

配列には順番が付けられます。異なったソート関数を用いて順番を変更することも可能です。 より詳細な情報については、配列関数 を参照ください。
count() 関数を使用することで、 配列の要素数を数えることが可能です。

Example#6 配列のソート

```
<?php
sort($files);
print_r($files);
?>
```

配列の値は何でも良いため、その値を他の配列とすることも可能です。 これにより、再帰的な配列や多次元の配列を作成することが可能です。

Example#7 再帰および多次元配列

```
<?php
$fruits = array ( "fruits"  => array ( "a" => "orange",
                                       "b" => "banana",
                                       "c" => "apple"
                                     ),
                  "numbers" => array ( 1,
                                       2,
                                       3,
                                       4,
                                       5,
                                       6
                                     ),
                  "holes"   => array (      "first",
                                       5 => "second",
                                            "third"
                                     )
                );

// 上の配列の内容を取得するための例
echo $fruits["holes"][5];    // "second"  を表示します
echo $fruits["fruits"]["a"]; // "orange"  を表示します
unset($fruits["holes"][0]);  // "first"   を削除します

// 新しい多次元配列を作成します
$juices["apple"]["green"] = "good";
```

```
?>
```

配列の割り当てにおいては、常に値がコピーされることに注意してください。 これは、current() および類似の関数が使用する 内部配列ポインタの値がリセットされるということを意味します。 配列をリファレンスでコピーする場合には、 リファレンス演算子を使う必要があります。

```
<?php
$arr1 = array(2, 3);
$arr2 = $arr1;
$arr2[] = 4; // $arr2 が変更されます。
             // $arr1 は array(2,3) のままです。

$arr3 = &$arr1;
$arr3[] = 4; // $arr1 と $arr3 は同じ内容になります。
?>
```

オブジェクト

オブジェクトの初期化

オブジェクトを初期化するためには、*new* 命令によりオブジェクトのインスタンスを変数に作成します。

```
<?php
class foo
{
    function do_foo()
    {
        echo "foo を実行します。";
    }
}

$bar = new foo;
$bar -> do_foo ();
?>
```

詳細な事項については、 クラスおよびオブジェクト に関するセクションを参照ください。

オブジェクトへの変換

オブジェクトがオブジェクトに変換される場合、それは修正されません。 他の型の値がオブジェクトに変換される場合、ビルトインクラスである *stdClass* の新しいインスタンスが生成されます。 値が null の場合、新しいインスタンスは空となります。 配列がオブジェクトに変換される場合、配列のキーがプロパティ名となり、 配列の値がプロパティの値となります。他の値の場合、 *scalar* という名前のメンバ変数が値を格納します。

```
<?php
$obj = (object) 'ciao';
echo $obj->scalar;  // 'ciao' を出力します
?>
```

リソース

リソースは特別な変数であり、外部リソースへのリファレンスを保持しています。 います。リソースは、特別な関数により作成され、使用されます。これらの関数および対応する全てのリソース型の一覧については、 付録 を参照ください。

> **注意**: リソース型は、PHP 4 で導入されました。

get_resource_type() も参照ください。

リソースへの変換

リソース型は、オープンされたファイル、データベース接続、 イメージキャンバスエリアのような特殊なハンドルを保持するため、 他の値をリソースに変換することはできません。

リソースの開放

PHP 4 の Zend エンジンに導入されたリファレンスカウンティングシステムのおかげで、 あるリソースがもう参照されなくなった場合に (Java と全く同様に)、 そのリソースは自動的に削除されます。この場合、このリソースが作成した 全てのリソースは、ガベージコレクタにより開放されます。 このため、free_result 関数を用いて手動でメモリを開放する必要が生じるのはまれです。

> **注意**: 持続的データベース接続は特別で、ガベージコレクタにより破棄されません。 持続的接続 も参照ください。

NULL

特別な **NULL** 値は、ある変数が値を持たないことを表します。**NULL** は、NULL 型の唯一の値です。

> **注意**: NULL 型は、PHP 4 で導入されました。

変数は、以下の場合に **NULL** とみなされます。

- 定数 **NULL** が代入されている場合。
- まだ値が何も代入されていない場合。
- unset() されている場合。

構文

NULL 型の値は一つだけで、 大文字小文字を区別しないキーワード **NULL**です。

```
<?php
$var = NULL;
?>
```

is_null() および unset() も参照ください。

本ドキュメントにおける疑似的な型および変数

mixed

mixed は、引数に多様な型 (全てである必要はない) を使うことができることを示します。

例えば gettype() 関数は全ての PHP の型を受け入れるのに対し、 str_replace() は文字列と配列のみを受け入れます。

number

number は引数が integer または float のどちらでもよいことを示します。

callback

call_user_func() や usort() 等の関数は、ユーザが定義するコールバック関数を引数として受け入れます。 コールバック関数は、単純な関数だけでなく、オブジェクトのメソッド あるいはクラスの静的メソッドであってもかまいません。

PHP 関数はその名前を単に文字列として渡されます。 どのようなビルトインまたはユーザ定義の関数も渡すことができます。 ただし下記を除きます。 array(), echo(), empty(), eval(), exit(), isset(), list(), print() unset().

オブジェクトのインスタンスを作成するための方法の 1 つは、 オブジェクトを 0 番目の要素、 メソッド名を 1 番目の要素として含む配列を渡す方法です。

静的なクラスメソッドの場合、 0 番目の要素としてオブジェクトを渡す代わりにクラス名を渡すことにより、 オブジェクトのインスタンスを作成せずに渡すことができます。

一般的なユーザ定義関数とは異なり、create_function() では無名コールバック関数を作成することができます。

Example#1 コールバック関数の例

```
<?php
// コールバック関数の例
function my_callback_function() {
    echo 'hello world!';
}

// コールバックメソッドの例
class MyClass {
    function myCallbackMethod() {
        echo 'Hello World!';
    }
}

// タイプ 1: 単純なコールバック
call_user_func('my_callback_function');

// タイプ 2: スタティッククラスメソッドのコール
call_user_func(array('MyClass', 'myCallbackMethod'));

// タイプ 3: オブジェクトメソッドのコール
$obj = new MyClass();
call_user_func(array($obj, 'myCallbackMethod'));
?>
```

> **注意**: PHP4 では、実際のオブジェクトを指すコールバックを作成するには 参照を使用する必要があります。そのコピーを使用してはいけません。 詳細は 参照についての説明 を参照ください。

void

返り値の型が *void* である場合は、 返り値に意味がないことを表します。パラメーター一覧で *void* が使用されている場合は、 その関数がパラメータを受

け付けないことを表します。

...

関数のプロトタイプ宣言における *$...* は、 …など を表します。 この変数名を用いるのは、たとえば任意の数の引数を取りうる関数などです。

型の相互変換

PHP は、変数定義時に明示的な型定義を必要と(または、サポート) しません。ある変数の型は、その変数が使用される文により定義されます。 これは、ある文字列を変数 *var* に代入した場合には、 *var* は文字列になることを意味しています。 ある整数値を *var* に代入した場合には、 その変数は整数になります。

PHP の自動型変換の例の一つは、加算演算子 '+' です。 オペランドのどれかが float の場合、全てのオペランドは float として評価され、結果は float になります。 その他の場合、オペランドは整数として解釈され、結果も整数になります。 この自動型変換は、 オペランド自体の型を変更するものではないということに注意してください。 変わるのは、オペランドがどのように評価されるかだけです。

```
<?php
$foo = "0";  // $foo は文字列です (ASCII 48)
$foo += 2;   // ここでは、$foo は整数です (2)
$foo = $foo + 1.3;  // ここでは、$foo はfloatです (3.3)
$foo = 5 + "10 Little Piggies"; // $foo は整数です (15)
$foo = 5 + "10 Small Pigs";     // $foo は整数です (15)
?>
```

最後の二つの例が奇妙に思える場合には、 文字列変換 を参照ください。

ある変数を強制的にある特定の型として評価させたい場合には、 型キャスト のセクションを参照ください。ある変数の型を変更したい場合には、settype() を参照してください。

本節の例をテストしたい場合には、 var_dump() を使用することが可能です。

> **注意**: 配列への自動変換の動作は現時点で定義されていません。
> また、PHP では配列の添字と同じ構文を使用した文字列へのアクセスをサポートしているので、 次の例はあらゆるバージョンの PHP で成立します。
>
> ```
> <?php
> $a    = 'car'; // $a は文字列です
> $a[0] = 'b';   // $a はここでも文字列です
> echo $a;       // bar
> ?>
> ```
>
> 詳細は、 文字として文字列をアクセスするというセクションを参照してください。

型キャスト

PHP の型キャストは、C 言語と同様に動作します。つまり、 変換しようとする型を括弧で括り、キャストする変数の前に置きます。

```
<?php
$foo = 10;   // $foo は整数です
$bar = (boolean) $foo;   // $bar はbooleanです
?>
```

使用可能なキャストを以下に示します。

- (int), (integer) - 整数へのキャスト
- (bool), (boolean) - 論理値へのキャスト
- (float), (double), (real) - float へのキャスト
- (string) - 文字列へのキャスト
- (binary) - バイナリ文字列へのキャスト (PHP 6)
- (array) - 配列へのキャスト
- (object) - オブジェクトへのキャスト

(binary) によるキャストや b プレフィックスのサポートは、PHP 5.2.1 で追加されました。

括弧の中でタブとスペースを使用することができることに注意してください。 したがって、次の文は機能的に等価です。

```
<?php
$foo = (int) $bar;
$foo = ( int ) $bar;
?>
```

リテラル文字列や変数を、バイナリ文字列にキャストします。

```
<?php
$binary  =  (binary)$string;
$binary  = b"binary  string";
?>
```

> **注意**: ある変数を文字列にキャストする代わりに、 二重引用符で括ることもできます。

```
<?php
$foo = 10;            // $foo は整数です
$str = "$foo";        // $str は文字列です
$fst = (string) $foo; // $fst も文字列です

// 以下は、"they are the same"を出力します
if ($fst === $str) {
    echo "they are the same";
}
?>
```

型の間でキャストを行う際の動作は、必ずしも明確ではありません。 詳細については、以下の節を参照ください。

- 論理値への変換
- 整数への変換
- 浮動小数点数への変換
- 文字列への変換
- 配列への変換
- オブジェクトへの変換
- リソース型への変換
- 型の比較表

変数

目次

基本的な事

PHP の変数はドル記号の後に変数名が続く形式で表されます。 変数名は大文字小文字を区別します。

変数名は、PHPの他のラベルと同じルールに従います。 有効な変数名は文字またはアンダースコアから始まり、任意の数の文字、 数字、アンダースコアが続きます。正規表現によれば、これは次の ように表現することができます。 '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

> **注意**: ここで言うところの文字とはa-z、A-Z、127から255まで (0x7f-0xff)のアスキー文字を意味します。

> **注意**: *$this* は特別な変数であり、ここに代入することはできません。

ヒント

ユーザレベルでの命名の手引き もご覧になるとよいでしょう。

変数関連の関数に関する情報については、 変数関数リファレンス を参照ください。

```
$var = 'Bob';
$Var = 'Joe';
echo "$var, $Var";      // "Bob, Joe"を出力します。

$4site = 'not yet';     // 無効：数字で始まっている。
$_4site = 'not yet';    // 有効：アンダースコアで始まっている。
$täyte = 'mansikka';    // 有効：'ä' はアスキーコード228です。
```

PHP 3では、変数は常にその値により代入されていました。 これは、つまり、ある変数にある式を代入する際、元の式の 値全体がコピーされる側の変数にコピーされるということです。 これは、例えば、ある変数の値を他の変数に代入した後で、 これらの変数の1つを変更しても他の変数には影響を与えないという ことを意味します。この種の代入に関するより詳細な情報については、 式 を参照ください。

PHP 4以降では、変数に値の代入を行う別の方法を提供します。それは、 *参照による代入* です。 この場合、新規の変数は元の変数を参照するだけです。 (言いかえると、元の変数の"エイリアスを作る"または元の変数を"指す") 新規の変数への代入は、元の変数に影響し、その逆も同様となります。

参照により代入を行うには、代入する変数(ソース変数)の先頭に アンパサンドを加えます。たとえば、次の簡単なコードは 'My name is Bob'を二度出力します。

```
<?php
$foo = 'Bob';              //  値'Bob'を$fooに代入する。
$bar = &$foo;              //  $fooを$barにより参照
$bar = "My name is $bar";  //  $barを変更...
echo $bar;
```

```
echo $foo;                  //  $fooも変更される。
?>
```

注意すべき重要な点として、名前のある変数のみが参照により代入できる ということがあります。

```
<?php
$foo = 25;
$bar = &$foo;      //  これは有効な代入です。
$bar = &(24 * 7);  //  無効です。名前のない式を参照しています。

function test() {
    return 25;
}

$bar = &test();    //  無効。
?>
```

PHP では変数を初期化する必要はありませんが、そのようにするのはとても よいことです。初期化されていない変数の値は、その型のデフォルト値 - **FALSE**、ゼロ、空の文字列あるいは空の配列となります。

Example#1 初期化されていない変数のデフォルト値

```
<?php
echo ($unset_bool ? "true" : "false"); // false
$unset_int += 25; // 0 + 25 => 25
echo $unset_string . "abc"; // "" . "abc" => "abc"
$unset_array[3] = "def"; // array() + array(3 => "def") => array(3 => "def")
?>
```

初期化されていない変数のデフォルト値に依存すると、そのファイルを include している別のファイルで同名の変数が使用されていた場合などに 問題を起こします。また、register_globals が on の場合には重大なセキュリティリスク を抱えることになります。初期化されていない変数を使用すると、E_NOTICE レベルのエラーが発生します。 しかし、初期化されていない配列に要素を追加する場合はエラーにはなりません。 変数が初期化されているかどうかの判断には、isset() を使用します。

定義済みの変数

PHPは、実行する全てのスクリプトに定義済みの多くの変数を 提供します。しかし、これらの変数の多くは、 実行するサーバーの種類、サーバーのバージョンおよび設定、 その他の要素に依存しており、完全に記述することはできません。 これらの変数のいくつかは、PHPを コマンドライン で実行した場合には利用できません。 これらの変数の一覧については、 予約済みの定義済みの変数 のセクションを参照してください。

警告

PHP 4.2.0以降では、PHPディレクティブ register_globalsの デフォルト値が*off*に変更されています。 これは、PHPにおける大きな変更です。register_globalsを*off*にすると、 グローバルスコープに定義済みの変数に影響を与えます。 例えば、*DOCUMENT_ROOT*を取得するには、*$DOCUMENT_ROOT*のかわりに *$_SERVER['DOCUMENT_ROOT']*を使用することに なります。また、URL *http://www.example.com/test.php?id=3* から *$id*の代わりに*$_GET['id']*、 *$HOME*のかわりに*$_ENV['HOME']*を 使用します。

この変更に関する情報については、 register_globalsに関する設定 エントリ、セキュリティに関する章の register_globalsの使用、 また、» 4.1.0および » 4.2.0の Release Announcementsを参照してください。

スーパーグローバル 配列のようなPHPの予約済みの定義済み変数を使用することが 推奨されます。

バージョン4.1.0以降、PHPに(使用する場合)Webサーバ、環境変数、 ユーザ入力からの変数を値とする定義済みの配列が追加されています。 これらの新しいは配列は、自動グローバル、すなわち、自動的に全ての スコープで利用可能です。このため、これらは "スーパーグローバル" といわれることもあります。 (PHPには、ユーザ定義のスーパーグローバルという機構はありません。) スーパーグローバルのリストを以下に示します。 しかし、これらの内容のリストおよび定義済みのPHP変数とそれらの特性に 関する更なる議論については、 定義済みの予約変数の セクションを参照してください。 より古い定義済みの変数(*$HTTP_*_VARS*)もまだ 存在します。 PHP 5.0.0 以降、PHP の長い 定義済みの変数 配列は register_long_arrays ディレクティブにより無効にすることができます。

> **注意**: **可変変数** 可変変数として スーパーグローバルを使うことはできません。
>
> **注意**: スーパーグローバルと HTTP_*_VARS は同時に存在させることができますが、 お互い同一ではありません。つまり、一方を変更してももう一方はそのままです。

variables_orderに ある変数が設定されていない場合、これらに対応するPHPの定義済み 変数も空のままとなります。

PHP スーパーグローバル

$GLOBALS
: スクリプトのグローバルスコープの中で現在利用可能な全ての 変数へのリファレンスを含みます。この配列のキーは、グローバル 変数の名前です。 *$GLOBALS* は、PHP 3以降存在します。

$_SERVER
: Webサーバーまたはカレントのスクリプトの実行環境に直接関係する ものにより設定された変数。以前の(まだ利用可能ですが、推奨 されません) *$HTTP_SERVER_VARS*配列に 類似のものです。

$_GET
: URL クエリ文字列によりスクリプトに入力された変数。以前の (まだ利用可能ですが、推奨されません) *$HTTP_GET_VARS*配列に類似のもので

す。

$_POST
: HTTP POSTによりスクリプトに入力された変数。以前の (まだ利用可能ですが、推奨されません) *$HTTP_POST_VARS*配列に類似のものです。

$_COOKIE
: HTTP Cookieによりスクリプトに入力された変数。以前の (まだ利用可能ですが、推奨されません) *$HTTP_COOKIE_VARS*配列に類似のものです。

$_FILES
: HTTP POSTファイルアップロードによりスクリプトに渡される変数。 以前の*$HTTP_POST_FILES*配列(まだ利用可能ですが、 推奨されません)と類似しています。詳細については、 POSTメソッド のアップロードを参照してください。

$_ENV
: 環境によりスクリプトに指定される変数。 以前の*$HTTP_ENV_VARS*配列(まだ利用可能ですが、 推奨されません)と類似しています。

$_REQUEST
: ユーザ入力機構によりスクリプトに入力される全ての変数で、 このため、信頼することができません。この配列に含まれる 変数の存在と順番は、設定ディレクティブ variables_order に基づき定義されます。この配列は、4.1.0より前のバージョンのPHP には直接類似するものはありません。 import_request_variables()も参照してください。

警告

PHP 4.3.0以降、*$_FILES*によるファイル情報は *$_REQUEST*には存在しません。

注意: コマンドライン で実行している場合、この変数には、 *argv* および *argc* エントリ は含まれません。これらは、*$_SERVER* 配列には 存在します。

$_SESSION
: スクリプトのセッションに現在登録されている変数。 以前の*$HTTP_SESSION_VARS*配列(まだ利用可能ですが、 推奨されません)と類似しています。詳細については、 セッション処理関数 を参照ください。

変数のスコープ

変数のスコープは、その変数が定義されたコンテキストです。ほとんどの PHP 変数は、スコープを1つだけ有しています。このスコープの範囲は、includeやrequireにより読みこまれたファイルも含みます。例えば、

```
<?php
$a = 1;
include 'b.inc';
?>
```

この例で、変数*$a*はインクルードされた *b.inc* スクリプトの中でも利用可能です。しかし、 ユーザー定義の関数の中では変数の有効範囲はローカル関数の中となりま す。関数の中で使用された変数はデフォルトで有効範囲が関数内部に制限 されます。例えば、

```
<?php
$a = 1; /* グローバルスコープ */

function Test() {
    echo $a; /* ローカルスコープ変数の参照 */
}

Test();
?>
```

このスクリプトは、出力を全く行いません。これは、echo 命令がローカ ル版の *$a* 変数を参照しているにもかかわらず、こ のスコープでは値が代入されていないからです。この動作は、特にローカ ルな定義で上書きしない限りグローバル変数が自動的に関数で使用可能で ある C 言語と少々異なっていると気がつかれるかもしれません。C言語の ような場合、グローバル変数を不注意で変更してしまうという問題を生じ る可能性があります。PHP では、グローバル変数は、関数の内部で使用す る場合、関数の内部でグローバルとして宣言する必要があります。

globalキーワード

まず、*global*の使用例を示します。

Example#1 globalの使用

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
?>
```

上のスクリプトは、"3" を出力します。関数の内部で *$a*、*$b* をグローバル宣言を行うことにより、両変数への参照は、グローバル変数 の方を参照することになります。ある関数により操作できるグローバル変 数の数は無制限です。

グローバルスコープから変数をアクセスする2番目の方法は、PHPが定義す る配列*$GLOBALS*を使用することです。先の例は、次 のように書き換えることができます。

Example#2 globalのかわりに*$GLOBALS*を使用する

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}

Sum();
echo $b;
?>
```

配列*$GLOBALS*は連想配列であり、グローバル変数の 名前がキー、その変数の内容が配列要素の値となっています。 *$GLOBALS* は スーパーグローバル であるため、*$GLOBALS*は全てのスコープに存在します。 以下にスーパーグローバルの効果を示す例を示します。

Example#3 スーパーグローバルとスコープの例

```
<?php
function test_global()
{
    // ほとんどの定義済み変数は"スーパー"ではなく、関数内の
    // ローカルスコープで有効とするには'global'をコールする必要があります。
    global $HTTP_POST_VARS;

    echo $HTTP_POST_VARS['name'];

    // スーパーグローバルはどのスコープでも有効であり
    // 'global'をコールする必要がありません。
    // スーパーグローバルはPHP4.1.0以降で利用できます。
    // HTTP_POST_VARS は今や非推奨とされています。
    echo $_POST['name'];
}
?>
```

静的変数の使用

変数のスコープに関する別の重要な機能は、*静的 (static)* 変数です。静的変数はローカル関数スコープのみに 存在しますが、プログラム実行がこのスコープの外で行われるようになっ てもその値を失わないません。次の例を見てください。

Example#4 静的変数が必要な場面の例

```
<?php
function Test()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

この関数は、コールされる度に*$a*を *0*にセットし、"0" を出力するのでほとん ど役にたちません。変数を1増やす $a++ は、関数から外に出ると変数*$a*が消えてしまうために目的を達成しません。現在 のカウントの追跡ができるようにカウント関数を使用できるようにするた めには、変数*$a*をstaticとして宣言します。

Example#5 静的変数の使用例

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

こうすると、Test() 関数がコールされる度に*$a*の値 を出力し、その値を増加させます。

static変数は、再帰関数を実現する1つの手段としても使用されます。再帰 関数は、自分自身をコールする関数です。再帰関数を書くときには、無限 に再帰を行う可能性があるため、注意する必要があります。適当な方法に より再帰を確実に終了させる必要があります。次の簡単な関数は、中止す るタイミングを知るためにstatic変数*$count*を用いて、 10 回まで再帰を行います。

Example#6 再帰関数での静的変数の使用

```
<?php
function Test()
{
    static $count = 0;

    $count++;
```

```
    echo $count;
    if ($count < 10) {
        Test();
    }
    $count--;
}
?>
```

注意: 静的変数は、上の例に見られるような方法で宣言されます。 式の結果を静的変数に代入しようとすると、パースエラーが 発生します。

Example#7 静的変数の宣言

```
<?php
function foo(){
    static $int = 0;          // 正しい
    static $int = 1+2;        // 間違い（式を代入しています）
    static $int = sqrt(121);  // 間違い（同じく式を代入しています）

    $int++;
    echo $int;
}
?>
```

グローバル変数と静的変数のリファレンス

*PHP4*を駆動するZend Engine 1では、 リファレンス変数の修正子 *static* および *global* を実装しています。 例えば、関数スコープ内に*global* 命令により実際にインポートされた真のグローバル変数は、 実際にグローバル変数へのリファレンスを作成します。 これにより、以下の例が示すように予測できない動作を引き起こす可能性 があります。

```
<?php
function test_global_ref() {
    global $obj;
    $obj = &new stdclass;
}

function test_global_noref() {
    global $obj;
    $obj = new stdclass;
}

test_global_ref();
var_dump($obj);
test_global_noref();
var_dump($obj);
?>
```

この例を実行すると以下の出力が結果として表示されます。

```
NULL
object(stdClass)(0)  {
}
```

類似の動作が*static*命令にも適用されます。 リファレンスは静的に保存することができません。

```
<?php
function &get_instance_ref() {
    static $obj;

    echo 'Static object: ';
    var_dump($obj);
    if (!isset($obj)) {
        // Assign a reference to the static variable
        $obj = &new stdclass;
    }
    $obj->property++;
    return $obj;
}

function &get_instance_noref() {
    static $obj;

    echo 'Static object: ';
    var_dump($obj);
    if (!isset($obj)) {
        // Assign the object to the static variable
        $obj = new stdclass;
    }
    $obj->property++;
    return $obj;
}

$obj1 = get_instance_ref();
$still_obj1 = get_instance_ref();
echo "¥n";
$obj2 = get_instance_noref();
$still_obj2 = get_instance_noref();
?>
```

この例を実行すると以下の出力となります。

```
Static object: NULL
```

```
Static object: NULL

Static object: NULL
Static object: object(stdClass)(1) {
   ["property"]=>
   int(1)
}
```

この例は、static変数にリファレンスを代入した時に *&get_instance_ref()*関数を2回目に コールした際に保持されていないことを示しています。

可変変数

変数名を可変にできると便利なことが時々あります。可変変数では、変数 名を動的にセットし使用できます。通常の変数は、次のような命令でセッ トします。

```
<?php
$a = 'hello';
?>
```

可変変数は、変数の値をとり、変数の名前として扱います。上の例では、 *hello* は、ドル記号を二つ使用することにより、 変数の名前として使用することができます。つまり、

```
<?php
$$a = 'world';
?>
```

ここまでで、二つの変数が定義され、PHP シンボルツリーに定義されてい ます。これらは、"hello" を値とする*$a*と "world" を値とする*$hello*です。そこで、次の命令

```
<?php
echo "$a ${$a}";
?>
```

の出力は、次の命令と全く同じとなります。

```
<?php
echo "$a $hello";
?>
```

すなわち、両方共、hello worldを出 力します。

可変変数を配列で使用する際には、曖昧さの問題を解決する必要がありま す。つまり、*$$a[1]*と書いた場合、 *$a[1]*を変数として使用したいのか、*$$a*を変数とし [1] を変数の添え字としたいのかを、 パーサが知る必要があるのです。この曖昧さを解決するには、前者では *${$a[1]}*とし、後者では *${$a}[1]*とする構文を用います。

警告

関数やクラスメソッドの内部で、可変変数と PHP の スーパーグローバル配列 とを組み合わせては使用できないということに注意してください。 *$this* も特別な変数であり、 動的に参照することはできません。

PHPの外部から来る変数

HTML フォーム (GET と POST)

フォームが PHP スクリプトに投稿された時、フォームから渡された全て の変数は PHP により自動的にスクリプトから使用可能となります。 この情報にアクセスする手段は複数あります。例を以下に示します。

Example#1 簡単なHTMLフォーム

```
<form action="foo.php" method="post">
      Name:  <input type="text" name="username" /><br />
      Email: <input type="text" name="email" /><br />
       <input type="submit" name="submit" value="Submit me!" />
</form>
```

特定の設定や個別の設定に依存し、HTMLフォームからのデータにアクセス する手段は多くあります。いくつかの例を以下に示します。

Example#2 簡単なPOST HTMLフォームからのデータにアクセスする

```
<?php
// PHP 4.1.0以降で利用可能

     echo $_POST['username'];
     echo $_REQUEST['username'];

     import_request_variables('p', 'p_');
     echo $p_username;

// Available since PHP 3. As of PHP 5.0.0, these long predefined
```

```
// variables can be disabled with the register_long_arrays directive.

     echo $HTTP_POST_VARS['username'];

// PHPディレクティブregister_globals = onの場合に利用可能。
// PHP 4.2.0以降、register_globalsのデフォルト値はoffとなっています。
// この方法の使用/依存は推奨されません。

    echo $username;
?>
```

GETフォームを使用した場合も同じですが、かわりに適当な定義済みの GET変数を使用するところが異なります。 GETは、QUERY_STRING (URLの '?'の後の情報)にも代入されます。 例えば、 *http://www.example.com/test.php?id=3*には、 *$_GET['id']*によりアクセス可能なGETデータ が含まれます。 $_REQUEST および import_request_variables()も参照ください。

> **注意**: *$_POST*および*$_GET*のような スーパーグローバル 配列がPHP 4.1.0で利用可能となっています。

前記のようにPHP 4.2.0より前のバージョンでは、 register_globals のデフォルト値は*on*でした。 PHP 3では、常にonとなっています。 PHPコミュニティは、このディレクティブに依存しないことを推奨し、 このオプションが*off*でのコードの動作を仮定 することを推奨しています。

> **注意**: magic_quotes_gpcの設定 はGET、POSTそしてCookieの値に影響します。onになっていると (It's "PHP!") という値は自動的に (It\'s \"PHP!\") となり、DBへの挿入時の エスケープが不要になります。addslashes()、 stripslashes()そしてmagic_quotes_sybaseも参照してください。

PHPではフォーム変数のコンテキスト内で配列が使用可能です(FAQの関連箇所も参照してください)。 例えば、関連する変数をグループ化したり、select inputで複数の値を 取得するといったことが可能です。フォームを同じスクリプトに投稿し、 投稿したデータを表示する例を示します。

Example#3 より複雑なフォーム変数

```
<?php
if ($_POST) {
     echo '<pre>';
      echo htmlspecialchars(print_r($_POST, true));
     echo '</pre>';
}
?>
<form action="" method="post">
      Name:   <input type="text" name="personal[name]" /><br />
      Email:  <input type="text" name="personal[email]" /><br />
     Beer: <br />
      <select multiple name="beer[]">
            <option value="warthog">Warthog</option>
            <option value="guinness">Guinness</option>
            <option value="stuttgarter">Stuttgarter Schwabenbräu</option>
     </select><br />
      <input type="submit" value="submit me!" />
</form>
```

PHP 3では、配列変数は1次元配列に限定されていました。PHP 4以降ではこの ような制約はありません。

IMAGE SUBMIT 変数名

フォームを投稿する際、次のタグのように標準の投稿ボタンの代わりに 画像を使用することができます。

```
<input type="image" src="image.gif" name="sub" />
```

画像のどこかがクリックされた場合、二つの変数 sub_x および sub_y が付け加えられてこのフォームはサーバーに転送されます。これらの変 数は、ユーザーがこの画像をクリックした座標を示しています。経験の ある人は、ブラウザにより送られた変数の名前においてアンダースコア がピリオドになってしまっていることを心配するかもしれません。 しかし、PHP はピリオドをアンダースコアに自動的に変換します。

HTTP Cookie

PHP は、» Netscapeの規約 に定 義されたHTTP Cookieを完全にサポートします。Cookieは、リモート ブラウザにデータを保持し、再訪するユーザーを追跡し、特定する機構 です。setcookie() 関数によりCookieをセットす ることができます。Cookieは、HTTP ヘッダの一部なので、SetCookie 関数をブラウザに何かを出力する前にコールする必要があります。 この制約は、header() 関数のものと同じです。 Cookieのデータは、*$_COOKIE*, *$HTTP_COOKIE_VARS*のような適当なCookieデータ 配列で参照可能です。また、 *$_REQUEST*でも 参照可能です。詳細および例については、setcookie()のマニュアルページを参照してください。

単一のCookieに複数の値を代入したい場合は、配列として 代入することが可能です。以下に例を示します。

```
<?php
   setcookie("MyCookie[foo]", 'Testing 1', time()+3600);
  setcookie("MyCookie[bar]", 'Testing 2', time()+3600);
?>
```

上記スクリプトにおいては、2つの異なるCookieを生成されますが、 この場合、スクリプトではMyCookieという単一の配列になります。 一つのCookieに複数の値を設定したい場合、最初の値に serialize()または explode()を用いることを考えてください。

Cookieは、パスまたはドメインが異ならない限り、 以前のクッキーをブラウザ上の同じ名前の変数に置き換えることに 注意してください。 さて、買い物かご(Shopping Cart) プログラムの場合、カウンタを保持し、 受け渡したいと思うかもしれません。 これは、次のようになります。

Example#4 setcookie()の例

```php
<?php
if (isset($_COOKIE['count'])) {
    $count = $_COOKIE['count'] + 1;
} else {
    $count = 1;
}
setcookie('count', $count, time()+3600);
setcookie("Cart[$count]", $item, time()+3600);
?>
```

外部変数名のドット

通常、PHP はスクリプトに渡された変数の名前を変更しません。しかし、 ドット(ピリオド、終止符)はPHPの変数名で有効な文字ではないというこ とに注意する必要があります。次の例を見てみましょう。

```php
<?php
$varname.ext;  /* 無効な変数名 */
?>
```

ここで、パーサは、*$varname*という名前の変数の後に 文字列結合演算子があり、その後に、裸の文字列(すなわち、既知のキー または予約語にマッチしない引用符無しの文字列) 'ext' が続くとして 解釈します。この場合、明らかに意図する結果にはなりません。

重要なことを記述しておくと、このため、外部変数に含まれるドットを PHP は自動的にアンダースコアに変換します。

変数の型の定義

PHPは、変数の型を定義し、必要に応じて変換します。このため、ある変 数の型がある時点で常に明らかであるわけではありません。PHPは、変数 の型を調べる複数の関数をサポートしています。それらは、 gettype(), is_array(), is_float(), is_int(), is_object(), is_string() です。 型の章も参照ください。

定数

目次

- 自動的に定義される定数

定数は簡単な値のためのID(名前)です。この名前が示すように、定数の値は スクリプト実行中に変更できません (マジック定数 は例外で、これらは実際は定数ではありません)。 デフォルトで定数では大文字小文字を区別します。慣習的に、 定数は常に大文字で表記されます。

定数の名前は、PHP のラベルと同じ規則に従います。有効な定数の名前は、 文字またはアンダースコアで始まり、任意の数の文字、数字、 アンダースコアが後に続きます。正規表現で示すと次のようになります。 *[a-zA-Z_¥x7f-¥xff][a-zA-Z0-9_¥x7f-¥xff]**

ヒント

ユーザレベルでの命名の手引き もご覧になるとよいでしょう。

Example#1 有効/無効な定数名の例

```php
<?php

// 有効な定数名
define("FOO",     "something");
define("FOO2",    "something else");
define("FOO_BAR", "something more");

// 無効な定数名
define("2FOO",    "something");

// 有効だが、避けるべき。
// 将来 PHP に定数の予約語が追加された場合に
// スクリプトが動作しなくなる可能性がある
define("__FOO__", "something");

?>
```

> **注意**: 本節の目的においては、文字は a-z, A-Z, および127から255まで (0x7f-0xff)のASCII文字を指します。

superglobalsと同様に定数のスコープはグローバルです。 つまり、スコープによらずスクリプトの中ではどこでも定数に アクセスできます。スコープの詳細についてはマニュアルの 変数のスコープ をご覧ください。

構文

define() 関数を使用することにより、 定数を定義することが可能です。定数が一度定義されると、 変更または未定義とすることはできません。

定数に指定できるのは、スカラデータ (boolean, integer, double, string) のみです。 resource の定数を指定しないでください。

単に定数の名前を指定することにより、その値を得ることが可能です。 変数とは異なり、その前に *$* は不要です。 定数の名前を動的に得る必要がある

場合、定数の値を読むために関数 constant() を使用することも可能です。 定義済の定数の一覧を得るには、 get_defined_constants() を使用してください。

> **注意**: 定数と(グローバル)変数は、異なる名前空間にあります。 例えば、**TRUE** と *$TRUE* は違うものを意味します。

未定義の定数を使用した場合、ちょうどstringとして コールしたかのように(CONSTANT vs "CONSTANT")、 PHPはその定数自体の名前を使用したと仮定します。 この際、E_NOTICE が発生します。 ある定数が設定されているかどうかを知るには、 defined() 関数を使用してください。 なぜ$foo[bar]が間違っている （まず*bar*を定数としてdefine()しなければ） のかというマニュアルもご覧ください。 定数がセットされているかを単にチェックするには defined()を使用してください。

変数との違いは次のようになります。

- 定数は、前にドル記号($)を要しません。
- 定数を定義することができるのは、define()関数 のみです。単なる代入による定義はできません。
- 定数は、定義することができ、変数のスコープ規則に関係なく、あら ゆる場所からアクセス可能です。
- 定数は一度設定されると再定義または未定義とすることはできません。
- 定数は、スカラー値としてのみ評価可能です。

Example#2 定数の定義

```
<?php
define("CONSTANT", "Hello  world.");
echo CONSTANT; //  "Hello  world."を出力
echo Constant; //  "Constant"  を出力し、警告 (notice) を発行
?>
```

オブジェクト定数 も参照ください。

自動的に定義される定数

PHPには実行されるスクリプトで使用可能な多くの 定義済みの定数があります。 しかし、これらの定数の多くは、種々の拡張モジュールにより作成され、 動的なロードやコンパイル時の組込みにより、これらの拡張モジュールが 使用可能である場合にのみ定義されます。

使われ方によって変化する自動的に定義される定数（マジカル定数）が5つあります。 例えば、**__LINE__**はスクリプト上において 呼び出された行番号です。特別定数は大文字小文字を区別しません。 内容は以下のとおりです:

PHP の "マジック" 定数

名前	説明
__LINE__	ファイル上の現在の行番号。
__FILE__	ファイルのフルパスとファイル名。インクルードされるファイルの 中で使用された場合、インクルードされるファイルの名前が返されます。 PHP 4.0.2 以降では、**__FILE__** は常に絶対パスです。それより前のバージョンでは、場合によっては 相対パスが返されることもあります。
__FUNCTION__	関数名（PHP4.3.0で追加されました）。 PHP 5以降、この定数は宣言時の関数名(ケース依存)を返します。 PHP 4では、この値は常に小文字で返されました。
__CLASS__	クラス名（PHP4.3.0で追加されました）。 PHP 5以降、この定数は宣言時のクラス名(ケース依存)を返します。 PHP 4では、この値は常に小文字で返されました。
__METHOD__	クラスのメソッド名（PHP5.0.0で追加されました）。 メソッド名は宣言時と同じ(ケース依存)を返します。

get_class(), get_object_vars(), file_exists(), function_exists()も参照してください。

式

式は、PHP における最も重要な基盤石です。PHPにおいては、 ほとんど全てのものは式で記述されます。 最も簡単で最も正確な式の定義は、"全ての式には値がある。" です。

考えられる簡単な例は、定数と変数です。 "$a = 5" と入力すると、$a に '5' を代入することになります。 '5' は、明らかに、 5 という値です。 言葉を変えると '5' は 5 という値を有する式なのです。 (この場合、'5' は整数定数です。)

この代入の後、$a の値は、5 であることが期待されます。 よって、$b = $a と書いた場合、$b = 5 と書いたのと 同じように動作することが期待されます。 言い換えると $a は 5 という値を持つ式なのです。 全てが正しく動作する場合、何が起こるかをこのことが正確に表現しています。

式をもう少し複雑にしたのが関数です。 例えば、次の関数を考えてみましょう。

```
<?php
function  foo  ()  {
    return  5;
}
```

```
?>
```

あなたが関数の概念に慣れていると仮定すると (そうでない場合は、関数 に関する章を参照ください。)、 *$c = foo()* と入力することは、 *$c = 5* と書くことと本質的に全く同じで あると予想されたかもしれません。この予想は、正しいです。 関数は、その返り値を値とする式なのです。 foo() は 5 を返すので、式 'foo()' の値は 5 です。 通常、関数は、決まった数だけを返すのではなく、何かを計算します。

もちろん、PHP の値は整数である必要はありませんし、 多くの場合、そうではありません。 PHP は、4 種類のスカラー型: 整数(integer)、 浮動小数点数(float)、文字列(string)、 真偽値(boolean) をサポートします。 (スカラーとは、配列とかと異なり、より小さな部分に'分割する'ことが できない値のことです。) PHP は、2種類の複合(非スカラー)型(配列とオブジェクト)もサポートします。 これらの型の値は、変数に代入することができ、 関数からの返り値とすることができます。

PHP は、他の多くの言語が行うのと同じ手法で、 更に多くの式を使用可能です。 PHP 3 は、ほとんど全てが式であるという意味で、式指向の言語です。 既に取り扱った '$a=5' という例について考えてみましょう。 この式には、整数定数の '5' と 5 に更新された $a の値という 2 つの値が現れているということに容易に気づくことでしょう。 しかし、実際には、ここにはもうひとつの値が含まれています。 それは、代入自体の値です。 代入式は、それ自体、代入値を評価します。 この場合、その値は 5 になります。 このことは、実際には、'$a = 5' は、それが何をするかによらず、 値 5 を有する式であることを意味します。 つまり、 '$b = ($a = 5)' のように書くことは、'$a = 5; $b = 5;' と書くのと 同様なのです。(セミコロンは、文の終わりを示します。) 代入は、右から左へ実行されるため、'$b = $a = 5' と書くことも 可能です。

式の配置に関する別の良い例は、前置、後置加算子、あるいは減算子です。 PHP と他の多くの言語のユーザーは、variable++ や variable-- といった表記法に慣れていることでしょう。 これらは、加算子および減算子です。 PHP/FI 2 においては、文 '$a++' は値を持ちませんでした。 (この文は、式ではありませんでした。) このため、この式に代入したり、なんらかの手法でこれを使用することが できませんでした。 PHP は、C 言語のようにこれらの式を同時に作ることにより 加算子、減算子の機能を拡張しました。 PHP においては、C 言語のように、 前置加算と後置加算という、2 種類の加算があります。 前置加算と後置加算は、両方とも、基本的には変数を増加させ、 変数に対する効果は同じです。 異なっているのは、加算する式の値です。 前置加算は、'++$variable' と書かれますが、 加算後の値を評価します。 (PHP はその値を読む前に変数を増加させるので、'前置加算(pre-increment)' という名前がついています。) 後置加算は、'$variable++' と書かれますが、加算される前の $variable の元の値を評価します。 (PHP は、その値を読んだ後に変数を増加させるので、 '後置加算(post-increment)' という名前がついています。)

比較演算子は、 極めて標準的な式です。 比較演算子は、**FALSE** または **TRUE** のどちらかを値とします。 PHPは、>(大なり)、 >=(大なりイコール)、=(イコール)、 <(小なり)、<=(小なりイコール)をサポートします。 PHP 言語は、いくつかの厳密な等価演算子: === (イコールかつ同じ型) そして !== (イコールではないまたは型が違う) も サポートします。 これらの式は、*if*文のような条件式の内部で一般的に 使用されます。

式の最後の例として、ここでは、演算子+代入式の複合演算式 を扱います。 既にご存知のように、$a に 1 を加えたい場合は、'$a++' または '++$a' と書くだけで十分です。 しかし、1より大きな数、例えば 3 を加えたい場合は、どうすればよいの でしょう? '$a++' を複数回使うこともできますが、当然これはあまり効率的で快適な 手法ではありません。 多くの場合は、'$a = $a + 3' と書かれます。 '$a + 3' は、$a + 3 の値を評価し、$a に代入します。 この結果、$a に 3 が加えられます。 PHP においては、C のような他の言語と同様に、 この例をより短かく書くことができます。 これにより、より明確になり、同時に理解も迅速になります。 $a の現在の値に 3 を加える式は、 '$a += 3' と書くことができます。 この式の正確な意味は、 "$a の値を取得し、それに 3 を加え、$a に再代入しなさい。" です。 より短く、明確になっただけでなく、実行もより高速になります。 '$a =+ 3' の値は、通常の代入と同様に、代入された値です。 この値は 3 ではなく、$a に 3 を加えた加算値 (この値が、$a に代入された値です。)であることに注意してください。 '$a -= 5' ($a から 5 を引く)や '$b *= 7' ($b に 7 をかける)等のように、 全ての 2 項演算子は、この演算子+代入式のモードで使用することができます。

もう一種類、terniary 条件文という式がありますが、 他の言語で見たことがない場合には理解できないかもしれません。

```
<?php
$first ? $second : $third
?>
```

最初の部分式の値が **TRUE** (非ゼロ)の場合、二番目の部分式が評価され、 この条件文の結果となります。 そうでない場合、三番目の部分式が評価され、この文の値となります。

次の例は、前置および後置加算子と多少一般的な式の理解を 助けてくれることでしょう。

```
<?php
function double($i) {
     return $i*2;
}
$b = $a = 5;          /* 値 5 を $a と $b に代入します */
$c = $a++;            /* 後置加算なので、$c に代入される値は、$a の
                         元の値 (5) です */
$e = $d = ++$b;       /* 前置加算なので、$d と $e に代入される値は、
                         加算後の $b の値 (6) です */

/* ここままで、$d と $e は、6 です */

$f = double($d++);    /* $f には、$d が加算される前の値を2倍した値、
                         つまり 2*6 = 12 が、代入されます。
$g = double(++$e);    /* $g には、$e が加算された後の値を2倍した値、
                         つまり 2*7 = 14 が、代入されます。
$h = $g += 10;        /* まず、$g に 10 が加算され、24 になります。
                         代入値 (24) は、$h に代入されます。
                         そして、$h も同様に 24 になります。 */
?>
```

式が、文として扱われることがあります。 この場合、文は、'式';' 、つまり式の後にセミコロンがついた形式です。 '$b=$a=5' において、$a=5 は有効な式ですが、自身を値とする文では ありません。しかし、'$b=$a=5;' は有効な文です。

最後に、有益な事項として式の論理値について説明します。 多くのイベント、主に条件付き実行とループにおいて、 式の特定な値には関心がないが、**TRUE** または **FALSE** のどちらを 意味するかに関心があるということがあります。 定数**TRUE**と**FALSE**(大文字小文字を区別しない)は、論理型の値がとり得

る 値です。必要に応じて式は論理値に変換されます。詳細な手法については、 型キャストに関するセクション を参照ください。

PHP は、完全で強力な式の実装を提供します。 それを完全に記述することは、このマニュアルの範囲を超えています。 上記の例は、式とは何か、そして、便利な式をどうやって作るかということ に関して良いアイデアを与えるに違いありません。 本マニュアルの残りの部分では*expr*という マークを使用しますが、これは PHP の有効な式を意味します。

演算子

目次

演算子とは、ひとつ以上の値 (あるいはプログラミング用語における「式」) から別の値 (制御構造が式になるように) を生み出すものです。 つまり、値を返す関数や制御構造 (たとえば print) は演算子と考えられますし、 何も値を返さないもの (たとえば echo) はそれ以外のものとなります。

演算子には 3 種類あります。ひとつめは単項演算子で、これはひとつの値に 対してのみ作用します。例えば ! (否定演算子) や ++ (加算子) などです。ふたつめは二項演算子と呼ばれるものです。PHP がサポートしている演算子の ほとんどはここに含まれ、その一覧は 演算子の優先順位 にあります。

最後のグループは、三項演算子 ?: です。これは、2 つの文や実行経路から選択すると いうよりも、3 番目の式に応じて 2 つの式から選択するために使用されるべきです。 この演算子を使用する式は、括弧で囲んでおくことをお勧めします。

演算子の優先順位

演算子の優先順位は、二つの式が"緊密に"結合している度合いを指定します。 例えば、式 *1 + 5 * 3* の答えは *16* になり、*18* とはなりません。 これは乗算演算子("*")は、加算演算子("+")より高い優先順位を有するか らです。必要に応じて強制的に優先順位を設定するために括弧を使用する ことが可能です。例えば、*18*と評価するためには、 *(1 + 5) * 3* とします。 演算子の優先順位が等しい場合は、左から右へ順に評価されます。

以下の表では、優先順位が高い順に演算子を挙げています。 同じ行にある演算子は優先順位が等しくなります。そのような場合は、 結合時の評価にしたがって評価順が決まります。

演算子の優先順位

結合時の評価	演算子	追加情報
結合しない	new	new
left	[	array()
結合しない	++ --	加算子/減算子
結合しない	~ - (int) (float) (string) (array) (object) @	型
結合しない	instanceof	型
right	!	論理演算子
left	* / %	代数演算子
left	+ - .	代数演算子 そして 文字列演算子
left	<< >>	ビット演算子
結合しない	< <= > >=	比較演算子
結合しない	== != === !==	比較演算子
left	&	ビット演算子 そして リファレンス
left	^	ビット演算子
left	¦	ビット演算子
left	&&	論理演算子
left	¦¦	論理演算子
left	? :	三項演算子

結合時の評価	演算子	追加情報
right	= += -= *= /= .= %= &= ¦= ^= <<= >>=	代入演算子
left	and	論理演算子
left	xor	論理演算子
left	or	論理演算子
left	,	さまざまな利用法

結合時の評価が left の場合は式が左から右に評価され、一方 right の場合は その逆となります。

Example#1 結合時の評価

```
<?php
$a = 3 * 3 % 5; // (3 * 3) % 5 = 4
$a = true ? 0 : true ? 1 : 2; // (true ? 0 : true) ? 1 : 2 = 2

$a = 1;
$b = 2;
$a = $b += 3; // $a = ($b += 3) -> $a = 5, $b = 5
?>
```

コードの可読性を高めるためには括弧を使用します。

> **注意**: = は他のほとんどの演算子よりも優先順位が低いはずなのにもかかわらず、 PHP は依然として *if (!$a = foo())* のような式も許します。この場合は *foo()* の返り値が *$a* に代入されます。

代数演算子

学校で習った基礎代数を憶えていますか? この演算子はそれらと同様に動作します。

代数演算子

例	名前	結果
-$a	負にする	$a の逆
$a + $b	加算	$a および $b の合計
$a - $b	減算	$a と $b の差
$a * $b	乗算	$a および $b の積
$a / $b	除算	$a および $b の商
$a % $b	剰余	$a を $b で割った余り

除算演算子 ("/") の返す値は浮動小数点数となります。 ただし、ふたつのオペランドがともに整数 (あるいは整数に変換できる文字列) であり、かつ結果が割り切れる場合には整数値を返します。

剰余演算子は、まず両方のオペランドを整数に直し (小数点以下を切り捨てます) てから処理を行います。

> **注意**: *$a* が負の場合、*$a % $b* は負の値となることを覚えておきましょう。

マニュアルの 数学関数の項も参照してください。

代入演算子

代入演算子の基本となるものは "=" です。この演算子に関して最初に 思い付く意味は"等しい"であるかもしれません。しかし、そうではありません。 本当は、左オペランドに右オペランドの式の値を設定する("得て代入する") ことを意味します。

代入式の値は、代入される値です。つまり、"$a = 3" の値は、3 です。 これにより、以下のようなトリッキーなことができるようになります。

```
<?php

$a = ($b = 4) + 5; // $a は 9 に等しく、$b は 4 にセットされます。

?>
```

基本代入演算子に加えて、全ての バイナリ演算子、配列結合および文字列演算子に関して 「複合演算子」があります。 これにより、式の中の値を使用し、その値をその式の結果とすることができます。 例えば、

```
<?php

$a = 3;
$a += 5; // $a を 8 にセットします。$a = $a + 5; と同じです。
$b = "Hello ";
$b .= "There!"; // $bを"Hello There!"にセットします。$b = $b . "There!";と同じです。
```

```
?>
```

代入は、元の変数を新しい変数にコピーする(値による代入)ため、 片方の変数に対する変更はもう片方に影響を与えないということに 注意してください。この動作により、密なループの内側で大きな配列のようなものを コピーする必要がある場合には問題を生じる可能性があります。 PHP 4 以降では、$var = &$othervar; 構文により参照による代入をサポートしていますが、PHP 3 ではサポート しません。'参照による代入'は、両方の変数が同じデータを指し、コピーを 行わないことを意味します。参照に関する詳細については、 リファレンスの説明も 参照ください。 PHP 5 では、新しいキーワード clone を使用して明示的に指定しない限り、オブジェクトは自動的に参照による代入になります。

ビット演算子

ビット演算子は、整数における特定のビットをオンまたはオフにすることを 可能にします。もし左辺値と右辺値共に文字列であった場合にはビット演算子は 文字の ASCII 値に対して作用します。

```
<?php
echo 12 ^ 9; //  '5'を出力します

echo "12" ^ "9"; //  バックスペース文字を出力します(ascii  8)
                    // ('1' (ascii 49)) ^ ('9' (ascii 57)) = #8

echo "hallo" ^ "hello"; // ascii値の  #0  #4  #0  #0  #0  を出力します
                        // 'a' ^ 'e' = #4
?>
```

ビット演算子

例	名前	結果
$a & $b	ビット積	$a および $b の両方にセットされているビット
$a ¦ $b	ビット和	$a または $b のどちらかにセットされているビット
$a ^ $b	排他的論理和	$a または $b にセットされており、両方にセットされていないビット
~ $a	否定	$a にセットされているビットはセットせず、そうでないものは逆にする
$a << $b	左シフト	$a のビットを左に $b ビットシフトする(各シフトは "2をかける" ことを意味します)
$a >> $b	右シフト	$a のビットを右に $b ビットシフトします (各シフトは "2で割る" ことを意味します)

警告

32 ビットシステムでは 32 ビット以上の右シフトは行わないでください。 また、結果が 32 ビットを超えてしまうような左シフトも行わないでください。

比較演算子

比較演算子は、その名前が示すように、二つの値を比較します。

比較演算子

例	名前	結果
$a == $b	等しい	$a が $b に等しい時に TRUE。
$a === $b	等しい	$a が $b に等しく同じ型でである場合に TRUE (PHP 4 で導入)。
$a != $b	等しくない	$a が $b に等しくない場合に TRUE。
$a <> $b	等しくない	$a が $b に等しくない場合に TRUE
$a !== $b	等しくない	$a が $b と等しくないか、同じ型でない場合に TRUE (PHP 4 で導入)。
$a < $b	より少ない	$a が $b より少ない時に TRUE。
$a > $b	より多い	$a が $b より多い時に TRUE。
$a <= $b	より少ないか等しい	$a が $b より少ないか等しい時に TRUE。
$a >= $b	より多いか等しい	$a が $b より多いか等しい時に TRUE。

整数値を文字列と比較する際、文字列が 数値に変換されます。 数値形式の文字列を比較する場合、それは整数として比較されます。これらの ルールは、 switch 文にも適用されます。

```
<?php
var_dump(0 == "a"); // 0 == 0 -> true
var_dump("1" == "01"); // 1 == 1 -> true
var_dump("1" == "1e0"); // 1 == 1 -> true

switch ("a") {
case 0:
     echo "0";
     break;
case "a": // "a" は 0 にマッチするので、決してここにはたどりつきません
```

```
    echo "a";
    break;
}
?>
```

多くの型では、以下の表に（上から順に）したがって比較が行われます。

さまざまな型の比較

第 1 オペランドの型	第 2 オペランドの型	結果
null または string	string	NULL を "" に変換し、数値または文字として比較します
bool または null	あらゆる型	bool に変換し、FALSE < TRUE と判断します
object	object	組み込みクラスには独自の比較基準が定義されています。それ以外の クラスは比較できません。同じクラスであるかどうかは - プロパティが 同じ値であるかどうかを配列形式で比較（PHP 4）、PHP 5 では ここで説明されています。
string, resource または number	string, resource または number	文字列やリソースを数値に変換し、算術演算を行います
array	array	要素数の少ない配列のほうが小さくなります。オペランド 1 のキーが オペランド 2 に存在しない場合、配列は比較できません。そうでない場合は 個々の要素の値を比較します（以下の例を参照ください）
array	あらゆる型	array のほうが常に大きくなります
object	あらゆる型	object のほうが常に大きくなります

Example#1 一般的な配列の比較

```
<?php
// 標準の比較演算子を用いて、配列はこのように比較されます
function standard_array_compare($op1, $op2)
{
    if (count($op1) < count($op2)) {
        return -1; // $op1 < $op2
    } elseif (count($op1) > count($op2)) {
        return 1; // $op1 > $op2
    }
    foreach ($op1 as $key => $val) {
        if (!array_key_exists($key, $op2)) {
            return null; // uncomparable
        } elseif ($val < $op2[$key]) {
            return -1;
        } elseif ($val > $op2[$key]) {
            return 1;
        }
    }
    return 0; // $op1 == $op2
}
?>
```

strcasecmp(), strcmp()、 配列演算子, マニュアルの 型 のセクションも参照してください。

三項演算子

もうひとつの条件演算子として "?:"（あるいは三項）演算子があります。

Example#2 デフォルト値を設定する

```
<?php
// 三項演算子の使用例
$action = (empty($_POST['action'])) ? 'default' : $_POST['action'];

// 上記は以下の if/else 式と同じです。
if (empty($_POST['action'])) {
    $action = 'default';
} else {
    $action = $_POST['action'];
}

?>
```

(expr1) ? (expr2) : (expr3) という式は、式1 が **TRUE** の場合に 式2 を、 式1 が **FALSE** の場合に 式3 を値とします。

> **注意**: 三項演算子は式であり、値としては評価されずに式の結果として評価される ことに注意してください。演算結果をリファレンスとして返したい場合に、 これを知っておくことが大切です。結果をリファレンスとして返す関数で *return $var == 42 ? $a : $b;* とすることはできず、 新しいバージョンの PHP では警告を発生します。
>
> **注意**: 三項演算子を "積み重ねて" 使用することは避けましょう。 ひとつの文の中で複数の三項演算子を使用した際の PHP の振る舞いは、少々わかりにくいものです。
>
> **Example#3 三項演算子のわかりにくい挙動**
>
> ```
> <?php
> // ぱっと見た感じでは、これは 'true' と表示されると思うでしょう。
> echo (true?'true':false?'t':'f');
> ```

```
// しかし、実際には上の出力結果は 't' です。
// なぜなら、三項演算子は左から右へ順に評価されるからです。

// 上のコードをもう少しわかりやすく書くと、このようになります。
echo ((true ? 'true' : 'false') ? 't' : 'f');

// まず、最初の式が 'true' と評価されます。この 'true' は
// (bool)true と評価されるので、それをもとに二番目の三項
// 演算子が評価されます。
?>
```

エラー制御演算子

PHP はエラー制御演算子(@)をサポートしています。PHP の式の前に付けた場合、 その式により生成されたエラーメッセージは無視されます。

track_errors 機能が 有効な場合、式により生成されたエラーメッセージはグローバル変数 $php_errormsg に保存されます。この変数はエラーが発生するたびに上書きされます。 そのため、この変数を使用したい場合には速やかに確認する必要があります。

```
<?php
/* 意図的なエラー */
$my_file = @file ('non_existent_file') or
    die ("Failed opening file: error was '$php_errormsg'");

// この演算子は関数だけでなく、全ての式で動作します。
$value = @$cache[$key];
// インデックス $key が存在しない場合でも、警告を発生しません。

?>
```

> **注意**: @演算子は、式 でのみ動作します。基本的なルールは次のようになります。 値を得ることができるものの場合、@ 演算子を前に付けることが可能です。 例えば、変数、関数、include()コール、定数等の 前にこの演算子をつけることが可能です。関数またはクラスの定義や *if*や*foreach* 等のような 条件構造の前にこの演算子を付けることはできません。

error_reporting() と、 エラー処理とログ出力関数 も参照してください。

警告

現在、誤差制御演算子プレフィックス"@"は、スクリプトの実行を 終了するような致命的なエラーの出力さえ抑圧します。このため、ある関数の エラー出力を抑制するために "@" を使用した場合、その関数が 利用できなかったり、ミスタイプがあった場合でも、原因を示すことなく その場所でスクリプトは終了してしまいます。

実行演算子

PHP は 1 種類の実行演算子、バッククォート (``) をサポートします。 シングルクォートではないことに注意してください! PHP は、バッククォートの中身をシェルコマンドとして実行しようとします。出力が返されます (すなわち、出力を単にダンプするのではなく、変数に代入することが できます)。 バッククォート演算子の使用は shell_exec() と等価です。

```
<?php
$output = `ls -al`;
echo "<pre>$output</pre>";
?>
```

> **注意**: バッククオート演算子は、セーフモード が有効な場合 もしくは shell_exec() が無効な場合は無効となります。

escapeshellcmd(), exec(), passthru(), popen(), shell_exec(), system() PHPをコマンドラインから使用する も参照してください。

加算子/減算子

PHP は C 言語形式の加算子/減算子（前置・後置ともに）をサポートします。

> **注意**: 加算子/減算子は bool 型の値には何も変更を加えません。 同じく **NULL** に減算子を適用しても何も起こりませんが、**NULL** に加算子を 適用すると *1* となります。

加算子/減算子

例	名前	効果
++$a	前置加算子	$a に 1 を加え、$a を返します。
$a++	後置加算子	$a を返し、$a に1を加えます。
--$a	前置減算子	$a から 1 を引き、$a を返します。
$a--	後置減算子	$aを返し、$a から 1 を引きます。

以下に簡単なスクリプトの例を示します。

```
<?php
echo "<h3>後置加算</h3>";
$a = 5;
echo "5 となります: " . $a++ . "<br>¥n";
echo "6 となります: " . $a . "<br>¥n";

echo "<h3>前置加算</h3>";
$a = 5;
echo "6 となります: " . ++$a . "<br>¥n";
echo "6 となります: " . $a . "<br>¥n";

echo "<h3>後置減算</h3>";
$a = 5;
echo "5 となります: " . $a-- . "<br>¥n";
echo "4 となります: " . $a . "<br>¥n";

echo "<h3>前置減算</h3>";
$a = 5;
echo "4 となります: " . --$a . "<br>¥n";
echo "4 となります: " . $a . "<br>¥n";
?>
```

PHP は、算術演算子で文字変数を扱った場合に C ではなく Perl の慣習に 従います。例えば、perl では 'Z'+1 は 'AA' を返しますが C では 'Z'+1 は '[' (ord('Z') == 90, ord('[') == 91) を返します。 文字変数はインクリメントされることは可能ですがデクリメントは不可能であるということ、 またプレーンな ASCII 文字 (a-z および A-Z) のみがサポートされるということに注意しましょう。

Example#1 文字変数に対する算術演算子の使用

```
<?php
$i = 'W';
for ($n=0; $n<6; $n++) {
    echo ++$i . "¥n";
}
?>
```

上の例の出力は以下となります。

```
X
Y
Z
AA
AB
AC
```

論理型に対する加算/減算は何の影響も及ぼしません。

論理演算子

論理演算子

例	名前	結果
$a and $b	論理積	$a および $b が共に **TRUE** の場合に **TRUE**
$a or $b	論理和	$a または $b のどちらかが **TRUE** の場合に **TRUE**
$a xor $b	排他的論理和	$a または $b のどちらかが **TRUE** でかつ両方とも **TRUE** でない場合に **TRUE**
! $a	否定	$a が **TRUE** でない場合 **TRUE**
$a && $b	論理積	$a および $b が共に **TRUE** の場合に **TRUE**
$a ¦¦ $b	論理和	$a または $b のどちらかが **TRUE** の場合に **TRUE**

"and" および "or" 演算子が 2 種類あるのは、演算が行われる際の優先順位が 異なっているためです (演算子の優先順位 を参照ください)。

Example#1 論理演算子についての説明

```
<?php

// foo() は決してコールされることはありません。これらの演算子は短絡評価を行うからです。
$a = (false && foo());
$b = (true  || foo());
$c = (false and foo());
$d = (true  or  foo());

// "||" の優先順位は "or" より高くなります
$e = false || true; // $e に代入されるのは、(false || true) の評価結果、つまり true です
$f = false or true; // $f には false が代入されます
var_dump($e, $f);

// "&&" の優先順位は "and" より高くなります
$g = true && false; // $g に代入されるのは、(true && false) の評価結果、つまり false です
$h = true and false; // $h には true が代入されます
var_dump($g, $h);
```

```
?>
```

上の例の出力は、たとえば以下のようになります。

```
bool(true)
bool(false)
bool(false)
bool(true)
```

文字列演算子

文字列の演算子は 2 種類あります。最初のは結合演算子('.')で、右引数と 左引数を結合したものを返します。2 番目は、結合代入演算子('.=')で、 この演算子は右側の引数に左側の引数を追加します。詳細は、代入演算子 を 参照ください。

```
<?php
$a = "Hello ";
$b = $a . "World!"; // $b は、"Hello World!" となります。

$a = "Hello ";
$a .= "World!"; // $a は、"Hello World!" となります。
?>
```

文字列 と 文字列 関数も参照してください。

配列演算子

Array Operators

例	名前	結果
$a + $b	結合	$a および $b を結合する。
$a == $b	同等	$a および $b のキー/値のペアが等しい場合に TRUE。
$a === $b	同一	$a および $b のキー/値のペアが等しく、その並び順が等しく、 かつデータ型も等しい場合に TRUE。
$a != $b	等しくない	$a が $b と等しくない場合に TRUE。
$a <> $b	等しくない	$a が $b と等しくない場合に TRUE。
$a !== $b	同一でない	$a が $b と同一でない場合に TRUE。

+ 演算子は、 右側の配列の要素のうち、左側の配列に存在しないキーのものを左側の配列に追加します。 重複しているキーは上書き「されません」。

```
<?php
$a = array("a" => "apple", "b" => "banana");
$b = array("a" => "pear", "b" => "strawberry", "c" => "cherry");

$c = $a + $b; // Union of $a and $b
echo "Union of ¥$a and ¥$b: ¥n";
var_dump($c);

$c = $b + $a; // Union of $b and $a
echo "Union of ¥$b and ¥$a: ¥n";
var_dump($c);
?>
```

このスクリプトを実行すると、以下のように出力されます。

```
Union of $a and $b:
array(3) {
   ["a"]=>
   string(5) "apple"
   ["b"]=>
   string(6) "banana"
   ["c"]=>
   string(6) "cherry"
}
Union of $b and $a:
array(3) {
   ["a"]=>
   string(4) "pear"
   ["b"]=>
   string(10) "strawberry"
   ["c"]=>
   string(6) "cherry"
}
```

同じキーと値を保持している場合に、配列が等しいとみなされます。

Example#1 配列の比較

```php
<?php
$a = array("apple", "banana");
$b = array(1 => "banana", "0" => "apple");

var_dump($a == $b); //  bool(true)
var_dump($a === $b); //  bool(false)
?>
```

配列と 配列関数も参照してください。

型演算子

instanceof を使用して、 ある PHP 変数が特定の クラス のオブジェクトのインスタンスであるかどうかを調べます。

Example#1 クラスでの instanceof の使用法

```php
<?php
class MyClass
{
}
class NotMyClass
{
}
$a = new MyClass;

var_dump($a instanceof MyClass);
var_dump($a instanceof NotMyClass);
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(false)
```

instanceof は、ある変数が 特定の親クラスを継承したクラスのオブジェクトのインスタンスであるかどうかを調べることもできます。

Example#2 継承したクラスでの instanceof の使用法

```php
<?php
class ParentClass
{
}
class MyClass extends ParentClass
{
}
$a = new MyClass;

var_dump($a instanceof MyClass);
var_dump($a instanceof ParentClass);
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(true)
```

最後に、*instanceof* は、ある変数が特定の インターフェイス を実装したクラスのオブジェクトのインスタンスであるかどうかも調べることができます。

Example#3 クラスでの instanceof の使用法

```php
<?php
interface MyInterface
{
}
class MyClass implements MyInterface
{
}
$a = new MyClass;

var_dump($a instanceof MyClass);
var_dump($a instanceof MyInterface);
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(true)
```

通常、*instanceof* ではリテラルのクラス名を使用しますが、 別のオブジェクトや文字列変数を使用することもできます。

Example#4 変数を用いた instanceof の使用法

```
<?php
interface MyInterface
{
}
class MyClass implements MyInterface
{
}
$a = new MyClass;
$b = new MyClass;
$c = 'MyClass';
$d = 'NotMyClass';
var_dump($a instanceof $b); // $b MyClass クラスのオブジェクトです
var_dump($a instanceof $c); // $c は文字列 'MyClass' です
var_dump($a instanceof $d); // $d は文字列 'NotMyClass' です
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(true)
bool(false)
```

注意すべき落とし穴があります。PHP 5.1.0 より前のバージョンでは、 *instanceof* は、クラス名が存在しない場合に __autoload() をコールしていました。 さらに、クラスが読み込めなかった場合に致命的なエラーが発生していました。 この問題の回避策としては、*動的なクラス参照* を使用するか、クラス名を含む文字列変数を使用します。

Example#5 PHP 5.0 における、クラス名検索時の致命的エラーの回避策

```
<?php
$d = 'NotMyClass';
var_dump($a instanceof $d); //  これで、致命的なエラーは発生しません
?>
```

上の例の出力は以下となります。

```
bool(false)
```

instanceof 演算子は PHP 5 から使用可能になりました。 それ以前には is_a() が使用されていましたが、 現在は is_a() は推奨されておらず、*instanceof* の使用が推奨されています。

get_class() および is_a() も参照ください。

制御構造

目次

導入

全ての PHP スクリプトは、一連の文からなります。 文としては、代入、関数コール、ループ、条件文、そして何もしない文(空の文) さえ使用すること

ができます。 文は、通常セミコロンで終了します。加えて、文は、中括弧によるグループ文で カプセル化することによりグループ化することが可能です。 グループ文は、同時に文にもなります。 本章では、様々な文の型が記述されています。

---

if

if 構文は、PHP を含む全ての言語において最も重要な 機能の一つです。 この構文は、命令の条件実行を可能にします。 PHP では、C 言語に似た次のような *if* 構文が使用されます。

```
if (式)
    文
```

式のセクションで 記述したように式は論理値で評価されます。 式が **TRUE** と評価された場合、 PHP は文を実行します。**FALSE** と評価された場合は、これを無視します。どのような値が **FALSE** と評価されるかについては論理値への変換 を参照してください。

以下の例は、*$a* が *$b* より大きい場合、aはbより大きい を表示します。

```
if ($a > $b)
     echo "aはbより大きい";
```

条件分岐させたい文が一つ以上ある場合もしばしばあります。 もちろん、各々の文を*if* 文で括る必要はありません。 代わりに、複数の文をグループ化することができます。 例えば、このコードは、*$a* が *$b* よりも大きい場合に aはbよりも大きいを表示し、 *$a* の値を *$b* に 代入します。

```
if ($a > $b) {
     echo "aはbより大きい";
     $b = $a;
}
```

*if*文は、他の*if*文の中で無限に入れ子にできます。 これは、プログラムの様々な部分の条件付実行について 完全な柔軟性を提供します。

---

else

ある条件が満たされている場合にある文を実行し、 その条件が満たされていない場合に別の文を実行したいと考えた ことが度々あるかと思います。 このために*else*があります。 *else*は、*if*文における式の値が **FALSE**の場合にある文を 実行するように*if*文を拡張します。 例えば、次のコードは、*$a*が *$b*よりも大きい場合に aはbより大きいと表示し、 そうでない場合に、 aはbよりも大きくないと表示します。

```
if ($a > $b) {
     echo "aはbよりも大きい";
} else {
     echo "aはbよりも大きくない";
}
```

else 文は、*if*式が **FALSE**と評価された場合のみ実行されます。 また、*elseif*式がある場合には、それも **FALSE**と評価された場合にのみ実行されます。 (elseif参照)

---

elseif

*elseif*は、その名前から分かるように、*if* と*else*の組み合わせです。*elseif*は、 *else*のように、元の*if*式の値が **FALSE**の場合に別の文を実行するように*if* 文を拡張します。 しかし、*else*とは異なり、*elseif*式が **TRUE**の場合にのみ代わりの式を実行します。 例えば、次のコードは、aはbより大きい、 aはbに等しい、 aはbより小さいを出力します。

```
if ($a > $b) {
     echo "aはbより大きい";
} elseif ($a == $b) {
     echo "aはbと等しい";
} else {
     echo "aはbより小さい";
}
```

複数の*elseif*を同じ*if*文の中で 使用することができます。 **TRUE**と評価された最初の*elseif*式 が実行されます。PHPでは、(単語二つで)'else if'と書くこともできます。 動作は(一単語の) 'elseif'と同じです。文法的な意味はやや異なっています。 (あなたが C 言語に詳しいとすると、C 言語のそれと同じ動作です。) しかし、最終的な両者の動作は全く同じです。

*elseif*文は、前にある全ての*if*文と *elseif*の値が**FALSE**であり、 カレントの*elseif*式の値が**TRUE**である 場合にのみ実行されます。

制御構造に関する別の構文

PHPは、いくつかの制御構造、つまり、*if*、 *while*、*for*、 *foreach*、*switch* に関する別の構文を提供します。 各構造において開き波括弧をコロン(:)、閉じ波括弧をそれぞれ *endif*;,*endwhile*;, *endfor*;,*endforeach*;, *endswitch*;に変更するのが 別の構文の基本的な形式となります。

```
<?php if ($a == 5): ?>
Aは5に等しい
<?php endif; ?>
```

上の例では、HTMLブロック"Aは5に等しい"はこの構文で 書かれた*if*文の内部で入れ子になっています。 このHTMLブロックは、*$a*が5の場合にのみ表示されます。

この方法は、*else*や*elseif*にも 同様に適用することができます。 次の例は、この形式で*if*文を*elseif* および*else*と共に使用しています。

```
if ($a == 5):
     echo "aは5に等しい";
     echo "...";
elseif ($a == 6):
     echo "aは6に等しい";
     echo "!!!";
else:
     echo "aは5でも6でもない";
endif;
```

より多くの例を参照するには、 while, for, ifも参照ください。

---

while

*while*ループは、PHPで最も簡単なタイプのループです。 このループは、CのWHILEループと同様の動作をします。 *while*ループの基本形は次のようになります。

```
while (式)
 文
```

*while*文の意味は簡単です。 *while*文は、式の値が**TRUE**である間、 入れ子の文を繰り返し実行することをPHPに指示します。 式の値は各反復処理の開始時にチェックされるので、ループ内の文の実行により この値が代わった場合でもループ実行は各ループを終るまで終わりません。 (PHPによるループ内の文の実行が1回分の反復に相当します) *while*式の値が初めから**FALSE**の 場合は、内部の文は一回も実行されません。

*if*文と同様に、波括弧で複数の文を囲うか、 以下に示す別の構文を用いることにより、同じ*while* ループの中に複数の文をグループ化することができます。

```
while (式):
    文
    ...
endwhile;
```

次の例は同じです。どちらも 1 から 10 までの数を出力します。

```
/* 例 1 */

$i = 1;
while ($i <= 10) {
     echo $i++;  /* 出力される値は、足される前の
                    $iの値です。
                    (後置加算) */
}

/* 例 2 */

$i = 1;
while ($i <= 10):
     echo $i;
     $i++;
endwhile;
```

---

do-while

*do-while*ループは、論理式のチェックが各反復の 最初ではなく最後に行われること以外は、*while*ループと 全く同じです。通常の*while*ループとの主な差は、 *do-while*ループは最低1回の実行を保証されていることです。 一方、通常の*while*ループは、実行されないかもしれません。 (論理式は各反復の最初でチェックされるので、 最初から値が**FALSE**である場合はループの実行は すぐに終わります。)

do-while ループの構文は次の一つのみです。

```
$i = 0;
do {
     echo $i;
} while ($i>0);
```

上記のループは必ず一度だけ実行されます。その原因は、最初の反復の後、 論理値のチェックを行った際に値が**FALSE**と なり($iは0より大きくない)、ループの実行が終了するためです。

優れたCプログラマは、コードブロック中での実行中止が可能な *do-while*ループの別の使用法について熟知している かもしれません。 これは、*do-while*(0)でコードを括り、 *break* 文を使用する方法です。次のコードは、この方法の例を示しています。

```
do {
     if ($i < 5) {
          echo "i は十分大きくはありません。";
          break;
     }
     $i *= $factor;
```

```
     if ($i < $minimum_limit) {
         break;
    }
     echo "iはOKです。";

     ...iを使った処理...

}  while(0);
```

この例をすぐに理解できなかったり、全く理解できなかったりしても 問題ありません。この'機能'を使用しなくても スクリプトコードを書くことができますし、強力なスクリプトでさえ 作成することが可能です。

---

for

for ループは、PHPで最も複雑なループです。 *for* は、Cのforループと同様に動作します。 *for*ループの構文は、次のようになります。

```
for (式1; 式2; 式3)
 文
```

最初の式(*式1*)は、ループ開始時に無条件に 評価(実行)されます。

各繰り返しの開始時に、*式2*が評価されます。 その式の値が**TRUE**が場合、ループは継続され、括弧 内の文が実行されます。値が**FALSE**の場合、ループの 実行は終了します。

各繰り返しの後、*式3*が評価(実行)されます。

各式は空にすることもできますし、複数の式をカンマで区切って指定することもできます。 *式2* でカンマ区切りの式を使用すると、 すべての式を評価します。しかし、結果として取得するのは最後の式の結果となります。 *式2* を空にすると、無限実行ループになります (PHP は、この状態を C 言語のように暗黙の内に **TRUE** とみなします)。 *for*論理式を使用するよりも条件付 *break* 文によりループを終了させる方が好ましい場合には、この機能は思った ほど便利ではないかもしれません。

次の例について考えてみましょう。以下の例はすべて 1 から 10 までの数を表示します。

```
/* 例 1 */

for ($i = 1; $i <= 10; $i++) {
     echo $i;
}

/* 例 2 */

for ($i = 1;;$i++) {
     if ($i > 10) {
         break;
     }
     echo $i;
}

/* 例 3 */

$i = 1;
for (;;) {
     if ($i > 10) {
         break;
     }
     echo $i;
     $i++;
}

/* 例 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
```

もちろん、最初の例(もしくは 4番目の例)が最善であると考えられます。 しかし、*for*ループにおいて空の式を使用できると、 多くの場合、便利だということに気づかれるかと思います。

PHPは、*for*ループ用に"コロン構文"もサポートします。

```
for (式1; 式2; 式3):
 文;
 ...;
endfor;
```

---

foreach

PHP 4には、Perlや他の言語とよく似た *foreach*構文があります。これにより 配列要素に関する反復処理が容易になります。この構造には、 2種類の構文があります。2番目の構文はあまり知られていませんが、 最初の構文の便利な拡張になっています。

```
foreach(array_expression  as  $value)
 文
foreach(array_expression  as  $key  =>  $value)
 文
```

最初の形式は、*array_expression*で指定した配列に 関してループ処理を行います。各ループにおいて現在の要素の値が *$value*に代入され、内部配列ポインタが一つ前に 進められます。(よって、次のループでは次の要素を見ることになります。)

2番目の形式も同様ですが、各ループで現在の要素のキーが変数 *$key*に代入されるところが異なります。

PHP 5 では、 オブジェクトのイタレーション を用いることもできます。

> **注意**: *foreach*の実行開始時に内部配列ポインタは、 配列の先頭要素を指すように自動的にリセットされます。 このため、*foreach*ループの前に reset()をコールする必要はありません。

> **注意**: 配列が リファレンス でない限り、*foreach*は、指定した配列自体に対してではなく、 そのコピーに対して処理を行います。 *foreach* は配列のポインタに副作用を及ぼします。 foreach の最中やその後で配列のポインタを使用する際は、まずポインタをリセットしてください。

PHP 5 以降、*$value* の前に & を付けることで、 容易に配列の要素の値を変更できるようになっています。 これにより、値をコピーするのではなく、リファレンス が代入されます。

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
// $arr は array(2, 4, 6, 8) となります
unset($value); // 最後の要素への参照を解除します
?>
```

この機能は、ループ処理される配列が参照可能である場合 (すなわち、 変数である) のみ使用可能です。

警告

foreach ループを終えた後でも、 *$value* は配列の最後の要素を参照したままとなります。 unset() でその参照を解除しておくようにしましょう。

> **注意**: *foreach* は、'@' によりエラーメッセージ出力を抑 制する機能をサポートしていません。

既にご存知かと思いますが、以下の文は機能的に等価です。

```
<?php
$arr = array("one", "two", "three");
reset($arr);
while (list(, $value) = each($arr)) {
    echo "Value: $value<br />¥n";
}

foreach ($arr as $value) {
    echo "Value: $value<br />¥n";
}
?>
```

以下の文も機能的に等価です。

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
     echo "Key: $key; Value: $value<br />¥n";
}

foreach ($arr as $key => $value) {
     echo "Key: $key; Value: $value<br />¥n";
}
```

使用法を示すためにその他の例を示します。

```
/* foreach の例 1: 値のみ */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
    echo "Current value of ¥$a: $v.¥n";
}

/* foreachの例2: 値 (説明用にキーを出力) */

$a = array (1, 2, 3, 17);

$i = 0; /* 説明用 */

foreach($a as $v) {
     echo "¥$a[$i] => $v.¥n";
     $i++;
}

/* foreachの例3: キーと値 */

$a = array(
     "one" => 1,
     "two" => 2,
     "three" => 3,
     "seventeen" => 17
);

foreach($a as $k => $v) {
     echo "¥$a[$k] => $v.¥n";
}
```

```
/* foreach の例4: 多次元配列 */
$a = array();
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
     foreach ($v1 as $v2) {
          echo "$v2¥n";
     }
}

/* foreach の例5: 動的配列 */

foreach(array(1, 2, 3, 4, 5) as $v) {
     echo "$v¥n";
}
```

---

break

*break*は、現在実行中の *for*, *foreach*, *while*, *do-while*, *switch*構造の実行を終了します。

*break*では、オプションの引数で ネストしたループ構造を抜ける数を指定することができます。

```
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list(, $val) = each ($arr)) {
    if ($val == 'stop') {
       break;     /* ここでは、'break 1;'と書くこともできる。 */
    }
    echo "$val<br />¥n";
}

/* オプション引数を使用する。 */
$i = 0;
while (++$i) {
     switch ($i) {
     case 5:
          echo "At 5<br />¥n";
          break 1;  /* switch構造のみを抜ける */
     case 10:
          echo "At 10; quitting<br />¥n";
          break 2;  /* switchとwhileを抜ける */
     default:
          break;
     }
}
```

---

continue

*continue*は、ループ構造において現在の繰り返しループ の残りの処理をスキップし、条件式を評価した後に 次の繰り返しの最初から実行を続けるために使用されます、

> **注意**: switch 命令はループ構造における*continue*の効果 を考慮してつくられていることに留意してください。

*continue*では、オプションの引数で 処理をスキップするループ構造のレベルの数を指定できます。

```
while (list($key,$value) = each ($arr)) {
     if (!($key % 2)) { // キーが偶数の組をスキップ
          continue;
     }
     do_something_odd($value);
}

$i = 0;
while ($i++ < 5) {
     echo "Outer<br />¥n";
     while (1) {
          echo "  Middle<br />¥n";
          while (1) {
               echo "  Inner<br />¥n";
               continue 3;
          }
          echo "This never gets output.<br />¥n";
     }
     echo "Neither does this.<br />¥n";
}
```

Omitting the semicolon after *continue* の後のセミコロンを省略すると混乱を生じる ことがあります。以下にすべきでないことの例を示します。

```
<?php
  for ($i = 0; $i < 5; ++$i) {
       if ($i == 2)
            continue
       print "$i¥n";
  }
?>
```

これは、以下のような結果になることを期待していたのでしょう。

```
0
1
3
4
```

しかし、このスクリプトの出力は以下のようになります。

```
2
```

これは、print()コールの返り値が *int(1)* であり、前記のオプションの数値引数のように 見えてしまうためです。

---

switch

*switch*文は、同じ式を用いてIF文を並べたのに似ています。 同じ変数を異なる値と比較し、値に応じて異なったコードを実行したいと 思うことがしばしばあるかと思います。 *switch*文は、まさにこのためにあるのです。

> **注意**: 他の言語とは違って、 continue命令は switchにも適用され、*break*と同じ動作をします。 ループの内部でswitchを使用しており、外側のループの処理を続行させたい場合には、*continue 2* を使用してください。

> **注意**: switch/case が行うのは、 緩やかな比較 であることに注意しましょう。

次の二つの例は、同じことを二つの異なった方法で書いたものです。 一つは、*if*文を、もう一つは*switch* 文を使っています。

Example#1 *switch*構造

```
<?php
if ($i == 0) {
    echo "iは0に等しい";
}
if ($i == 1) {
    echo "iは1に等しい";
}
if ($i == 2) {
    echo "iは2に等しい";
}

switch ($i) {
     case 0:
         echo "iは0に等しい";
          break;
     case 1:
         echo "iは1に等しい";
          break;
     case 2:
         echo "iは2に等しい";
          break;
}
?>
```

Example#2 *switch*構造では文字列を使用できる

```
<?php
switch ($i) {
case "apple":
     echo "i is apple";
     break;
case "bar":
     echo "i is bar";
     break;
case "cake":
     echo "i is cake";
     break;
}
?>
```

失敗を避けるために*switch*文がどのように実行されるのかを 理解することが重要です。 *switch*文は、行毎に実行されます。 (実際には、文毎に実行されます。)初めは、何も実行しません。 *switch*式の値と一致する値を有する *case*文が見つけられたときに初めてPHPにより 命令の実行が行われます。 PHPは*switch*ブロックの終わりまたは最初の *break*文まで実行を続けます。 CASE文の終わりに*break*文を書かない場合は、PHPは 次のCASE文を実行しつづけます。例えば、

```
<?php
switch ($i) {
case 0:
     echo "iは0に等しい";
case 1:
     echo "iは1に等しい";
case 2:
     echo "iは2に等しい";
}
?>
```

ここで、*$i*が0に等しい場合は、PHPは全ての echo 文を出力してしまいます! *$i*が1の場合、PHPは最後の二つの echo 文を出力します。 *$i*が2に等しい場合のみ、'期待した'動作をし、 'iは2に等しい'と表示します。 このため (ある種の状況下では、BREAKを付加することを避けたいと 思うかもしれませんが)、 *break*文を忘れないようにすることが重要です。

*switch*文では、条件は1度だけ評価され、 その結果が各*case*文と比較されます。 *elseif*文では、条件は、再度評価されます。 使用する条件が単純な比較処理よりも複雑な処理を行ったり、 重い繰り返し処理を行う場合、*switch*の方が より処理が速い可能性があります。

caseに付随する文は、空とすることが可能です。 この場合、次のcaseに付随する文に制御が移行します。

```
<?php
switch ($i) {
case 0:
case 1:
case 2:
   echo "iは3より小さいですが負ではありません";
    break;
case 3:
    echo "iは3です";
}
?>
```

default は、case 文の特別な場合です。これは他の全ての case にマッチしない場合に実行されます。例を以下に示します。

```
<?php
switch ($i) {
case 0:
    echo "iは0に等しい";
    break;
case 1:
    echo "iは1に等しい";
    break;
case 2:
    echo "iは2に等しい";
    break;
default:
    echo "iは0,1,2に等しくない";
}
?>
```

*case*式は、スカラー型に式を評価する 任意の式、つまり、整数、浮動小数点、文字列とすることができます。 配列又はオブジェクトは、単純な型にキャストされていない限り 使用することができません。

switch文の制御構造に関する別の構文がサポートされています。 詳細は、 制御構造に関する別の構文を参照ください。

```
<?php
switch ($i):
case 0:
    echo "iは0に等しい";
    break;
case 1:
    echo "iは1に等しい";
    break;
case 2:
    echo "iは2に等しい";
    break;
default:
    echo "iは0でも1でも2でもない";
endswitch;
?>
```

---

declare

declare 命令は、あるコードブロックの中に 実効命令をセットするために使用されます。*declare* の文法は他の制御構造と似ています。

```
declare （命令）
 文
```

*命令*の箇所で、セットされた *declare*ブロックの挙動を指定することが出来ます。 現在のところ、使用できる命令は*ticks*の 一つだけです。(ticksに関しては以下を参照してください)

*declare*ブロックの文 の実行のされ方や実行時にどのような作用が起こるかについては *命令*に何が指定されたかに依存します。

*declare*構造はグローバルスコープしても使用され、 それはそれ以降のコード上の全てにおいて影響します。

```
<?php
// 以下は同じです:

// you can use this:
declare(ticks=1) {
    // ここに全てのスクリプトを書く
}

// or you can use this:
declare(ticks=1);
    // ここに全てのスクリプトを書く
?>
```

ticks

tickとは*declare*ブロックの実行中にパーサが *N*個の低レベル命令を実行するごとに 発生するイベントのことです。*N*の値は *declare*ブロックの命令の箇所で *ticks=N*のように 指定します。

tickごとに発生させるイベントはregister_tick_function() を使用して指定します。詳細は以下の例を参照してください。1回のtickで 複数のイベントが起こり得ることに注意してください。

Example#1 PHPのコードの一部をプロファイルする

```
<pre>
<?php
// 呼び出されるとその時間を記録する関数
function profile ($dump = FALSE)
{
    static $profile;

    // 格納されたプロファイルを返し、その値を削除する
    if ($dump) {
        $temp = $profile;
        unset($profile);
        return $temp;
    }

    $profile[] = microtime();
}

// tickハンドラの設定
register_tick_function("profile");

// declareブロックの前で初期化しておく
profile ();

// 2命令ごとにtickを投げるように設定しブロックを実行する
declare (ticks=2) {
    for ($x = 1; $x < 50; ++$x) {
        echo similar_text (md5($x), md5($x*$x)), "<br />";
    }
}

// プロファイラに格納されたデータを表示する
print_r (profile(TRUE));
?>
</pre>
```

この例では、'declare'ブロックのコード中で2個の低レベル 命令が実行されるごとにその時間を記録してプロファイルを 行っています。この情報はコードのあるセクションの中で 遅い箇所を探すのに使用できます。この処理は他の手法でも 使用できます。ticksを使用することで実装が簡単になる 場合が多々あるのです。

ticksはデバッグ、単純なマルチタスク処理、バックグラウンドI/Oや その他多くの処理を実装するのに便利です。

register_tick_function()と unregister_tick_function()も参照してください。

---

return

関数内で呼び出されると、return()文は即座に その関数の実行を停止し、引数を関数の値として返します。 return()はまた、eval()文や スクリプト自体の実行を終了させることが出来ます。

グローバルスコープで呼び出されると、現在実行中のスクリプトが終了 します。もしそのスクリプトがinclude()もしくは require()されたものである場合、制御は呼び出し元 のファイルに戻ります。また、そのスクリプトがinclude() されたものである場合は、return()に与えられた引数 の値はinclude()の戻り値となります。 return()がメインスクリプトで呼び出された場合は スクリプトが終了します。また、 設定ファイルの auto_prepend_file 又は auto_append_fileオプションで指定されたスクリプトの場合も 同様にそのスクリプトが終了します。

さらに詳しい情報に関しては 返り値を参照してください。

> **注意**: return() は関数ではなく言語構造であるため、 引数を括弧で囲う必要があるのはないことに注意しましょう。 括弧で囲わずそのままにしておくのが一般的です。 またそのほうが PHP にかかる負荷も低くなります。
>
> **注意**: 変数をリファレンスで返す場合は、*決して*引数を 括弧で囲うべきではありません。そのようにすると正しく動作しません。 *return ($a);* とすると、変数を返すのではなく *($a)* を評価した結果を返すことになります （この場合は、もちろん *$a* の値です）。

---

require()

require()文は指定されたファイルを読み込み、評価します。 ファイルが読み込まれ評価される際の詳細な情報に関しては include()に記述されています。

require() と include() は、エラーの扱い方を除けば全く同様に振舞います。エラーが発生するとどちらも Warning を出力しますが、require() を使用している場合は Fatal Error となります。 言い換えると、指定されたファイルが無い場合に処理を停止したい場合は require() を使用した方が良い、とい

うことになります。 include() を使用すると、 読み込むべきファイルが存在しない場合も処理が続行されます。 include_path を適切に設定することも忘れないでください。

Example#1 基本的なrequire()の例

```
<?php

require 'prepend.php';

require $somefile;

require ('somefile.txt');

?>
```

include()のドキュメントにはさらに多くの例がありますので 参照してください。

> **注意**: PHP 4.0.2以前での挙動は以下の様になっています: require()は その行が実行される/されないにかかわらず常に指定されたファイルを読み込もうと します。従って条件文はrequire()には影響を与えません。 しかしながら、require()がある行が実行されない場合、 読み込まれるファイル内のコードは実行されません。同様に、ループ構造は require()の動作には影響しません。読み込まれるファイル内の コードがループに依存している場合でも require()は 読み込みを一回しか行いません。
>
> **注意**: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

警告

PHP 4.3.0 より前のバージョンの Windows 版 *PHP* は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

include(), require_once(), include_once(), get_included_files(), eval(), file(), readfile(), virtual() および include_path も参照ください。

---

include()

include()文は指定されたファイルを読み込み、評価します。

以下の記述内容はrequire()にも当てはまります。これら2つの構文は、 エラーの扱い方を除けば全く同様に振舞います。エラーが発生するとどちらも Warning を出力しますが、 require() を使用している場合は Fatal Error となります。 言い換えると、指定されたファイルが無い場合に処理を停止したい場合は require() を使用した方が良い、ということになります。 include() を使用すると、 読み込むべきファイルが存在しない場合も処理が続行されます。 include_path を適切に設定することも忘れないでください。 PHP 4.3.5 より前のバージョンでは include されたファイルにパースエラーがあっても処理が続行されましたが、それ以降のバージョンでは処理を停止します。

読み込むファイルはまずカレントのワーキングディレクトリからの相対パスとしてinclude_path で探され、それから、カレントのスクリプトのディレクトリからの相対パスとしてinclude_path で探されます。 例えば、include_pathが *libraries*で、カレントの ワーキングディレクトリが */www/* の 場合、*include/a.php* を読み込んで、そのファイルの中に *include "b.php"* と書いてあったとすると、 *b.php* がまず */www/libraries*で探され、その後、*/www/include/*で探されます。 ファイル名が *./* あるいは *../* で始まっている場合は、 カレントのワーキングディレクトリからの相対パスとして探されるのみとなります。

ファイルが読み込まれるとそのファイルに含まれるコードは、 include()もしくはrequire()が実行された 行の変数スコープを継承します。 呼び出し側の行で利用可能である全ての変数は、読み込まれたファイル内で利用可能です。 しかし、読み込まれたファイル内で定義されている関数やクラスはすべてグローバルスコープとなります。

Example#1 基本的なinclude()の例

```
vars.php
<?php

$color = 'green';
$fruit = 'apple';

?>

test.php
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

呼び出し側のファイルの関数定義の中で読み込みが行われた場合は、 読み込まれるファイルに含まれる全てのコードは、 その関数内で定義されているものとして動作します。 従って変数のスコープもその関数のものが継承されます。 ただ マジック定数 は例外で、これは読み込みを行う前にパーサが評価します。

Example#2 関数内での読み込み

```
<?php

function foo()
{
global $color;

     include 'vars.php';

     echo "A $color $fruit";
}

/* vars.php は foo() のスコープを継承するため *
 * $fruit はこの関数の外では無効となります。  *
 * $color はglobalとして宣言されているため    *
 * 有効です。                                 */

foo();                          // A green apple
echo "A $color $fruit";         // A green

?>
```

ファイルが読み込まれるときには、読み込まれるファイルの先頭で PHPモードを抜けてHTMLモードになり、最後に再びPHPモードに戻ります。 このため、読み込むファイル中のPHPコードとして実行する必要がある コードは、 有効なPHPの開始タグおよび終了タグで括る必要があります。

"URL fopenラッパー"が 有効になっている場合(デフォルト設定では有効です)、ローカルなパス名 の代わりにURL(HTTP経由)を用いて読み込むファイルを指定することが可能です。 URLで指定されたサーバがファイルをPHPコードとして解釈することが 出来る場合には、HTTP GETを使用してURLリクエストに引数を指定することが 出来ます。これはファイルの読み込み云々やスコープの継承とは関係なく、 ただ単純にスクリプトがリモートのサーバで実行されて結果がローカルの スクリプトに読み込まれる、というだけのことです。

警告

PHP 4.3.0 より前のバージョンの Windows 版 *PHP* は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

Example#3 HTTP経由のinclude()

```
<?php

/* この例は www.example.com が.phpはPHPスクリプトとして扱い、.txtは通常の *
 * ファイルとして扱うように設定されていると仮定しています。また、ここでの *
 * '動作します'という言葉の意味は、変数$fooと$barが読み込まれる側のファイ *
 * ルで使用可能である、ということです。                                    */

// 動作しません: www.example.com では file.txt はPHPコードとして解釈されません。
include 'http://www.example.com/file.txt?foo=1&bar=2';

// 動作しません: 'file.php?foo=1&bar=2' という名前のファイルをローカルファイル
// システム上から探し出そうとします。
include 'file.php?foo=1&bar=2';

// 動作します。
include 'http://www.example.com/file.php?foo=1&bar=2';

$foo = 1;
$bar = 2;
include 'file.txt';  // 動作する
include 'file.php';  // 動作する

?>
```

警告

セキュリティの警告

リモートファイルはリモートサーバ上で実行されます（ファイルの拡張子や リモートサーバが PHP の実行を許可しているかどうかに依存します）が、有効な PHP スクリプトである必要があります。なぜならそれはローカル サーバ上で処理されるからです。もしリモートサーバ上で処理された結果が ほしいだけならば、readfile() を使用するほうがよい でしょう。そうでなければ、リモートスクリプトが有効な期待通りのコードを 生成していることを確認するため、注意を払うことが必要になります。

リモートファイル, fopen(), file()も参照してください。

値の返し方: 読み込まれたファイル内では、ファイルの実行処理を終了し呼出側の スクリプトに戻るためにreturn()文を実行することが可能です。 読み込まれたファイルから値を返すことも可能です。 通常の関数で行うのと同様にincludeコールの値を取得することができます。 しかし、読み込まれたリモートファイル(ローカルファイルの場合も同様)の出力が、 有効なPHPの開始／ 終了タグを有していない限り、リモートファイルを読み込む際に値を取得することはできません。 必要な変数をこれらのタグの中で宣言することができ、これらの変数は ファイルが読み込まれた位置で使用可能となります。

include() は特別な言語構造であるため、 引数の前後に括弧は不要です。 返り値を比較する際には注意してください。

Example#4 インクルードの戻り値を比較する

```
<?php
// won't work, evaluated as include(('vars.php') == 'OK'), i.e. include('')
if (include('vars.php') == 'OK') {
     echo 'OK';
```

```
}

// works
if ((include 'vars.php') == 'OK') {
    echo 'OK';
}
?>
```

> **注意**: PHP 3では関数ブロック以外のブロック内でreturnを 使用することはできません。 return()が適用されるのは関数であり、ファイル全体ではありません。

Example#5 include()とreturn()文

```
return.php
<?php

$var = 'PHP';

return $var;

?>

noreturn.php
<?php

$var = 'PHP';

?>

testreturns.php
<?php

$foo = include 'return.php';

echo $foo; // 'PHP'と出力されます

$bar = include 'noreturn.php';

echo $bar; // 1が出力されます

?>
```

読み込みが成功すると*$bar*の値は1となります。上の2つの例の違いに 注目してください。最初の例では読み込まれるファイル側でreturn() を使用し、もう一方では使用していません。 ファイルが読み込めなかった場合、**FALSE** が返され、 *E_WARNING* が発生します。

読み込まれるファイルで定義された関数がある場合、 これらは、return()の前後によらず メインファイルで使用できます。 このファイルが二度読み込まれた場合、PHP 5は関数が定義済みであるため 致命的なエラーを発生します。一方、PHP 4は return()の後に定義された関数については、 エラーを発生しません。 ファイルが読み込み済みであるかどうかを調べ、 読み込まれるファイルの内容を条件分岐で返すかわりに include_once()を使用することを推奨します。

PHP ファイルの内容を変数に "include する" もうひとつの方法は、 出力制御関数 を include() とともに用いて 出力をキャプチャすることです。たとえば、

Example#6 出力バッファリングを用い、 PHP ファイルの内容を文字列として読み込む

```
<?php
$string = get_include_contents('somefile.php');

function get_include_contents($filename) {
    if (is_file($filename)) {
        ob_start();
        include $filename;
        $contents = ob_get_contents();
        ob_end_clean();
        return $contents;
    }
    return false;
}

?>
```

スクリプト中で自動的にファイルをインクルードするには、*php.ini* の auto_prepend_file および auto_append_file オプションも参照ください。

> **注意**: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

require(), require_once(), include_once(), get_included_files(), readfile(), virtual() および include_path も参照ください。

---

require_once()

require_once()文は、スクリプトの実行中に指定された ファイルを読み込み、評価します。この振る舞いはrequire() 文と似ていて、唯一の違いは既にコードが読み込まれている場合には再度読み込まれる ことがないことです。この文がどのように動作するかについては require()を参照してください。

require_once()は、スクリプトの実行時に同じファイ ルが複数回読み込まれ、評価される可能性がある場合に、関数の再定義や 変数値の再代入といった問題を回避するために一回だけ読み込ませるため に使用します。

require_once()および include_once()の使用例に関する他の例については、 最新のPHPソースコード配布ファイルに含まれる » PEARのコードを参照ください。

戻り値は、include()と同じです。 ファイルが既にインクルードされている場合、この関数は **TRUE**を返します。

> **注意**: require_once()はPHP 4.0.1で追加されました。
>
> **注意**: (Windowsのように)大文字小文字を区別しないオペレーティングシステムでは、 require_once()および include_once()の動作が意図したものにならない 可能性があるので注意してください。
>
> **Example#1 require_once() はWindowsでは大文字小文字を区別しません**
>
> ```
> require_once "a.php"; // a.phpを読み込みます
> require_once "A.php"; // Windowsはこれもa.phpを読み込みます! (PHP 4のみ)
> ```
> この動作は、PHP 5で変更されました。パスはまず正規化されます。 このため、 *C:¥PROGRA~1¥A.php*は *C:¥Program Files¥a.php*と同じと認識され、 ファイルは一回だけ読み込まれます。

警告

PHP 4.3.0 より前のバージョンの Windows 版 *PHP* は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

require(), include(), include_once(), get_required_files(), get_included_files(), readfile(), virtual()も参照してください。

---

include_once()

include_once()命令は、スクリプトの実行時に指定 したファイルを読み込み評価します。この動作は、 include()命令と似ていますが、ファイルからのコー ドが既に読み込まれている場合は、再度読み込まれないという重要な違い があります。その名が示す通り、ファイルは一度しか読み込まれません。

include_once()は、スクリプトの実行時に同じファイ ルが複数回読み込まれ、評価される可能性がある場合に、関数の再定義や 変数値の再代入といった問題を回避するために一回だけ読み込ませるため に使用します。

require_once()および include_once()の使用例に関する他の例については、 最新のPHPソースコード配布ファイルに含まれる » PEARのコードを参照ください。

返り値は include() と同じです。 ファイルが既に読み込まれている場合は、この関数は **TRUE** を返します。

include_once() は、PHP 4.0.1で追加されました。

> **注意**: (Windowsのように)大文字小文字を区別しないオペレーティングシステムでは、 include_once()および require_once()の動作が意図したものにならない 可能性があるので注意してください。
>
> **Example#1 include_once() はWindowsでは大文字小文字を区別しません**
>
> ```
> include_once "a.php"; // a.phpを読み込みます
> include_once "A.php"; // Windowsはこれもa.phpを読み込みます! (PHP 4 のみお)
> ```

警告

PHP 4.3.0 より前のバージョンの Windows 版 *PHP* は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

include()、require()、 require_once()、 get_required_files()、 get_included_files()、 readfile()、virtual() も参照してください。

関数

目次

ユーザー定義関数

関数は次のような構文で定義されます。

Example#1 関数の使用法を説明するための擬似コード

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "関数の例\n";
    return $retval;
}
?>
```

関数の中では、他の関数や クラス 定義 を含む PHP のあらゆる有効なコードを使用することができます。

関数名は、PHP の他のラベルと同じ規則に従います。関数名として有効な 形式は、まず文字かアンダースコアで始まり、その後に任意の数の文字・ 数字・あるいはアンダースコアが続くものです。正規表現で表すと、 *[a-zA-Z_¥x7f-¥xff][a-zA-Z0-9_¥x7f-¥xff]** となります。

ヒント

ユーザレベルでの命名の手引き もご覧になるとよいでしょう。

PHP 3 では、関数は参照される前に定義されている必要がありました。 PHP 4 以降ではそのような制限はありません。 ただし以下の二つの例のように、条件付きで関数が 定義されるような場合を*除きます*。

次の二つの例のように、ある条件下でのみ関数が定義される場合には、 その関数定義は関数がコールされる*前に* 行われていなければなりません。

Example#2 条件つきの関数

```
<?php

$makefoo = true;

/* ここでは関数foo()はまだ定義されていないので
   コールすることはできません。
   しかし関数 bar()はコールできます。 */

bar();

if ($makefoo) {
  function foo()
  {
    echo "I don't exist until program execution reaches me.¥n";
  }
}

/* ここでは $makefooがtrueと評価されているため
   安全にfoo()をコールすることができます。 */

if ($makefoo) foo();

function bar()
{
  echo "I exist immediately upon program start.¥n";
}

?>
```

Example#3 関数の中の関数

```
<?php
function foo()
{
  function bar()
  {
    echo "I don't exist until foo() is called.¥n";
  }
}

/* ここでは関数bar()はまだ定義されていないので
   コールすることはできません。 */

foo();

/* foo()の実行によって bar()が
   定義されるためここではbar()を
   コールすることができます。*/

bar();

?>
```

PHP では、関数やクラスはすべてグローバルスコープにあります - 関数の内部で定義したものであっても関数の外部からコールできますし、 その逆も可能です。

PHP は関数のオーバーロードをサポートしていません。 また、宣言された関数の定義を取り消したり再定義することも できません。

> **注意**: 関数名は大文字小文字を区別しませんが、通常は 関数宣言時と同じ名前で関数をコールする方が好ましいです。

PHP 3 では、引数のデフォルト値(詳細は、引数のデフォルト値を 参照ください)はサポートしていますが、関数の引数を可変とすることはで きません。PHP 4 以降は両方ともサポートしています。詳細は、 可変長引数リスト および func_num_args(), func_get_arg(), func_get_args() に関する関

数リファレンスを 参照ください。

PHP では、関数を再帰的にコールすることが可能です。ただし、100 から 200 を超えるような再帰呼び出しは避けてください。そんなことをすると、 スタックが破壊され、スクリプトが異常終了してしまいます。

Example#4 再帰的な関数

```
<?php
function recursion($a)
{
    if ($a < 20) {
        echo "$a¥n";
        recursion($a + 1);
    }
}
?>
```

関数の引数

引数のリストにより関数へ情報を渡すことができます。 このリストは、カンマで区切られた式のリストです。

PHP は、値渡し(デフォルト)、 参照渡し、 デフォルト引数値 をサポートしています。可変長引数リストは、PHP 4以降でのみサポート されています。詳細は、 可変長引数リスト および func_num_args(), func_get_arg(), func_get_args() に関する関数リファレンスを 参照ください。PHP 3でも関数に引数の配列を渡すことにより 同様の効果を得ることができます。

Example#1 Passing arrays to functions

```
<?php
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
?>
```

参照渡しで引数を作成する

デフォルトで、関数の引数は値で渡されます。(このため、関数の内部で 引数の値を変更しても関数の外側では値は変化しません。)関数がその引 数を修正できるようにするには、その引数を参照渡しとする必要があり ます。

関数の引数を常に参照渡しとしたい場合には、関数定義において アンパサンド(&) を引数名の前に付加することができます。

Example#2 Passing function parameters by reference

```
<?php
function add_some_extra(&$string)
{
    $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // outputs 'This is a string, and something extra.'
?>
```

デフォルト引数値

関数は、スカラー引数に関して次のように C++ スタイルのデフォルト値を 定義することができます。

Example#3 関数におけるデフォルトパラメータの使用法

```
<?php
function makecoffee($type = "cappuccino")
{
    return "Making a cup of $type.¥n";
}
echo makecoffee();
echo makecoffee(null);
echo makecoffee("espresso");
?>
```

上のコードにより、次のような出力が行われます。

```
Making a cup of cappuccino.
Making a cup of .
Making a cup of espresso.
```

PHPでは、配列および特殊な型NULLをデフォルト値とすることも可能です。 例えば、

Example#4 スカラー型以外をデフォルト値として使用する

```
<?php
function makecoffee($types = array("cappuccino"), $coffeeMaker = NULL)
{
    $device = is_null($coffeeMaker) ? "hands" : $coffeeMaker;
```

```
    return "Making a cup of ".join(", ", $types)." with $device.¥n";
}
echo makecoffee();
echo makecoffee(array("cappuccino", "lavazza"), "teapot");
?>
```

デフォルト値は、定数式である必要があり、 (例えば) 変数やクラスのメンバーであってはなりません。

引数のデフォルト値を使用する際には、デフォルト値を有する引数はデ フォルト値がない引数の右側に全てある必要があることに注意して下さ い。そうでない場合、意図したような動作が行われません。次の簡単な コードを見てみましょう。

Example#5 関数の引数のデフォルト値の 間違った使用法

```
<?php
function makeyogurt($type = "acidophilus", $flavour)
{
    return "Making a bowl of $type $flavour.¥n";
}

echo makeyogurt("raspberry");   //  期待通りには動作しません。
?>
```

上記の例の出力は次のようになります。

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry .
```

ここで、上の例を次のコードと比べてみましょう。

Example#6 関数の引数のデフォルト値の 正しい使用法

```
<?php
function makeyogurt($flavour, $type = "acidophilus")
{
    return "Making a bowl of $type $flavour.¥n";
}

echo makeyogurt("raspberry");   //  期待通り動作します
?>
```

この例の出力は、次のようになります。

```
Making a bowl of acidophilus raspberry.
```

> **注意**: PHP 5以降、デフォルトで値はリファレンス渡しとなります。

可変長引数リスト

PHP 4 以降は、可変長引数をユーザー定義関数でサポートしています。 可変長引数の使用法は非常に簡単で、 func_num_args(), func_get_arg(), func_get_args() 関数を使用します。

可変長引数に関して特別な構文は必要としません。 引数リストは従来と同様に関数定義で明示的に指定することができ、 動作も従来と変わりません。

返り値

オプションの return 文により値を返すことができます。 リストやオブジェクトを含むあらゆる型を返すことができます。 これにより、関数の実行を任意の箇所で終了し、その関数を呼び出した 箇所に制御を戻すことが出来ます。詳細に関しては return()を参照してください。

Example#1 return()の使用法

```
<?php
function square($num)
{
    return $num * $num;
}
echo square(4);   //   '16'を出力
?>
```

複数の値を返すことはできませんが、リストを返すことにより 同じ効果を得ることができます。

Example#2 複数の値を得るために配列を返す

```
<?php
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
?>
```

関数からリファレンスを返すには、リファレンス演算子 & を関数宣 言部および変数への返り値を代入する際の両方で使用する必要があります。

Example#3 関数からリファレンスを返す

```
<?php
function &returns_reference()
{
    return $someref;
}

$newref =& returns_reference();
?>
```

リファレンスに関するさらに詳しい情報がリファレンスの説明にあります。

可変関数

PHP は可変関数(variable functions)の概念をサポートします。 これにより、変数名の後に括弧が付いている場合、その値が何であろうと PHPは、同名の関数を探し実行を試みます。 この機能は、コールバック、関数テーブル等を実装するために使用可能です。

可変関数は、echo(), unset(), isset(), empty(), include(), print() のような言語構造と組み合わせて使用する ことはできません。これらの言語構造を可変変数として使うには 独自のラッパ関数を使う必要があります。

Example#1 可変関数の例

```
<?php
function foo()
{
    echo "In foo()<br />¥n";
}

function bar($arg = '')
{
    echo "In bar(); argument was '$arg'.<br />¥n";
}

// This is a wrapper function around echo
function echoit($string)
{
    echo $string;
}

$func = 'foo';
$func();        // This calls foo()

$func = 'bar';
$func('test');  // This calls bar()

$func = 'echoit';
$func('test');  // This calls echoit()
?>
```

オブジェクトのメソッドを可変関数を使ってコールすることもできます。

Example#2 可変メソッドの例

```
<?php
class Foo
{
    function Variable()
    {
        $name = 'Bar';
        $this->$name(); // Bar() メソッドのコール
    }

    function Bar()
    {
        echo "This is Bar";
    }
}

$foo = new Foo();
$funcname = "Variable";
$foo->$funcname();  // $foo->Variable() をコールする

?>
```

可変変数や function_exists()も参照してください。

内部（ビルトイン）関数

PHPは標準で多くの関数と言語構造を持っています。また他にも コンパイル済みの特定のPHPエクステンションを必要とする関数があります。 それらはもしコンパイルされていなければ"undefined function（未定義の関数）" として致命的エラーを発するでしょう。例えば、 imagecreatetruecolor() のような 画像関数を使用するには、 GDサポートを有効にしてPHPをコンパイルしておく必要があります。 また、mysql_connect()を使う場合もやはり MySQLサポートを有効にしてPHPが コンパイルされている必要があります。 stringや variable関数のように どのバージョンのPHPでも含まれているコアの関数もたくさんあります。 phpinfo()やget_loaded_extensions()を コールすることで使用しているPHPにロードされているエクステンション

を 見ることができます。また、多くのエクステンションはデフォルトで有効に なっており、PHPのマニュアルはエクステンション毎に分けられている ことにも 注意してください。PHPのセットアップについては 設定, インストール,そして個々のエクステンション の項をご覧ください。

関数のプロトタイプに関する解説はマニュアルの 関数の定義を読むにはを 参照してください。関数の戻り値や引数が直接与えられた場合に どのように動くかを認識することは重要です。 例えば、str_replace()は変更された文字列を 返すのに対し、usort()は与えられた引数そのものを 変更します。マニュアルの各項にはそれぞれの関数に関する情報があります。 関数の引数、振る舞いの変更、成功した場合失敗した場合の それぞれの戻り値、可用性に関する情報などです。 これらの重要な（時には微妙な）違いを知ることは、 正しいPHPコードを書くうえで極めて重要なことです。

> **注意**: 関数へのパラメータとして関数が想定しているのとは異なるものを渡した場合、 例えば文字列を想定しているところに配列を渡した場合などの場合は 関数の返り値は未定義となります。たいていの場合は **NULL** を返すでしょう。しかしこれはあくまでも規約にすぎず、これに依存することはできません。

function_exists(), the function reference, get_extension_funcs(), dl()も参照してください。

クラスとオブジェクト (PHP 4)

目次

クラス

クラスは、変数およびこれらの変数で動作する関数の集まりです。 変数は *var* で、そして関数は *function* で定義します。 クラスは次のような構文により定義されます。

```
<?php
class Cart
{
    var $items;  // 買い物カゴの中のアイテム

    // $num 個の $artnr を買い物カゴに加えます

    function add_item ($artnr, $num)
    {
        $this->items[$artnr] += $num;
    }

    // $num 個の $artnr を買い物カゴから出します

    function remove_item ($artnr, $num)
    {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } elseif ($this->items[$artnr] == $num) {
            unset($this->items[$artnr]);
            return true;
        } else {
            return false;
        }
    }
}
?>
```

この例は、買い物カゴの中の物の連想配列と、買い物カゴにアイテムを加えたり 除いたりする二つの関数からなる Cart という名前のクラスを 定義します。

警告

複数のファイルで分割してクラスを定義することは *できません*。また、関数定義の内部分割されている 場合を除き、複数の PHP ブロックで分割してクラスを定義することも *できません*。次の例は正しく動作しません:

```
<?php
class test {
?>
<?php
    function test() {
        print 'OK';
    }
```

```
}
?>
```

しかし、次の例は許されます:

```
<?php
class test {
     function test() {
         ?>
         <?php
         print 'OK';
     }
}
?>
```

以下の注意書きはPHP 4に関するものです。

警告

名前*stdClass*は、Zendにより内部的に使用され、保 存されています。PHPで*stdClass*という名前のクラ スを使用することはできません。

警告

関数*__sleep*および*__wakeup*は、 PHPクラス内で使用される特殊な関数です。これらの関数に付随する特殊 な機能を使用する場合以外は、クラス内でこれらの名前を付けた関数を 使用することはできません。詳細な情報については、以下を参照ください。

警告

PHP は、__で始まる全ての関数名を特殊な関数として予約しています。 ドキュメントに記載された特殊関数の機能を使用する場合以外は、__を 有する関数名を使用しないことを推奨します。

PHP 4では、変数*var*については定数による初期化の みが可能です。定数以外で初期化を行う場合には初期化関数が必要です。 この初期化関数は、オブジェクトがクラスから構築される際に自動的に コールされます。このような関数はコンストラクタと呼ばれます。(以下 参照)

```
<?php
class Cart
{
    /* 以下のコードはPHP 4では動作しません。 */
    var $todays_date = date("Y-m-d");
    var $name = $firstname;
    var $owner = 'Fred ' . 'Jones';
    /* Arrays containing constant values will, though. */
    var $items = array("VCR", "TV");
}

/* 以下に正しい方法を示します。 */
class Cart
{
    var $todays_date;
    var $name;
    var $owner;
    var $items = array("VCR", "TV");

    function Cart()
    {
        $this->todays_date = date("Y-m-d");
        $this->name = $GLOBALS['firstname'];
        /* 等など. . . */
    }
}
?>
```

クラスは型、つまり、実際の変数の雛型です。*new*演 算子により所望の型の変数を作成する必要があります。

```
<?php
$cart = new Cart;
$cart->add_item("10", 1);

$another_cart = new Cart;
$another_cart->add_item("0815", 3);
?>
```

この例は、クラスCartのオブジェクト*$cart*および *$another_cart*を作成します。 *$cart*オブジェクトの関数 add_item()が商品番号10の商品一つがカートに追加されています。商品番号0815の商品3つが*$another_cart* に追加されています。

*$cart*と*$another_cart*は共に関数 add_item()、 remove_item() と変数items を有しています。これらは、異なる関数および変数です。 オブジェクトは、ファイルシステムのディレクトリに似たようなものであると 考えられます。ファイルシステムでは、別のディレクトリに置く限り、 二つの異なる*README.TXT*を保持することが可能です。 ディレクトリではトップディレクトリから各ファイルにアクセスするには、 フルパス名を入力する必要がありますが、これと全く同様にコールしたい 関数の完全な名前を指定する必要があります。PHPの用語では、最上位の ディレクトリはグローバル名前空間であり、パス名のセパレータは、 -> となります。つまり、*$cart->items* と *$another_cart->items* は、二つの異なる変数です。
*$cart->items*という名前の変数は、 *$cart->$items*ではない、つまり、PHP の変数名は ドル記号を一つだけ有することに注意してください。

```
// 正しい、$は一つ
$cart->items   = array("10" => 1);

// $cart->$items は、$cart->"" になるため、正しくない。
$cart->$items = array("10" => 1);
```

```
// 正しいが、意図しているかどうかによらず、
// $cart->$myvar は、 $cart->items となる
$myvar = 'items';
$cart->$myvar = array("10" => 1);
```

クラス定義の内部では、プログラムでアクセス可能なオブジェクト名を知ることはできません。 Cart クラスが書かれている時点では、そのオブジェクトの名前が後で *$cart* になるのか *$another_cart* になるのか、 それとも別のものになるのかはわかりません。 つまり、Cart クラスの中では *$cart->items* と書くことはできないのです。代わりに、 クラスの中からそのクラス内の関数や変数にアクセスするために、 疑似変数 *$this* を使用することが可能です。 *$this* は、「自分自身の」または「カレントのオブジェクト」 と読み変えることができます。つまり、 '*$this->items[$artnr]* += *$num*' は、 「同じクラス内の配列 items の *$artnr* カウンタに *$num* を追加する」または「カレントオブジェクト内の配列 items の *$artnr* カウンタに *$num* を追加する」と読み変えることが可能です。

> **注意**: 擬似変数 *$this* は、メソッドが静的にコールされた場合に 常に定義されるわけではありません。しかし、はっきりしていることは: もしメソッドが別のオブジェクトから静的にコールされている場合は *$this* が定義されるということです。 この場合、*$this* は呼び出し元のオブジェクトとなります。 これを説明したのが次の例です:
>
> ```
> <?php
> class A
> {
>     function foo()
>     {
>         if (isset($this)) {
>             echo '$this is defined (';
>             echo get_class($this);
>             echo ")¥n";
>         } else {
>             echo "¥$this is not defined.¥n";
>         }
>     }
> }
>
> class B
> {
>     function bar()
>     {
>         A::foo();
>     }
> }
>
> $a = new A();
> $a->foo();
> A::foo();
> $b = new B();
> $b->bar();
> B::bar();
> ?>
> ```
>
> 上の例の出力は以下となります。
>
> ```
> $this is defined (a)
> $this is not defined.
> $this is defined (b)
> $this is not defined.
> ```

> **注意**: クラスとオブジェクトを処理する優れた関数がいくつかあります。 クラス/オブジェクト関数 をざっとみてみると良いでしょう。

---

extends

他の既存のクラスに似た変数や関数を有するクラスが必要になることがよ くあります。実際、全てのプロジェクトで使用可能な一般的なクラスを定 義し、このクラスを特定のプロジェクトの各々の要求に合わせて調整する というのは、良いやり方です。これを簡単に行うにために、他のクラスを 拡張してクラス作成することが可能です。拡張あるいは派生クラスは、基 底クラスの全ての変数と関数を有します。(これは、実際には誰も亡くなっ ていませんが、'継承'と呼ばれます) この派生クラスには、派生クラスの 定義で追加したものも含まれます。クラスから定義を取り除く、つまり、 既存の関数や変数を未定義とすることはできません。派生クラスは、常に 単一の基底クラスに依存します。つまり、多重継承は、サポートされてい ません。クラスは、キーワード'extends'を用いて拡張されます。

```
<?php
class Named_Cart extends Cart
{
    var $owner;

    function set_owner ($name)
    {
        $this->owner = $name;
    }
}
?>
```

この例は、Cart の全ての変数及び関数に加えて変数*$owner*と 関数 set_owner() を保持するクラス Named_Cart を定義しています。 この定義により、名前付きのカゴを通常の手段で作成し、カゴの保有者を 設定したり得たりすることができます。 名前付きのカゴで元のカゴクラスの関数を使うことも可能です。

```
$ncart = new Named_Cart;     // 名前付きの籠を作成
$ncart->set_owner("kris");   // 籠の所有者の名前を設定
print $ncart->owner;         // 籠の所有者を出力
$ncart->add_item("10", 1);   // (籠から継承された機能)
```

"親と子"と呼ばれる関係もあります。ある親クラスを作成し、 この親クラスに基づく新しいクラス、つまり、子クラスを *extends*により作成します。この新しい子クラスを使用することやこの子クラスに基づき他の クラスを作成することが可能です。

> **注意**: クラスは、使用される前に定義されている必要があります! *Cart*を拡張した クラス *Named_Cart*を作成したい場合、 まず、*Cart*を定義する必要があります。 クラス*Named_Cart*に基づき *Yellow_named_cart*という名前の他のクラスを 作成する場合、まず *Named_Cart*を定義する 必要があります。要するに、クラスの定義の順序は、重要です。

コンストラクタ

コンストラクタは、*new*によりクラスの新しいインス タンスを作成する際に自動的にコールされるクラス関数です。 ある関数が、クラス名と同じ名前を有している場合にコンストラクタに なります。コンストラクタが存在しない場合、もし基底クラスの コンストラクタが存在すれば、それがコールされます。

```
<?php
// PHP 3 および PHP 4で動作します
class Auto_Cart extends Cart
{
     function Auto_Cart()
     {
         $this->add_item ("10", 1);
     }
}
?>
```

この例は、Cart にコンストラクタを加えたクラス Auto_Cart を定義しています。 このコンストラクタは、"new" により新しい Auto_Cart が作成される度に 籠に10番の物を一つ保持するように初期化します。コンストラクタは、オ プションとして引数をとります。これにより、コンストラクタは非常に便 利なものとなります。このクラスをパラメータが指定されない場合でも使 用できるようにするには、コンストラクタに指定する全てのパラメータに デフォルト値を指定してください。

```
<?php
class Constructor_Cart extends Cart
{
     function Constructor_Cart($item = "10", $num = 1)
     {
         $this->add_item ($item, $num);
     }
}

// しつこいが、前の例と同じものを買う
$default_cart = new Constructor_Cart;

// 実際に買うものをカゴに入れる...
$different_cart = new Constructor_Cart("20", 17);
?>
```

*@new*のようにコンストラクタで発生するエラーの 出力を抑制するために@演算子を使用することが 可能です。例：*@new*

```
<?php
class A
{
     function A()
     {
         echo "Aのコンストラクタです<br>\n";
     }

     function B()
     {
         echo "クラスAのBという名前の通常の関数<br>\n";
         echo "Aのコンストラクタではありません<br>\n";
     }
}

class B extends A
{
}

// これにより、B() がコンストラクタとしてコールされます。
$b = new B;
?>
```

クラスAの関数 B() は、意図されていない場合でも突然クラスB の コンストラクタになってしまいました。PHP 4 は、この関数が クラスBで定義されているかとかその関数が継承されているかどうかは 考慮しません。

警告

PHP 4 では派生クラスのコンストラクタから基底クラスの コンストラクタを自動的にコールすることはできません。 上流のコンストラクタを適切にコールするように伝播させることは あなたの責任でやるべきことです。

デストラクタは、unset()またはスコープから でることにより、オブジェクトが破棄される度に自動的にコールされる関数です。 PHPにはデストラクタはありません。デストラクタの機能の多くを シミュレーションするには、代わりに register_shutdown_function() を使用します。

::

警告

以下の記述は、PHP 4 以降でのみ有効です。

基底クラスの関数や変数を参照したり、まだ特定のインスタンスを持たない クラスの関数を参照したりできると便利であるような場合があります。 演算子 :: はこのような場合に使用されます。

```
<?php
class A
{
     function example()
     {
        echo "オリジナルの関数A::example().<br>\n";
     }
}

class B extends A
{
     function example()
     {
         echo "再定義された関数B::example().<br>\n";
         A::example();
     }
}

// クラスAのオブジェクトはありません
// この例は、「オリジナルの関数A::example().<br>」を出力しま
// す
A::example();

// クラスBのオブジェクトを作成
$b = new B;

// この例の出力は次のようになります。
//     再定義された関数B::example().<br>
//     オリジナルの関数A::example().<br>
$b->example();
?>
```

上の例では、関数example()がクラスAでコールされていますが、クラスA のオブジェクトはありません。このため、$a->example()のように書くことはできません。代わりに、example()を「クラス関数」、つまり、そのク ラスのオブジェクトではなく、クラス自体の関数としてとしてコールします。

クラス関数はありますが、クラス変数はありません。実際、コールをする 時点ではオブジェクトは存在しません。つまり、クラス関数はどのオブジェクト変数も使用することはできません。(しかし、ローカル変数やグロー バル変数は使用可能です。)また、*$this*を 使用することはできません。

上の例では、クラスBは関数example()を再定義しています。クラスAの元 の定義は隠され、::演算子を使用してクラスAのexample()の実装を明示的 に参照しない限り利用できなくなっています。これを行うには、 A::example()と書いてください。(実際には、次節で示すように parent::example()と書く必要があります。)

この状況では、カレントのオブジェクトがあり、オブジェクト変数を保持 することが可能です。つまり、オブジェクト関数の内部で使用された場合、*$this*およびオブジェクト変数を使用することが可能です。

親クラス

基底クラスの変数と関数を参照するコードを書くことが可能です。これは、 派生クラスが基底クラスのコードを特定の用途向けに改造したりする場合には、特に有用です。

コードの基底クラスのリテラル名を使用する代わりに、特別な名前 *parent*を使用する必要があります。この名前は、 クラスの*extends*宣言で指定された基底クラスの名前 を指しています。これにより、基底クラスの名前を複数の場所で使用する ことを避けることが可能です。実装の際に継承ツリーを変更した場合でも、 変更は簡単で、クラスの *extends* 宣言を変更するだけですみます。

```
<?php
class A
{
     function example()
     {
         echo "A::example()です。基本関数を提供します。<br>\n";
     }
}

class B extends A
{
     function example()
     {
         echo "B::example()です。付加的な機能を提供します。<br>\n";
         parent::example();
     }
}
```

```
$b = new B;

// B::example()をコールし、この関数の中からA::example()がコールされます。
$b->example();
?>
```

オブジェクトのシリアル化 - セッションでのオブジェクト

> **注意**: PHP 3では、オブジェクトはシリアル化、非シリアル化の課程でクラスの 相関を失ってしまいました。復元される変数は、オブジェクト型ですが、 クラスもメソッドもありません。このため、全く実用的ではありません でした。(変わった構文の配列のようになっていました。)

警告

以下の情報は、PHP 4 以降でのみ有効です。

serialize() は、PHPで保存可能な全ての値のバイト ストリーム表現を有する文字列を返します。 unserialize() は、この文字列を使用して元の変数 値を再生することが可能です。オブジェクトを保存するためにシリアル化 を行うと、オブジェクトの全ての変数が保存されます。オブジェクトの関 数は保存されません。クラス名だけが保存されます。

オブジェクトの unserialize() を可能とするために、 そのオブジェクトのクラスが定義される必要があります。つまり、クラス Aのオブジェクト*$a*を page1.phpで定義し、 これをシリアル化した場合、 クラスAを指す文字列が得られ、そこには、$aに含まれる変数の全ての値 が含まれます。page2.phpでこの文字列を非シリアル化したい場合、クラ スAの*$a*を再生します。クラスAの定義が、page2.phpに 現れます。これは、 例えば、クラスAのクラス定義をインクルードファイルの中に保存し、 page1.php および page2.php の中で共にこのファイルを読み込むことに より実行可能です。

```
<?php
// classa.inc:

  class A
  {
       var $one = 1;

       function show_one()
       {
            echo $this->one;
       }
  }

// page1.php:

  include("classa.inc");

  $a = new A;
  $s = serialize($a);
  // page2.phpが見付られる場所に$sを保存
  $fp = fopen("store", "w");
  fwrite($fp, $s);
  fclose($fp);

// page2.php:

  // これは非シリアル化が正しく動作するために必要
  include("classa.inc");

  $s = implode("", @file("store"));
  $a = unserialize($s);

  // オブジェクト$aの関数show_one()を使用する
  $a->show_one();
?>
```

セッションを使用している場合に、オブジェクトを登録するために session_register()を使用すると、これらのオブジェ クトは各PHPページの最後で自動的にシリアル化され、次のページで自動 的に非シリアル化されます。これは、基本的に、これらのオブジェクトが 一旦セッション変数となると、全てのページに現れることを意味します。

全てのページでこれらのクラスを実際には使用しない場合でも、全てのペー ジでこのような登録された全てのオブジェクトのクラス定義を読み込むことが強く推奨されます。これを行わずに、クラス定義が存在しない状態で オブジェクトが非シリアル化された場合、クラスの相関は失われ、全ての 関数が利用できなくなるため、クラスのオブジェクト *stdClass*は利用価値がかなり低くなります。

このため、上の例で、*session_register("a")*を実行 することにより $a がセッションの一部となった場合、page1.php および page2.phpだけでなく、全てのページでファイル *classa.inc*を読み込むべきです。

特殊関数*__sleep* および *__wakeup*

serialize() は、クラスに特殊な名前 *__sleep* の関数があるかどうかを調べます。 もしあれば、シリアル化の前にその関数を実行します。 この関数で、オブジェクトをクリアすることができます。 またこの関数は、シリアル化するオブジェクトについて、 すべての変数の名前を配列で返すことが前提となっています。 このメソッドが何も返さなかった場合は、**NULL** がシリアル化され、E_NOTICE が発生します。

典型的な *__sleep* の使用法は、 途中のデータをコミットしたり、 似たようなタスクのクリアを行うといったものです。 また、オブジェクトが非常に大きく、かつ、完全に保存する必要がない場合、 この関数が有用です。

逆に、 unserialize() は、特殊な名前 *__wakeup* を有する関数が存在するかどうかを確認し ます。これが存在する場合、この関数は、そのオブジェクトが有する全て のリソースを再構築することが可能です。

典型的な*__wakeup*の使用法は、シリアル化により失っ たデータベース接続を全て再度復旧したり、その他の再初期化作業を実行 したりといったものです。

コンストラクタの内部での参照

コンストラクタの中で参照を作成すると結果が混乱する可能性があります。 本節ではチュートリアル形式で説明しますが、 この問題を避けるために役立つはずです。

```
<?php
class Foo
{
    function Foo($name)
    {
        // 内部への参照グローバル配列 $globalref を作成
        global $globalref;
        $globalref[] = &$this;
        // name を指定した値に設定
        $this->setName($name);
        // それを出力
        $this->echoName();
    }

    function echoName()
    {
        echo "<br>",$this->Name;
    }

    function setName($name)
    {
        $this->Name = $name;
    }
}
?>
```

コピー演算子 = により作成された *$bar1* と 参照演算子 =& により作成された *$bar2* の間の差異があるかどうかを 確認してみましょう。

```
<?php
$bar1 = new foo('set in constructor');
$bar1->echoName();
$globalref[0]->echoName();

/* 出力:
set in constructor
set in constructor
set in constructor */

$bar2 =& new foo('set in constructor');
$bar2->echoName();
$globalref[1]->echoName();

/* 出力:
set in constructor
set in constructor
set in constructor */
?>
```

明らかに違いはありませんが、実際には動作は非常に異なっています。つまり、 *$bar1* と *$globalref[0]* は参照されておらず、同じ変数でもありません。 これは、"new" がデフォルトで参照を返さず、代わりにコピーを返すためです。

> **注意**: (PHP 4 以降ではリファレンスカウンティングを使用しているため)、 参照ではなくコピーを返すことで性能が低下することはありません。逆に 多くの場合、参照を使うよりも単純にコピーを使った方が良い結果となります。 これは、参照の作成には時間がかかりますが、コピーの作成には 理想的には時間が全くかからないからです (ただし、大きな配列 またはオブジェクトでその一つが変更されると、次々に参照先の他の要素に 参照先に波及するといった場合を除きます)。

上記の記述が正しいことを示すために以下のコードを見てみましょう。

```
<?php
// ここで、name を変更してみます。どうなるでしょうか?
// $bar と $globalref[0] の両方共名前が変わると予想するかもしれません...
$bar1->setName('set from outside');

// 前記のようにこの場合は違います。
$bar1->echoName();
$globalref[0]->echoName();

/* 出力:
set from outside
set in constructor */

// $bar2 と $globalref[1] の差を見てみましょう
$bar2->setName('set from outside');

// うまく行けば、値が等しいだけでなく、同じ変数となります。
```

```
// つまり、$bar2->Name と $globalref[1]->Name も同じになります。
$bar2->echoName();
$globalref[1]->echoName();

/* 出力:
set from outside
set from outside */
?>
```

最後に別の例について考えてみてください。

```
<?php
class A {
     function A($i) {
         $this->value = $i;
        // ここで参照を使う必要がない理由を考えてみてください
         $this->b = new B($this);
     }

     function createRef() {
         $this->c = new B($this);
     }

     function echoValue() {
          echo "<br>","class ",get_class($this),': ',$this->value;
     }
}

class B {
     function B(&$a) {
         $this->a = &$a;
     }

     function echoValue() {
          echo "<br>","class ",get_class($this),': ',$this->a->value;
     }
}

// 以下の単純なコピーが、* 印を付けた行で望ましくない結果を生む理由を
// 考えてみてください。
$a =& new A(10);
$a->createRef();

$a->echoValue();
$a->b->echoValue();
$a->c->echoValue();

$a->value = 11;

$a->echoValue();
$a->b->echoValue(); // *
$a->c->echoValue();

?>
```

上の例の出力は以下となります。

```
class A: 10
class B: 10
class B: 10
class A: 11
class B: 11
class B: 11
```

PHP4におけるオブジェクトの比較

PHP4では、とてもシンプルな作法でオブジェクトを比較できます。すなわち、 二つのオブジェクトは同じ属性と同じ値を持ち、同じクラスのインスタンス である場合には等しいのです。二つのオブジェクトを比較演算子 (===)で比較した場合でも同様のルールとなります。

以下の例のようなコードを実行するとしましょう:

Example#1 PHP 4におけるオブジェクトの比較の例

```
<?php
function bool2str($bool) {
     if ($bool === false) {
              return 'FALSE';
     } else {
              return 'TRUE';
     }
}

function compareObjects(&$o1, &$o2) {
     echo 'o1 == o2 : '.bool2str($o1 == $o2)."¥n";
     echo 'o1 != o2 : '.bool2str($o1 != $o2)."¥n";
     echo 'o1 === o2 : '.bool2str($o1 === $o2)."¥n";
     echo 'o1 !== o2 : '.bool2str($o1 !== $o2)."¥n";
}

class Flag {
```

```
    var $flag;

    function Flag($flag=true) {
            $this->flag = $flag;
    }
}

class SwitchableFlag extends Flag {

    function turnOn() {
        $this->flag = true;
    }

    function turnOff() {
        $this->flag = false;
    }
}

$o = new Flag();
$p = new Flag(false);
$q = new Flag();

$r = new SwitchableFlag();

echo "同じ引数で生成されたインスタンスの比較\n";
compareObjects($o, $q);

echo "\n違う引数で生成されたインスタンスの比較\n";
compareObjects($o, $p);

echo "\n親クラスとそのサブクラスのインスタンスの比較\n";

compareObjects($o, $r);
?>
```

上の例の出力は以下となります。

```
同じ引数で生成されたインスタンスの比較
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : TRUE
o1 !== o2 : FALSE

違う引数で生成されたインスタンスの比較
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE

親クラスとそのサブクラスのインスタンスの比較
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

比較のルールに基づいた結果が得られたと思います。 その属性に同じ値を持ち、かつ同じクラスから生成されたインスタンスだけが 等しいとみなされます。

複合的なオブジェクトの場合であっても、同じ比較ルールが適用されます。 以下の例では、Flagオブジェクトの連想配列 が格納されてるコンテナが生成されています。

Example#2 PHP4における複合的なオブジェクトの比較

```
<?php
class FlagSet {
    var $set;

    function FlagSet($flagArr = array()) {
        $this->set = $flagArr;
    }

    function addFlag($name, $flag) {
        $this->set[$name] = $flag;
    }

    function removeFlag($name) {
        if (array_key_exists($name, $this->set)) {
            unset($this->set[$name]);
        }
    }
}


$u = new FlagSet();
$u->addFlag('flag1', $o);
$u->addFlag('flag2', $p);
$v = new FlagSet(array('flag1'=>$q, 'flag2'=>$p));
$w = new FlagSet(array('flag1'=>$q));

echo "¥nComposite objects u(o,p) and v(q,p)¥n";
compareObjects($u, $v);

echo "¥nu(o,p) and w(q)¥n";
```

```
compareObjects($u, $w);
?>
```

上の例の出力は以下となります。

```
Composite objects u(o,p) and v(q,p)
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : TRUE
o1 !== o2 : FALSE

u(o,p) and w(q)
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

PHP 5におけるオブジェクトの比較

警告

この拡張モジュールは、 *実験的* なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

PHP 5では、オブジェクトの比較はPHP 4よりも複雑になり、 オブジェクト指向言語で期待される動作により近くなります。 (PHP 5はオブジェクト指向言語ではありません)

比較演算子(==)を用いた場合、 オブジェクト変数は単純に比較されます。つまり、 同じ属性と値を有し、同じクラスのインスタンスである場合に 二つのオブジェクトのインスタンスが等しくなります。

一方、identity演算子(===)の場合、 オブジェクト変数は、同じクラスの同じインスタンスを参照する場合のみ 等しくなります。

これらの規則を明確にするための例を示します。

Example#1 PHP 5のオブジェクト比較の例

```
<?php
function bool2str($bool) {
    if ($bool === false) {
            return 'FALSE';
    } else {
            return 'TRUE';
    }
}

function compareObjects(&$o1, &$o2) {
    echo 'o1 == o2 : '.bool2str($o1 == $o2)."¥n";
    echo 'o1 != o2 : '.bool2str($o1 != $o2)."¥n";
    echo 'o1 === o2 : '.bool2str($o1 === $o2)."¥n";
    echo 'o1 !== o2 : '.bool2str($o1 !== $o2)."¥n";
}

class Flag {
    var $flag;

    function Flag($flag=true) {
            $this->flag = $flag;
    }
}

class OtherFlag {
    var $flag;

    function OtherFlag($flag=true) {
            $this->flag = $flag;
    }
}

$o = new Flag();
$p = new Flag();
$q = $o;
$r = new OtherFlag();

echo "Two instances of the same class¥n";
compareObjects($o, $p);

echo "¥nTwo references to the same instance¥n";
compareObjects($o, $q);

echo "¥nInstances of two different classes¥n";
compareObjects($o, $r);
?>
```

この例の出力は以下のようになります。

```
Two instances of the same class
o1 == o2 : TRUE
o1 != o2 : FALSE
```

```
o1 === o2 : FALSE
o1 !== o2 : TRUE

Two references to the same instance
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : TRUE
o1 !== o2 : FALSE

Instances of similarly named classes in different namespaces
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

クラスとオブジェクト (PHP 5)

目次

はじめに

PHP 5では、新しいオブジェクトモデルが採用されています。PHPのオブジェクト処理は、 完全に書き直され、より高い性能と高機能を実現しています。

ヒント

ユーザレベルでの命名の手引き もご覧になるとよいでしょう。

クラスの基礎

class

各クラスの定義は、classキーワードで始まり、クラス名が続きます。 クラス名には、PHPの予約語以外のあらゆる名前を 使用することができます。 波括弧の中に、クラスのメンバーとメソッドの定義が記述されます。 メソッドがオブジェクトコンテキストからコールされる場合 (通常は、メソッドが属するオブジェクトですが、 メソッドが第二のオブジェクトのオブジェクトの コンテキストから スタティックに コールされる場合には、別のオブジェクトとなる場合もあります)、 疑似変数 *$this* が利用可能です。 以下にこの例を示します。

Example#1 オブジェクト指向言語における *$this* 変数

```php
<?php
class A
{
    function foo()
    {
        if (isset($this)) {
            echo '$this is defined (';
            echo get_class($this);
            echo ")¥n";
        } else {
            echo "¥$this is not defined.¥n";
        }
```

```
    }
}

class B
{
     function bar()
    {
        A::foo();
    }
}

$a = new A();
$a->foo();
A::foo();
$b = new B();
$b->bar();
B::bar();
?>
```

上の例の出力は以下となります。

```
$this is defined (a)
$this is not defined.
$this is defined (b)
$this is not defined.
```

Example#2 簡単なクラス定義

```
<?php
class SimpleClass
{
    // メンバ宣言
    public $var = 'a default value';

    // メソッド宣言
    public function displayVar() {
        echo $this->var;
    }
}
?>
```

デフォルト値は定数でなければなりません。(たとえば) 変数、クラスのメンバあるいは関数コールなどは使用できません。

Example#3 クラスのメンバのデフォルト値

```
<?php
class SimpleClass
{
    // 無効な形式のメンバ宣言
    public $var1 = 'hello '.'world';
    public $var2 = <<<EOD
hello world
EOD;
    public $var3 = 1+2;
    public $var4 = self::myStaticMethod();
    public $var5 = $myVar;

    // 有効な宣言
    public $var6 = myConstant;
    public $var7 = self::classConstant;
    public $var8 = array(true, false);

}
?>
```

注意: クラスやオブジェクトを扱うための、便利な関数があります。 クラス/オブジェクト関数 を参照ください。

new

あるクラスのインスタンスを生成する際、新たにオブジェクトが作成され、 変数に代入される必要があります。 新しいオブジェクトが作成される際には、そのオブジェクトが エラー時に 例外を投げる よう定義された コンストラクタを有していない限り、 常にオブジェクトが代入されます。 クラスは、そのインスタンスを作成する前に定義すべきです (これが必須となる場合もあります)。

Example#4 インスタンスを作成する

```
<?php
$instance = new SimpleClass();
?>
```

クラスのコンテキストにおいては、 *new self* や *new parent* のようにして新しいオブジェクトを作成することができます。

作成済みのクラスのインスタンスを新たな変数に代入する場合、新しい変数は、 代入されたオブジェクトと同じインスタンスにアクセスします。 この動作は、インスタンスを関数に渡す場合も同様です。 作成済みのオブジェクトのコピーは、その クローンを作成 することにより作成可能です。

Example#5 オブジェクトの代入

```
<?php
$assigned   =  $instance;
```

```
$reference   =& $instance;

$instance->var = '$assigned will have this value';

$instance = null; // $instance と $reference は null になります

var_dump($instance);
var_dump($reference);
var_dump($assigned);
?>
```

上の例の出力は以下となります。

```
NULL
NULL
object(SimpleClass)#1 (1) {
    ["var"]=>
      string(30) "$assigned will have this value"
}
```

extends

クラスは、宣言部にextendsキーワードを含めることで、他のクラスのメソッドとメソッドと メンバーを継承することができます。他重継承を行うことはできず、クラスが継承できるベース クラスは一つだけです。

継承されたメソッドとメンバーは、親クラスで final としてメソッドが定義されていない限り、親クラスで定義されたのと同じ名前で 再度宣言を行うことでオーバーライドすることができます。 parent::で 参照することにより、このオーバーライドされたメソッドまたはスタティックメンバーに アクセスすることができます。

Example#6 簡単なクラスの継承

```
<?php
class ExtendClass extends SimpleClass
{
    // 親クラスのメソッドを再定義
    function displayVar()
    {
        echo "Extending class¥n";
        parent::displayVar();
    }
}

$extended = new ExtendClass();
$extended->displayVar();
?>
```

上の例の出力は以下となります。

```
Extending class
a default value
```

オブジェクトのオートローディング

オブジェクト指向アプリケーションを作成する開発者の多くは、 クラス定義毎に一つのPHPソースファイルを作成します。 最大の問題は、各スクリプトの先頭に、必要な読み込みを行う長いリストを 記述する必要があることです(各クラスについて一つ)。

PHP 5では、これはもう不用です。 未定義のクラスを使用しようとした時に 自動的にコールされる __autoload 関数を定義することができます。 この関数をコールすることにより、 スクリプトエンジンは、PHPがエラーで止まる前にクラスをロードする最後の チャンスを与えます。

> **注意**: __autoload 関数の内部でスローされた例外は、 catch ブロックでキャッチすることが できず、致命的なエラーとなります。
>
> **注意**: オートローディングは、PHP を CLI 対話モード で実行している場合は使用できません。
>
> **注意**: クラス名をたとえば call_user_func() などで使用する場合、 ../ のような危険な文字が含まれることもあり得ます。 このような関数にはユーザの入力を渡さないことをおすすめします。 あるいは最低限 **__autoload()** の中で入力内容を検証するようにします。

Example#1 オートロードの例

この例は、 クラス *MyClass1* および *MyClass2* をそれぞれ *MyClass1.php* および *MyClass2.php* からロードします。

```
<?php
function __autoload($class_name) {
    include_once $class_name . '.php';
}

$obj  = new MyClass1();
$obj2 = new MyClass2();
?>
```

コンストラクタとデストラクタ

コンストラクタ

void **__construct** ([mixed $args [, $...]])

PHP 5 では、開発者がクラスのコンストラクタメソッドを宣言することが できます。コンストラクタメソッドを有するクラスは、新たにオブジェクトが 生成される度にこのメソッドをコールします。これにより、 そのオブジェクトを使用する前に必要な初期化を行うことができます。

> **注意**: 子クラスがコンストラクタを有している場合、親クラスのコンストラクタが 暗黙の内にコールされることはありません。 親クラスのコンストラクタを実行するには、子クラスのコンストラクタの 中で **parent::__construct()** をコールすることが 必要です。

Example#1 新しい統一されたコンストラクタを使用する

```
<?php
class BaseClass {
    function __construct() {
        print "In BaseClass constructor¥n";
    }
}

class SubClass extends BaseClass {
    function __construct() {
        parent::__construct();
        print "In SubClass constructor¥n";
    }
}

$obj = new BaseClass();
$obj = new SubClass();
?>
```

下位互換性を維持するため、PHP 5 が指定されたクラスの **__construct()** 関数をみつけられない場合には、 古い形式のコンストラクタ関数、つまり、そのクラスの名前と同じ関数が探されます。 実際、互換性の問題が発生する可能性があるのは、 そのクラスが **__construct()** という名前のメソッドを有しており、それが異なる用途で使用されている場合です。

デストラクタ

void **__destruct** (void)

PHP 5 では、C++ のような他のオブジェクト指向言語に似た概念のデストラクタが 導入されました。デストラクタメソッドは、特定のオブジェクトへの全てのリファレンスが 削除された直後やオブジェクトが明示的に破棄された直後にコールされます。 あるいは、スクリプトの終了時にも順不同でコールされます。

Example#2 デストラクタの例

```
<?php
class MyDestructableClass {
    function __construct() {
        print "In constructor¥n";
        $this->name = "MyDestructableClass";
    }

    function __destruct() {
        print "Destroying " . $this->name . "¥n";
    }
}

$obj = new MyDestructableClass();
?>
```

コンストラクタと同様、親クラスのデストラクタがエンジンにより暗黙のうちに コールされるということはありません。親クラスのデストラクタを実行するには、 デストラクタの中で明示的に **parent::__destruct()** をコールする必要があります。

> **注意**: スクリプトのシャットダウン時にデストラクタがコールされた場合は、 HTTP ヘッダはすでに送信されています。スクリプトのシャットダウン時の作業ディレクトリは、 SAPI によっては (たとえば Apache など) 異なります。

> **注意**: デストラクタの中から (スクリプトの終了処理時に) 例外をスローしようとすると、致命的なエラーを引き起こします。

アクセス権

プロパティまたはメソッドのアクセス権 (visibility) は、 キーワード: public, protected または private を指定することにより、 定義できます。public として定義されたアイテムには、どこからでもアクセス可能です。 protected は、派生クラスや親クラス (とそれを定義するクラス自体) にアクセスを制限します。private は、 それを定義するクラスのみにアクセス権を制限します。

メンバのアクセス権

クラスのメンバは、public, private, または protected として定義されなくてはなりません。

Example#1 メンバの宣言

```
<?php
/**
 * MyClass の定義
 */
class MyClass
{
     public $public = 'Public';
     protected $protected = 'Protected';
     private $private = 'Private';

     function printHello()
     {
          echo $this->public;
          echo $this->protected;
          echo $this->private;
     }
}

$obj = new MyClass();
echo $obj->public; // 動作します
echo $obj->protected; // Fatal エラー
echo $obj->private; // Fatal エラー
$obj->printHello(); // Public、Protected そして Private を表示します


/**
 * MyClass2 の定義
 */
class MyClass2 extends MyClass
{
    // public および protected メソッドは再定義できますが、
     // private はできません。
     protected $protected = 'Protected2';

     function printHello()
     {
          echo $this->public;
          echo $this->protected;
          echo $this->private;
     }
}

$obj2 = new MyClass2();
echo $obj2->public; // 動作します
echo $obj2->private; // 未定義です
echo $obj2->protected; // Fatal エラー
$obj2->printHello(); // Public、Protected2、Undefined を表示します

?>
```

> **注意**: キーワード *var* で変数を宣言する PHP 4 の方法は、互換性を保つために今でもサポートされています (これは public と同じ扱いになります)。PHP 5.1.3 より前では、 これを使用すると **E_STRICT** 警告が発生します。

メソッドのアクセス権

クラスメソッドは、public, private, または protected として定義される必要があります。どの宣言も有さないメソッドは、 public として定義されます。

Example#2 メソッドの宣言

```
<?php
/**
 * MyClass の定義
 */
class MyClass
{
    // コンストラクタは public でなければなりません
     public function __construct() { }

    // public メソッドの宣言
     public function MyPublic() { }

    // protected メソッドの宣言
     protected function MyProtected() { }

    // private メソッドの宣言
     private function MyPrivate() { }

    // これは public となります
     function Foo()
     {
          $this->MyPublic();
          $this->MyProtected();
          $this->MyPrivate();
     }
}

$myclass = new MyClass;
$myclass->MyPublic(); // 動作します
$myclass->MyProtected(); // Fatal エラー
$myclass->MyPrivate(); // Fatal エラー
$myclass->Foo(); // Public、Protected および Private が動作します
```

```
/**
 * MyClass2 の定義
 */
class MyClass2 extends MyClass
{
    // これは public となります
    function Foo2()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate(); // Fatal エラー
    }
}

$myclass2 = new MyClass2;
$myclass2->MyPublic(); // 動作します
$myclass2->Foo2(); // Public および Protected は動作しますが、Private は動作しません

class Bar
{
     public function test() {
        $this->testPrivate();
        $this->testPublic();
    }

     public function testPublic() {
        echo "Bar::testPublic¥n";
    }

     private function testPrivate() {
        echo "Bar::testPrivate¥n";
    }
}

class Foo extends Bar
{
     public function testPublic() {
        echo "Foo::testPublic¥n";
    }

     private function testPrivate() {
        echo "Foo::testPrivate¥n";
    }
}

$myFoo = new foo();
$myFoo->test(); // Bar::testPrivate
                // Foo::testPublic
?>
```

スコープ定義演算子 (::)

スコープ定義演算子 (またの名を Paamayim Nekudotayim)、 平たく言うと「ダブルコロン」は、トークンのひとつです。 static, 定数 およびオーバーライドされたクラスのメンバやメソッドにアクセスすることができます。

これらの要素をクラス定義の外から参照する際には、 クラスの名前を使用してください。

PHP 5.3.0 以降では、変数を用いてクラスを参照することも可能です。 *self* や *parent*、 *static* といったキーワードは 動的なクラス参照では使用できません。

なぜダブルコロンに Paamayim Nekudotayim という名前をつけたのか、 ちょっと奇妙に感じられるかもしれません。 しかし、Zend Engine 0.5 (PHP 3のエンジン) を 書いている時に、Zend チームはこう呼ぶと決めたのです。 この奇妙な名前は、実はダブルコロンを意味するヘブライ語なのです!

Example#1 クラス定義の外からの ::

```
<?php
class MyClass {
    const CONST_VALUE = 'A constant value';
}

$classname = 'MyClass';
echo $classname::CONST_VALUE;

echo MyClass::CONST_VALUE;
?>
```

二つの特別なキーワード*self*と*parent* がクラス定義の内部からメンバまたはメソッドにアクセスする際に使用されます。

Example#2 クラス定義の中からの ::

```
<?php
class OtherClass extends MyClass
{
    public static $my_static = 'static var';

     public static function doubleColon() {
        echo parent::CONST_VALUE . "¥n";
```

```
        echo self::$my_static . "¥n";
    }
}

$classname = 'OtherClass';
echo $classname::doubleColon();

OtherClass::doubleColon();
?>
```

拡張されたクラスが親クラスのメソッドの定義をオーバーライドする際、 PHPは親クラスのメソッドをコールしません。 親クラスのメソッドをコールするかしないかは、 拡張されたクラスに責任があります。 これは、コンストラクタおよびデストラクタ, オーバーロード, そして マジック メソッドの定義にも 適用されます。

Example#3 親クラスのメソッドをコールする

```
<?php
class MyClass
{
     protected function myFunc() {
        echo "MyClass::myFunc()¥n";
    }
}

class OtherClass extends MyClass
{
    // Override parent's definition
    public function myFunc()
    {
        // But still call the parent function
        parent::myFunc();
        echo "OtherClass::myFunc()¥n";
    }
}

$class = new OtherClass();
$class->myFunc();
?>
```

---

static キーワード

クラスメンバもしくはメソッドを static として宣言することで、 クラスのインスタンス化の必要なしにアクセスすることができます。 static なメンバは、インスタンス化されたクラスオブジェクトから アクセスすることはできません (static なメソッドからは可能です) 。

PHP 4 との互換性を維持するため、 可視性の宣言が ない場合、そのメンバまたはメソッドは、 *public*として宣言されていると みなされます。

static メソッドは、オブジェクトのインスタンスを生成せずに コールすることができます。疑似変数 *$this* は、 static として宣言されたメソッドの内部から利用することはできません。

static プロパティは、矢印演算子 -> によりオブジェクトからアクセス することはできません。

static でないメソッドを静的にコールすると、E_STRICT レベルの 警告が発生します。

PHP 5.3.0 以降では、変数を用いてクラスを参照することも可能です。 *self* や *parent*、 *static* といったキーワードは 動的なクラス参照では使用できません。

Example#1 static メンバの例

```
<?php
class Foo
{
     public static $my_static = 'foo';

     public function staticValue() {
        return self::$my_static;
    }
}

class Bar extends Foo
{
     public function fooStatic() {
        return parent::$my_static;
    }
}


print Foo::$my_static . "¥n";

$foo = new Foo();
print $foo->staticValue() . "¥n";
print $foo->my_static . "¥n";      // Undefined "Property" my_static

print $foo::$my_static . "¥n";
$classname = 'Foo';
print $classname::$my_static . "¥n";

print Bar::$my_static . "¥n";
$bar = new Bar();
print $bar->fooStatic() . "¥n";
```

```
?>
```

Example#2 static メソッドの例

```
<?php
class Foo {
     public static function aStaticMethod() {
        // ...
    }
}

Foo::aStaticMethod();
$classname = 'Foo';
$classname::aStaticMethod();
?>
```

オブジェクト定数

値が変更できない定数をクラス内に定義することができます。 定数は、通常の変数とは異なり、定義または使用する際に $ 記号を付けません。

定義する値は定数表現である必要があり、(例えば)変数・クラスのメンバー・ 演算結果あるいは関数のコールなどであってはいけません。

PHP 5.3.0 以降では、変数を用いてクラスを参照することも可能です。 *self* や *parent*、 *static* といったキーワードは 動的なクラス参照では使用できません。

Example#1 定数の定義と使用

```
<?php
class MyClass
{
    const constant = 'constant value';

    function showConstant() {
        echo self::constant . "¥n";
    }
}

echo MyClass::constant . "¥n";

$classname = "MyClass";
echo $classname::constant . "¥n";

$class = new MyClass();
$class->showConstant();

echo $class::constant."¥n";
?>
```

クラスの抽象化

PHP 5 では、抽象クラスとメソッドが導入されました。 abstractとして宣言されたクラスのインスタンスを生成することはできません。 1 つ以上の抽象メソッドを含む全てのクラスもまた抽象クラスとなります。 abstractとして定義されたメソッドは、そのメソッドの外観を宣言するのみで、 実装を定義することはできません。

抽象クラスから継承する際、親クラスの宣言で abstract としてマークされた 全てのメソッドは、子クラスで定義されなければなりません。加えて、これらのメソッドは同等 (あるいはより緩い制約) の 可視性 で定義される必要があります。例えば、抽象メソッドが protected として定義された場合、その関数の実装は protected または public のどちらかとして定義する必要があります。private にすることはできません。

Example#1 抽象クラスの例

```
<?php
abstract class AbstractClass
{
   // 拡張クラスにこのメソッドの定義を強制する
    abstract protected function getValue();
     abstract protected function prefixValue($prefix);

    // Common method
    public function printOut() {
        print $this->getValue() . "¥n";
    }
}

class ConcreteClass1 extends AbstractClass
{
    protected function getValue() {
        return "ConcreteClass1";
    }

    public function prefixValue($prefix) {
        return "{$prefix}ConcreteClass1";
    }
}

class ConcreteClass2 extends AbstractClass
{
```

```
     public function getValue() {
         return "ConcreteClass2";
     }

     public function prefixValue($prefix) {
         return "{$prefix}ConcreteClass2";
     }
}

$class1 = new ConcreteClass1;
$class1->printOut();
echo $class1->prefixValue('FOO_') ."¥n";

$class2 = new ConcreteClass2;
$class2->printOut();
echo $class2->prefixValue('FOO_') ."¥n";
?>
```

上の例の出力は以下となります。

```
ConcreteClass1
FOO_ConcreteClass1
ConcreteClass2
FOO_ConcreteClass2
```

'abstract'という名前のユーザー定義関数または関数を有さない コードは修正なしに動作します。

オブジェクト インターフェイス

オブジェクトインターフェイスにより、あるクラスが実装する必要があるメソッドの 種類を、これらのメソッドの実体を定義することなく、指定するコードを作成できる ようになります。

インターフェイスはキーワードinterfaceにより定義され、通常のクラスと 同様に定義することができますが、メソッドの実装は全く定義されません。

インターフェイス内で宣言される全てのメソッドはpublicである必要があります。 これは、インターフェイスの特性によります。

implements

インターフェイスを実装するには、*implements* 演算子を使用し、 このインターフェイスに含まれる全てのメソッドを実装する必要があります。 実装されていない場合、致命的エラーとなります。 各インターフェイスをカンマで区切って指定することで、 クラスは複数のインターフェイスを実装することができます。

> **注意**: ひとつのクラスの中で、同じ名前の関数を含む 2 つのインターフェイスを 実装することはできません。あいまいさを解決できなくなるためです。

例

Example#1 Interface の例

```
<?php
// インターフェイス 'iTemplate' を宣言する
interface iTemplate
{
     public function setVariable($name, $var);
     public function getHtml($template);
}

// インターフェイスを実装する。
// これは動作します。
class Template implements iTemplate
{
     private $vars = array();

     public function setVariable($name, $var)
     {
         $this->vars[$name] = $var;
     }

     public function getHtml($template)
     {
          foreach($this->vars as $name => $value) {
              $template = str_replace('{' . $name . '}', $value, $template);
          }

          return $template;
     }
}

// これは動作しません。
// Fatal error: Class BadTemplate contains 1 abstract methods
// and must therefore be declared abstract (iTemplate::getHtml)
class BadTemplate implements iTemplate
{
     private $vars = array();
```

```
     public function setVariable($name, $var)
     {
         $this->vars[$name] = $var;
     }
}

?>
```

instanceof 演算子も参照ください。

オーバーロード

メソッド呼び出しとメンバーへのアクセスのいずれについても、 __call, __get そして __set メソッドを通してオーバーロードすることができます。 オブジェクトや継承されたオブジェクトが アクセスしようとしている public メンバーもしくはメソッドを含んでいない場合でも、 これらのメソッドは実行されます。 オーバーロードしている全てのメソッドは、 static として定義されていてはいけません。 オーバーロードしている全てのメソッドは public として定義されていなければなりません。

PHP 5.1.0 以降、__isset や __unset メソッドを通じて isset() 関数や unset() 関数を個々にオーバーロードする事も可能です。 メソッド __isset は、empty() でもコールされます。

メンバーのオーバーロード

void **__set** (string $name , mixed $value)
mixed **__get** (mixed $name)
bool **__isset** (string $name)
void **__unset** (string $name)

独自のクラスで定義されているカスタムコードを実行するために、 特別な名前を持つメソッドによってクラスメンバーをオーバーロード することができます。 使用される *$name* パラメータは、 設定あるいは取得される変数名です。 __set() メソッドの *$value* パラメータにより オブジェクトが *$name* に設定する値を指定します。

> **注意**: *__set()* メソッドでは、引数を参照渡しで受け取ることはできません。

Example#1 __get, __set, __isset, __unset を使ったオーバーロードの例

```
<?php
class Setter
{
     public $n;
     private $x = array("a" => 1, "b" => 2, "c" => 3);

     public function __get($nm)
     {
          echo "Getting [$nm]¥n";

          if (isset($this->x[$nm])) {
              $r = $this->x[$nm];
              print "Returning: $r¥n";
              return $r;
          } else {
              echo "Nothing!¥n";
          }
     }

     public function __set($nm, $val)
     {
          echo "Setting [$nm] to $val¥n";

          if (isset($this->x[$nm])) {
              $this->x[$nm] = $val;
              echo "OK!¥n";
          } else {
              echo "Not OK!¥n";
          }
     }

     public function __isset($nm)
     {
          echo "Checking if $nm is set¥n";

          return isset($this->x[$nm]);
     }

     public function __unset($nm)
     {
          echo "Unsetting $nm¥n";

          unset($this->x[$nm]);
     }
}

$foo = new Setter();
$foo->n = 1;
$foo->a = 100;
$foo->a++;
$foo->z++;
```

```
var_dump(isset($foo->a)); //true
unset($foo->a);
var_dump(isset($foo->a)); //false

// this doesn't pass through the __isset() method
// because 'n' is a public property
var_dump(isset($foo->n));

var_dump($foo);
?>
```

上の例の出力は以下となります。

```
Setting [a] to 100
OK!
Getting [a]
Returning: 100
Setting [a] to 101
OK!
Getting [z]
Nothing!
Setting [z] to 1
Not OK!

Checking if a is set
bool(true)
Unsetting a
Checking if a is set
bool(false)
bool(true)

object(Setter)#1 (2) {
   ["n"]=>
   int(1)
    ["x":"Setter":private]=>
   array(2) {
     ["b"]=>
     int(2)
     ["c"]=>
     int(3)
  }
}
```

メソッドのオーバーロード

mixed **__call** (string $name , array $arguments)

特別なメソッド __call() を使用すると、存在しないメソッドの呼び出しを捕捉することができます。 つまり、__call() を使用すると、実際にコールされたメソッドの名前に応じた ユーザ定義の処理を実装することができるということです。 これは、たとえばプロキシを実装する場合などに便利です。 関数に渡された引数はパラメータ *$arguments* で受け取ることができ、 __call() メソッドの返り値が呼び出し元のメソッドの返り値となります。

Example#2 __call を使ったオーバーロードの例

```
<?php
class Caller
{
     private $x = array(1, 2, 3);

     public function __call($m, $a)
     {
          print "Method $m called:¥n";
          var_dump($a);
          return $this->x;
     }
}

$foo = new Caller();
$a = $foo->test(1, "2", 3.4, true);
var_dump($a);
?>
```

上の例の出力は以下となります。

```
Method test called:
array(4) {
     [0]=>
     int(1)
     [1]=>
     string(1) "2"
     [2]=>
     float(3.4)
     [3]=>
     bool(true)
}
array(3) {
     [0]=>
     int(1)
     [1]=>
     int(2)
     [2]=>
     int(3)
}
```

オブジェクトのイタレーション

PHP 5は、アイテムのリストに関して反復処理、例えば、 foreach命令、 を可能とするように、オブジェクトを定義する手段を提供します。 デフォルトで、 全ての アクセス権限がある プロパティは、反復処理に使用することができます。

Example#1 Simple Object Iteration

```php
<?php
class MyClass
{
     public $var1 = 'value 1';
     public $var2 = 'value 2';
     public $var3 = 'value 3';

     protected $protected = 'protected var';
     private   $private   = 'private var';

     function iterateVisible() {
        echo "MyClass::iterateVisible:¥n";
         foreach($this as $key => $value) {
             print "$key => $value¥n";
         }
     }
}

$class = new MyClass();

foreach($class as $key => $value) {
     print "$key => $value¥n";
}
echo "¥n";


$class->iterateVisible();

?>
```

上の例の出力は以下となります。

```
var1 => value 1
var2 => value 2
var3 => value 3

MyClass::iterateVisible:
var1 => value 1
var2 => value 2
var3 => value 3
protected => protected var
private => private var
```

出力からわかるように、 foreachによる 反復処理が全ての アクセス権がある 変数について行われています。 さらに、*Iterator*という名前のPHP 5の内部 interface を*implement*することもできます。 これにより、オブジェクトは、どうやって、反復されるかを決めることが できます。

Example#2 Iteratorを実装するオブジェクトの反復処理

```php
<?php
class MyIterator implements Iterator
{
     private $var = array();

     public function __construct($array)
     {
         if (is_array($array)) {
             $this->var = $array;
         }
     }
```

```
    public function rewind() {
        echo "rewinding¥n";
        reset($this->var);
    }

    public function current() {
        $var = current($this->var);
        echo "current: $var¥n";
        return $var;
    }

    public function key() {
        $var = key($this->var);
        echo "key: $var¥n";
        return $var;
    }

    public function next() {
        $var = next($this->var);
        echo "next: $var¥n";
        return $var;
    }

    public function valid() {
        $var = $this->current() !== false;
        echo "valid: {$var}¥n";
        return $var;
    }
}

$values = array(1,2,3);
$it = new MyIterator($values);

foreach ($it as $a => $b) {
    print "$a: $b¥n";
}
?>
```

上の例の出力は以下となります。

```
rewinding
current: 1
valid: 1
current: 1
key: 0
0: 1
next: 2
current: 2
valid: 1
current: 2
key: 1
1: 2
next: 3
current: 3
valid: 1
current: 3
key: 2
2: 3
next:
current:
valid:
```

単にPHP 5の*IteratorAggregate*インターフェイスを 実装することにより、*Iterator*関数を 全く定義する必要なく、自分のクラスを定義することも可能です。

Example#3 IteratorAggregateを実装するオブジェクトの反復処理

```
<?php
class MyCollection implements IteratorAggregate
{
    private $items = array();
    private $count = 0;

    // Required definition of interface IteratorAggregate
    public function getIterator() {
        return new MyIterator($this->items);
    }

    public function add($value) {
        $this->items[$this->count++] = $value;
    }
}

$coll = new MyCollection();
$coll->add('value 1');
$coll->add('value 2');
$coll->add('value 3');

foreach ($coll as $key => $val) {
    echo "key/value: [$key -> $val]¥n¥n";
}
?>
```

上の例の出力は以下となります。

```
rewinding
current: value 1
valid: 1
current: value 1
key: 0
key/value: [0 -> value 1]

next: value 2
current: value 2
valid: 1
current: value 2
key: 1
key/value: [1 -> value 2]

next: value 3
current: value 3
valid: 1
current: value 3
key: 2
key/value: [2 -> value 3]

next:
current:
valid:
```

注意: より詳細なイタレータの例については、 SPLエクステンションを 参照してください。

パターン

パターンは、最善の手順と良い設計を記述するための手段です。 パターンは、一般的なプログラム上の課題に柔軟な解決策を提供します。

Factory

Factory パターンにより、実行時にオブジェクトを初期化できるようになります。 オブジェクトを"製造する"ことに似ているため、これは Factory パターンと呼ばれています。 パラメータ化された Factory が、生成するクラス名を引数として受け取ります。

Example#1 パラメータ化された Factory メソッド

```php
<?php
class Example
{
    // パラメータ化された Factory メソッド
    public static function factory($type)
    {
          if (include_once 'Drivers/' . $type . '.php') {
             $classname = 'Driver_' . $type;
              return new $classname;
          } else {
              throw new Exception ('Driver not found');
          }
    }
}
?>
```

このメソドをクラス内で定義することで、実行時に ロードされるドライバを作成できるようになります。 *Example* クラスが、データベース抽象化クラスで、 *MySQL* および *SQLite* ドライバをロードするとすると以下のように行うことができます。

```php
<?php
// MySQL ドライバをロード
$mysql = Example::factory('MySQL');

// SQLite ドライバをロード
$sqlite = Example::factory('SQLite');
?>
```

Singleton

Singleton パターンは、クラスのインスタンスが一つだけであることが 必要である場合に適用されます。 この最も一般的な例は、データベースへの接続です。 このパターンを実装することで、プログラマは この単一のインスタンスが他の多くのオブジェクトから容易に アクセスできるようにすることができます。

Example#2 Singleton 関数

```php
<?php
class Example
{
    // クラスのインスタンスを保持する
    private static $instance;

    // private なコンストラクタ。オブジェクトが直接生成されるのを防ぐ
    private function __construct()
    {
        echo 'I am constructed';
    }
```

```
    // singleton メソッド
    public static function singleton()
    {
         if (!isset(self::$instance)) {
            $c = __CLASS__;
            self::$instance = new $c;
        }

         return self::$instance;
    }

    // とあるメソッド
    public function bark()
    {
         echo 'Woof!';
    }

    // ユーザーがインターフェースを複製するのを防ぐ
    public function __clone()
    {
        trigger_error('Clone is not allowed.', E_USER_ERROR);
    }

}

?>
```

このコードにより、*Example* クラスのインスタンスが一つ 作られ、取得されます。

```
<?php
// コンストラクタが private であるため、これは失敗します
$test = new Example;

// これにより、クラスの単一のインスタンスを取得します。
$test = Example::singleton();
$test->bark();

// これにより、E_USER_ERROR が発生する
$test_clone = clone $test;

?>
```

マジックメソッド

以下の関数名 *__construct*, *__destruct* (コンストラクタとデストラクタ参照), *__call*, *__get*, *__set*, *__isset*, *__unset* (オーバーローディング参照), *__sleep*, *__wakeup*, *__toString*, *__set_state* および __clone は、PHP クラスにおける特殊関数の名前です。 これらの関数に関連する特別な機能を使用する場合を除き、 クラス内にこれらの名前を有する関数を作成してはいけません。

警告

PHP は、__ で始まる関数名を特殊関数として予約しています。 文書化された特殊な機能を必要とする場合を除き、 __ で始まる関数名を使用しないことが推奨されます。

__sleep と *__wakeup*

serialize() は、クラスに特殊な名前 *__sleep* の関数があるかどうかを調べます。 もしあれば、シリアル化の前にその関数を実行します。 この関数で、オブジェクトをクリアすることができます。 またこの関数は、シリアル化するオブジェクトについて、 すべての変数の名前を配列で返すことが前提となっています。 このメソッドが何も返さなかった場合は、**NULL** がシリアル化され、E_NOTICE が発生します。

典型的な *__sleep* の使用法は、 途中のデータをコミットしたり、 似たようなタスクのクリアを行うといったものです。 また、オブジェクトが非常に大きく、かつ、完全に保存する必要がない場合、 この関数が有用です。

逆に、unserialize() は、 特殊な名前 *__wakeup* を有する 関数の存在を調べます。 もし存在する場合、この関数は、オブジェクトが有する可能性が あるあらゆるリソースを再構築することができます。

意図される *__wakeup* の使用法は、 シリアル化の際に失われたデータベース接続を再度確立したり、 その他の再初期化を行うことです。

Example#1 Sleep および wakeup

```
<?php
class Connection {
     protected $link;
     private $server, $username, $password, $db;

     public function __construct($server, $username, $password, $db)
     {
         $this->server = $server;
         $this->username = $username;
         $this->password = $password;
         $this->db = $db;
         $this->connect();
     }

     private function connect()
     {
         $this->link = mysql_connect($this->server, $this->username, $this->password);
         mysql_select_db($this->db, $this->link);
```

```
    }

     public function __sleep()
    {
         return array('server', 'username', 'password', 'db');
    }

     public function __wakeup()
    {
        $this->connect();
    }
}
?>
```

__toString

__toString メソッドにより、 クラスが文字列に変換される際の動作を決めることができます。

Example#2 簡単な例

```
<?php
// 簡単なクラスを宣言
class TestClass
{
     public $foo;

     public function __construct($foo) {
        $this->foo = $foo;
    }

     public function __toString() {
         return $this->foo;
    }
}

$class = new TestClass('Hello');
echo $class;
?>
```

上の例の出力は以下となります。

```
Hello
```

注意が必要なのは、PHP 5.2.0 より前では、 *__toString* メソッドは echo() または print(). と直接結合された場合のみコールされていたということです。 PHP 5.2.0 以降では、これはすべての文字列コンテキスト (たとえば printf() における *%s* 修飾子) でコールされます。しかし、その他の型のコンテキスト (たとえば *%d* 修飾子) ではコールされません。 PHP 5.2.0 以降では、*__toString* メソッドを持っていないオブジェクトを文字列に変換しようとすると **E_RECOVERABLE_ERROR** が発生します。

__set_state

この static メソッドは、 PHP 5.1.0 以降で var_export() によって エクスポートされたクラスのためにコールされます。

このメソッドの唯一のパラメータは、エクスポートされたプロパティを *array('property' => value, ...)* の形式で保持する 配列です。

---

finalキーワード

PHP 5 ではキーワードfinalが導入され、 finalを前に付けて定義されたメソッドは子クラスから上書きできません。 クラス自体がfinalと定義された場合には、このクラスを拡張することはできません。

Example#1 finalメソッドの例

```
<?php
class BaseClass {
    public function test() {
        echo "BaseClass::test() called¥n";
   }

    final public function moreTesting() {
        echo "BaseClass::moreTesting() called¥n";
   }
}

class ChildClass extends BaseClass {
    public function moreTesting() {
        echo "ChildClass::moreTesting() called¥n";
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

Example#2 finalクラスの例

```
<?php
final class BaseClass {
```

```
    public function test() {
        echo "BaseClass::test()  called¥n";
    }

    // Here it doesn't matter if you specify the function as final or not
    final public function moreTesting() {
        echo "BaseClass::moreTesting()  called¥n";
    }
}

class ChildClass extends BaseClass {
}
// Results in Fatal error: Class ChildClass may not inherit from final class (BaseClass)
?>
```

オブジェクトのクローン作成

オブジェクトのコピーを作成する際、そのプロパティも全て二重化することが、 常に望ましい動作であるわけではありません。 コピーコンストラクタが必要となる例として、 GTKウインドウを表すオブジェクトを有しており、 そのオブジェクトがGTKウインドウのリソースを保持している際、 コピーを作成する時に、同じプロパティを有するウインドウを作成し、 その新しいオブジェクトがその新しいウインドウのリソースを保持する ようにしたい場合が考えられます。 他の例としては、 オブジェクトがそのオブジェクトが使用する他のオブジェクトへのリファレンスを 保持しており、親オブジェクトをコピーする際に、そのコピーが独立したオブジェクトの コピーを有するように、そのオブジェクトのインスタンスを新たに作成したい場合が 考えられます。

オブジェクトのコピーは、clone キーワード (これは、そのオブジェクトの __clone()メソッドがある場合にこれをコールします)により作成されます。オブジェクトの__clone()メソッドを直接コールすることはできません。

```
$copy_of_object = clone $object;
```

オブジェクトのクローンが作成される際、PHP 5 は、そのオブジェクトのプロパティを 全てシャローコピーします。他の変数へのリファレンスを保持する全てのプロパティは、 リファレンスのままとなります。 __clone() メソッドが定義された場合、新規の作成されたオブジェクトの __clone() メソッドがコールされるため、この中で、プロパティに 必要な変更を行うことができます。

Example#1 オブジェクトのクローン作成

```
<?php
class SubObject
{
    static $instances = 0;
    public $instance;

    public function __construct() {
        $this->instance = ++self::$instances;
    }

    public function __clone() {
        $this->instance = ++self::$instances;
    }
}

class MyCloneable
{
    public $object1;
    public $object2;

    function __clone()
    {
        // this->object のコピーを作成します。こうしないと、
        // 同じオブジェクトを指すことになってしまいます。
        $this->object1 = clone $this->object1;
    }
}

$obj = new MyCloneable();

$obj->object1 = new SubObject();
$obj->object2 = new SubObject();

$obj2 = clone $obj;


print("元のオブジェクト\n");
print_r($obj);

print("クローンオブジェクト\n");
print_r($obj2);

?>
```

上の例の出力は以下となります。

```
元のオブジェクト
MyCloneable Object
(
    [object1] => SubObject Object
        (
            [instance] => 1
        )

    [object2] => SubObject Object
```

```
        (
            [instance] => 2
        )

)
```

クローンオブジェクト

```
MyCloneable Object
(
    [object1] => SubObject Object
        (
            [instance] => 3
        )

    [object2] => SubObject Object
        (
            [instance] => 2
        )

)
```

オブジェクトの比較

PHP 5では、オブジェクトの比較は、オブジェクト指向言語に期待される動作に対応し、 PHP 4よりも複雑になっています。(ただし、PHP 5はオブジェクト指向言語ではありません。)

比較演算子(==)を使用する際、 オブジェクト変数は、単純に比較されます。つまり、 二つのオブジェクトのインスタンスは、 同じ属性と値を有し、同じクラスのインスタンスである場合に、 等しいとされます。

一方、一致演算子(===)を使用する場合、 オブジェクト変数は、同じクラスの同じインスタンスを参照する場合のみ、 等しいとされます。

これらのルールを明確に示す例を以下に示します。

Example#1 PHP 5におけるオブジェクト比較の例

```php
<?php
function bool2str($bool)
{
    if ($bool === false) {
        return 'FALSE';
    } else {
        return 'TRUE';
    }
}

function compareObjects(&$o1, &$o2)
{
    echo 'o1 == o2 : ' . bool2str($o1 == $o2) . "¥n";
    echo 'o1 != o2 : ' . bool2str($o1 != $o2) . "¥n";
    echo 'o1 === o2 : ' . bool2str($o1 === $o2) . "¥n";
    echo 'o1 !== o2 : ' . bool2str($o1 !== $o2) . "¥n";
}

class Flag
{
    public $flag;

    function Flag($flag = true) {
        $this->flag = $flag;
    }
}

class OtherFlag
{
    public $flag;

    function OtherFlag($flag = true) {
        $this->flag = $flag;
    }
}

$o = new Flag();
$p = new Flag();
$q = $o;
$r = new OtherFlag();

echo "同一クラスの2つのインスタンス\n";
compareObjects($o, $p);

echo "\n同じインスタンスへの2つのリファレンス\n";
compareObjects($o, $q);

echo "\n2つの異なるクラスのインスタンス\n";
compareObjects($o, $r);
?>
```

上の例の出力は以下となります。

同一クラスの2つのインスタンス

```
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

同じインスタンスへの2つのリファレンス

```
o1 == o2 : TRUE
o1 != o2 : FALSE
o1 === o2 : TRUE
o1 !== o2 : FALSE
```

2つの異なるクラスのインスタンス

```
o1 == o2 : FALSE
o1 != o2 : TRUE
o1 === o2 : FALSE
o1 !== o2 : TRUE
```

> **注意**: 拡張モジュール内では、自前で作成したオブジェクトの比較方法を独自に定義することができます。

リフレクション

目次

はじめに

PHP 5には完全なリフレクション APIが付属しており、 クラス、インターフェイス、関数、メソッド、そしてエクステンションについて リバースエンジニアリングを行うことができます。 さらに、このリフレクション APIは関数、クラス、メソッドに 関するドキュメントコメントも取得することができます。

リフレクション APIは、Zend Engineのオブジェクト指向エクステンション で、以下のクラスから構成されています。

```php
<?php
class Reflection { }
interface Reflector { }
class ReflectionException extends Exception { }
class ReflectionFunction extends ReflectionFunctionAbstract implements Reflector { }
class ReflectionParameter implements Reflector { }
class ReflectionMethod extends ReflectionFunctionAbstract implements Reflector { }
class ReflectionClass implements Reflector { }
class ReflectionObject extends ReflectionClass { }
class ReflectionProperty implements Reflector { }
class ReflectionExtension implements Reflector { }
?>
```

> **注意**: これらのクラスに関する詳細については、次章を参照してください。

以下の例のコードを実行してみましょう。

Example#1 リフレクション APIの基本的な使用法

```php
<?php
Reflection::export(new ReflectionClass('Exception'));
?>
```

上の例の出力は以下となります。

```
Class [ <internal> class Exception ] {

  - Constants [0] {
  }

  - Static properties [0] {
  }

  - Static methods [0] {
  }

  - Properties [6] {
     Property [ <default> protected $message ]
     Property [ <default> private $string ]
```

```
    Property [ <default> protected $code ]
    Property [ <default> protected $file ]
    Property [ <default> protected $line ]
    Property [ <default> private $trace ]
  }

  - Methods [9] {
    Method [ <internal> final private method __clone ] {
    }

    Method [ <internal, ctor> public method __construct ] {

      - Parameters [2] {
        Parameter #0 [ <optional> $message ]
        Parameter #1 [ <optional> $code ]
      }
    }

    Method [ <internal> final public method getMessage ] {
    }

    Method [ <internal> final public method getCode ] {
    }

    Method [ <internal> final public method getFile ] {
    }

    Method [ <internal> final public method getLine ] {
    }

    Method [ <internal> final public method getTrace ] {
    }

    Method [ <internal> final public method getTraceAsString ] {
    }

    Method [ <internal> public method __toString ] {
    }
  }
}
```

Reflector

Reflector は、 すべてのエクスポート可能なリフレクションクラスが実装しているインターフェイスです。

```
<?php
interface Reflector
{
    public string __toString()
    public static string export()
}
?>
```

ReflectionException

ReflectionException は標準の Exception を継承しており、 リフレクション API によって投げられます。 固有のメソッドやプロパティは導入されていません。

ReflectionFunction

ReflectionFunctionクラスにより、 関数のリバースエンジニアリングが可能となります。

```
<?php
class ReflectionFunction extends ReflectionFunctionAbstract implements Reflector
{
    final private __clone()
    public void __construct(string name)
    public string __toString()
    public static string export(string name, bool return)
    public string getName()
    public bool isInternal()
    public bool isDisabled()
    public bool isUserDefined()
    public string getFileName()
    public int getStartLine()
    public int getEndLine()
    public string getDocComment()
    public array getStaticVariables()
    public mixed invoke([mixed args [, ...]])
    public mixed invokeArgs(array args)
    public bool returnsReference()
    public ReflectionParameter[] getParameters()
    public int getNumberOfParameters()
    public int getNumberOfRequiredParameters()
}
?>
```

親クラスであるReflectionFunctionAbstract にも、 **invoke()** および **invokeArgs()**、**export()**、 **isDisabled()** を除くすべてのメソッドが存在します。

> **注意**: **getNumberOfParameters()** と **getNumberOfRequiredParameters()** は PHP 5.0.3 で追加され、 **invokeArgs()** は PHP 5.1.0

で追加されました。

関数の内部を調べるために、まず、 ReflectionFunctionクラスのインスタンスを 生成する必要があります。 次にこのインスタンスの上のメソッドのどれかをコールすることができます。

Example#2 ReflectionFunctionクラスの使用法

```
<?php
/**
 * 簡単なカウンタ
 *
 * @return     int
 */
function counter()
{
    static $c = 0;
    return $c++;
}

// ReflectionFunction クラスのインスタンスを生成する
$func = new ReflectionFunction('counter');

// 基本情報を表示する
printf(
    "===> The %s function '%s'¥n".
    "     declared in %s¥n".
    "     lines %d to %d¥n",
    $func->isInternal() ? 'internal' : 'user-defined',
    $func->getName(),
    $func->getFileName(),
    $func->getStartLine(),
    $func->getEndline()
);

// ドキュメントコメントを表示する
printf("---> Documentation:¥n %s¥n", var_export($func->getDocComment(), 1));

// static 変数があれば表示する
if ($statics = $func->getStaticVariables())
{
    printf("---> Static variables: %s¥n", var_export($statics, 1));
}

// 関数を呼び出す
printf("---> Invokation results in: ");
var_dump($func->invoke());


// export() メソッドを使用する方が良い知れない
echo "¥nReflectionFunction::export() results:¥n";
echo ReflectionFunction::export('counter');
?>
```

注意: **invoke()**メソッドは、 call_user_func()と同様に 可変長の引数をとります。

ReflectionParameter

ReflectionParameterクラスは、 関数またはメソッドのパラメータに関する情報を取得します。

```
<?php
class ReflectionParameter implements Reflector
{
    final private __clone()
    public void __construct(string function, string parameter)
    public string __toString()
    public static string export(mixed function, mixed parameter, bool return)
    public string getName()
    public bool isPassedByReference()
    public ReflectionClass getDeclaringClass()
    public ReflectionClass getClass()
    public bool isArray()
    public bool allowsNull()
    public bool isPassedByReference()
    public bool isOptional()
    public bool isDefaultValueAvailable()
    public mixed getDefaultValue()
}
?>
```

注意: **getDefaultValue()**, **isDefaultValueAvailable()**, **isOptional()** は PHP 5.0.3 で追加され、 **isArray()** は PHP 5.1.0で追加されました。 **getDeclaringFunction()** および **getPosition()** は PHP 5.2.3 で追加されました。

関数パラメータの内部を調べる際には、まず、 ReflectionFunction クラスまたは ReflectionMethod クラスのインスタンスを 作成する必要があります。 次に、配列のパラメータを取得するために、そのインスタンスの **getParameters()**メソッドを使用してください。

Example#3 ReflectionParameterクラスの使用

```
<?php
function foo($a, $b, $c) { }
function bar(Exception $a, &$b, $c) { }
function baz(ReflectionFunction $a, $b = 1, $c = null) { }
function abc() { }
```

```
// コマンドラインから与えられたパラメータを使って
// ReflectionFunction のインスタンスを生成する
$reflect = new ReflectionFunction($argv[1]);

echo $reflect;

foreach ($reflect->getParameters() as $i => $param) {
    printf(
        "-- Parameter #%d: %s {¥n".
        "    Class: %s¥n".
        "    Allows NULL: %s¥n".
        "    Passed to by reference: %s¥n".
        "    Is optional?: %s¥n".
        "}¥n",
        $i,
        $param->getName(),
        var_export($param->getClass(), 1),
        var_export($param->allowsNull(), 1),
        var_export($param->isPassedByReference(), 1),
        $param->isOptional() ? 'yes' : 'no'
    );
}
?>
```

ReflectionClass

ReflectionClassクラスにより、 クラスやインターフェイスのリバースエンジニアリングが可能となります。

```
<?php
class ReflectionClass implements Reflector
{
     final private __clone()
     public void __construct(string name)
     public string __toString()
     public static string export()
     public string getName()
     public bool isInternal()
     public bool isUserDefined()
     public bool isInstantiable()
     public bool hasConstant(string name)
     public bool hasMethod(string name)
     public bool hasProperty(string name)
     public string getFileName()
     public int getStartLine()
     public int getEndLine()
     public string getDocComment()
     public ReflectionMethod getConstructor()
     public ReflectionMethod getMethod(string name)
     public ReflectionMethod[] getMethods()
     public ReflectionProperty getProperty(string name)
     public ReflectionProperty[] getProperties()
     public array getConstants()
     public mixed getConstant(string name)
     public ReflectionClass[] getInterfaces()
     public bool isInterface()
     public bool isAbstract()
     public bool isFinal()
     public int getModifiers()
     public bool isInstance(stdclass object)
     public stdclass newInstance(mixed args)
     public stdclass newInstanceArgs(array args)
     public ReflectionClass getParentClass()
     public bool isSubclassOf(ReflectionClass class)
     public array getStaticProperties()
     public mixed getStaticPropertyValue(string name [, mixed default])
     public void setStaticPropertyValue(string name, mixed value)
     public array getDefaultProperties()
     public bool isIterateable()
     public bool implementsInterface(string name)
     public ReflectionExtension getExtension()
     public string getExtensionName()
}
?>
```

> **注意**: **hasConstant()**, **hasMethod()**, **hasProperty()**, **getStaticPropertyValue()** および **setStaticPropertyValue()** は、PHP 5.1.0 で追加されました。 また、**newInstanceArgs()** は PHP 5.1.3 で追加されました。

クラスのイントロスペクションを行うには、まず ReflectionClass クラスのインスタンスを生成する必要があります。それから、 このインスタンスのメソッドをコールしてください。

Example#4 ReflectionClassクラスの使用法

```
<?php
interface Serializable
{
    // ...
}

class Object
{
    // ...
}

/**
 * カウンタクラス
 */
```

```
class Counter extends Object implements Serializable
{
     const START = 0;
     private static $c = Counter::START;

    /**
     * カウンタを呼び出す
     *
      * @access  public
      * @return  int
      */
    public function count() {
         return self::$c++;
    }
}

// ReflectionClass クラスのインスタンスを生成する
$class = new ReflectionClass('Counter');

// 基本情報を表示する
printf(
    "===> The %s%s%s %s '%s' [extends %s]¥n" .
    "      declared in %s¥n" .
    "      lines %d to %d¥n" .
    "      having the modifiers %d [%s]¥n",
        $class->isInternal() ? 'internal' : 'user-defined',
        $class->isAbstract() ? ' abstract' : '',
        $class->isFinal() ? ' final' : '',
        $class->isInterface() ? 'interface' : 'class',
        $class->getName(),
        var_export($class->getParentClass(), 1),
        $class->getFileName(),
        $class->getStartLine(),
        $class->getEndline(),
        $class->getModifiers(),
        implode(' ', Reflection::getModifierNames($class->getModifiers()))
);

// ドキュメントコメントを表示する
printf("---> Documentation:¥n %s¥n", var_export($class->getDocComment(), 1));

// このクラスが実装しているインターフェースを表示する
printf("---> Implements:¥n %s¥n", var_export($class->getInterfaces(), 1));

// クラス定数を表示する
printf("---> Constants: %s¥n", var_export($class->getConstants(), 1));

// クラスプロパティを表示する
printf("---> Properties: %s¥n", var_export($class->getProperties(), 1));

// クラスメソッドを表示する
printf("---> Methods: %s¥n", var_export($class->getMethods(), 1));

// このクラスがインスタンス化可能な場合、インスタンスを生成する
if ($class->isInstantiable()) {
    $counter = $class->newInstance();

    echo '---> $counter is instance? ';
    echo $class->isInstance($counter) ? 'yes' : 'no';

    echo "¥n---> new Object() is instance? ";
    echo $class->isInstance(new Object()) ? 'yes' : 'no';
}
?>
```

注意: **newInstance()**メソッドは、 call_user_func()と同様に 可変長の引数をとります。

注意: *$class = new ReflectionClass('Foo'); $class->isInstance($arg)* は、*$arg instanceof Foo* または *is_a($arg, 'Foo')*と等価です。

ReflectionObject

ReflectionObject クラスにより、 オブジェクトのリバースエンジニアリングが可能となります。

```
<?php
class ReflectionObject extends ReflectionClass
{
     final private __clone()
     public void __construct(mixed object)
     public string __toString()
     public static string export(mixed object, bool return)
}
?>
```

ReflectionMethod

ReflectionMethodクラスにより、 クラスメソッドのリバースエンジニアリングが可能となります。

```
<?php
class ReflectionMethod extends ReflectionFunctionAbstract implements Reflector
{
     public void __construct(mixed class, string name)
     public string __toString()
     public static string export(mixed class, string name, bool return)
     public mixed invoke(stdclass object [, mixed args [, ...]])
     public mixed invokeArgs(stdclass object, array args)
     public bool isFinal()
```

```
     public bool  isAbstract()
     public bool  isPublic()
     public bool  isPrivate()
     public bool  isProtected()
     public bool  isStatic()
     public bool  isConstructor()
     public bool  isDestructor()
     public int  getModifiers()
     public ReflectionClass  getDeclaringClass()

    //  ReflectionFunctionAbstract  から継承したメソッド
     final  private  __clone()
     public string  getName()
     public bool  isInternal()
     public bool  isUserDefined()
     public string  getFileName()
     public int  getStartLine()
     public int  getEndLine()
     public string  getDocComment()
     public array getStaticVariables()
     public bool  returnsReference()
     public ReflectionParameter[] getParameters()
     public int  getNumberOfParameters()
     public int  getNumberOfRequiredParameters()
}
?>
```

メソッドの内部を調べるために、まず、 ReflectionMethodクラスのインスタンスを 生成する必要があります。 次にこのインスタンスの上のメソッドのどれかをコールすることができます。

Example#5 ReflectionMethodクラスの使用

```
<?php
class  Counter
{
     private static $c = 0;

    /**
     * カウンタをインクリメントする
     *
     * @final
     * @static
     * @access   public
     * @return   int
     */
    final  public  static  function  increment()
    {
         return ++self::$c;
    }
}

// ReflectionMethod  クラスのインスタンスを生成する
$method  = new ReflectionMethod('Counter', 'increment');

// 基本情報を表示する
printf(
    "===>  The  %s%s%s%s%s%s%s  method  '%s'  (which  is  %s)¥n"  .
    "       declared in %s¥n" .
    "       lines %d to %d¥n" .
    "       having  the  modifiers  %d[%s]¥n",
        $method->isInternal() ? 'internal'  : 'user-defined',
        $method->isAbstract() ? '  abstract'  : '',
        $method->isFinal() ? ' final'  : '',
        $method->isPublic() ? '  public'  : '',
        $method->isPrivate() ? '  private'  : '',
        $method->isProtected() ? '  protected'  : '',
        $method->isStatic() ? '  static'  : '',
        $method->getName(),
        $method->isConstructor() ? 'the  constructor'  : 'a  regular  method',
        $method->getFileName(),
        $method->getStartLine(),
        $method->getEndline(),
        $method->getModifiers(),
        implode(' ', Reflection::getModifierNames($method->getModifiers()))
);

// ドキュメントコメントを表示する
printf("--->  Documentation:¥n  %s¥n", var_export($method->getDocComment(), 1));

// static  変数があれば表示する
if ($statics= $method->getStaticVariables()) {
    printf("--->  Static  variables:  %s¥n", var_export($statics, 1));
}

// メソッドを呼び出す
printf("--->  Invokation  results  in:  ");
var_dump($method->invoke(NULL));
?>
```

> **注意**: private, protectedまたはabstractメソッドのinvokeを行うと、 **invoke()**から例外がスローされます。

> **注意**: 上記のstaticメソッドの場合、 **invoke()**の最初の引数にNULLを渡す必要があります。 staticでないメソッドの場合、ここにクラスのインスタンスを指定してください。

ReflectionProperty

ReflectionPropertyクラスにより、 クラスプロパティに関する リバースエンジニアリングが可能となります。

```
<?php
class ReflectionProperty implements Reflector
{
     final private __clone()
     public void __construct(mixed class, string name)
     public string __toString()
     public static string export(mixed class, string name, bool return)
     public string getName()
     public bool isPublic()
     public bool isPrivate()
     public bool isProtected()
     public bool isStatic()
     public bool isDefault()
     public int getModifiers()
     public mixed getValue(stdclass object)
     public void setValue(stdclass object, mixed value)
     public ReflectionClass getDeclaringClass()
}
?>
```

プロパティの内部を調べるために、まず、 ReflectionPropertyクラスのインスタンスを 生成する必要があります。 次にこのインスタンスの上のメソッドのどれかをコールすることができます。

Example#6 ReflectionPropertyクラスの使用

```
<?php
class String
{
     public $length  = 5;
}

// ReflectionProperty クラスのインスタンスを生成する
$prop = new ReflectionProperty('String', 'length');

// 基本情報を表示する
printf(
     "===> The%s%s%s%s property '%s' (which was %s)¥n" .
     "       having the modifiers %s¥n",
          $prop->isPublic() ? ' public' : '',
          $prop->isPrivate() ? ' private' : '',
          $prop->isProtected() ? ' protected' : '',
          $prop->isStatic() ? ' static' : '',
          $prop->getName(),
          $prop->isDefault() ? 'declared at compile-time' : 'created at run-time',
          var_export(Reflection::getModifierNames($prop->getModifiers()), 1)
);

// String のインスタンスを生成する
$obj= new String();

// 現在の値を取得する
printf("---> Value is: ");
var_dump($prop->getValue($obj));

// 値を変更する
$prop->setValue($obj, 10);
printf("---> Setting value to 10, new value is: ");
var_dump($prop->getValue($obj));

// オブジェクトをダンプする
var_dump($obj);
?>
```

注意: privateまたはprotectedクラスプロパティの値の取得または設定を 行うと、例外がスローされます。

ReflectionExtension

The ReflectionExtensionクラスにより、 エクステンションのリバースエンジニアリングが可能となります。 実行時にロードされている全てのエクステンションを get_loaded_extensions()により取得することができます。

```
<?php
class ReflectionExtension implements Reflector {
     final private __clone()
     public void __construct(string name)
     public string __toString()
     public static string export(string name, bool return)
     public string getName()
     public string getVersion()
     public ReflectionFunction[] getFunctions()
     public array getConstants()
     public array getINIEntries()
     public ReflectionClass[] getClasses()
     public array getClassNames()
     public string info()
}
?>
```

メソッドの内部を調べるために、まず、 ReflectionExtensionクラスのインスタンスを 生成する必要があります。 次にこのインスタンスの上のメソッドのどれかをコールすることができます。

Example#7 ReflectionExtensionクラスの使用

```php
<?php
// ReflectionProperty クラスのインスタンスを生成する
$ext = new ReflectionExtension('standard');

// 基本情報を表示する
printf(
    "Name          : %s¥n" .
    "Version       : %s¥n" .
    "Functions     : [%d] %s¥n" .
    "Constants     : [%d] %s¥n" .
    "INI entries   : [%d] %s¥n" .
    "Classes       : [%d] %s¥n",
        $ext->getName(),
        $ext->getVersion() ? $ext->getVersion() : 'NO_VERSION',
        sizeof($ext->getFunctions()),
        var_export($ext->getFunctions(), 1),

        sizeof($ext->getConstants()),
        var_export($ext->getConstants(), 1),

        sizeof($ext->getINIEntries()),
        var_export($ext->getINIEntries(), 1),

        sizeof($ext->getClassNames()),
        var_export($ext->getClassNames(), 1)
);
?>
```

reflectionクラスを拡張する

組み込みクラスの特別なバージョンを作成したい場合 (例えば、、エクスポートする際に、色づけしたHTMLを作成したり、 メソッドの代わりに簡単にアクセスできるメンバー変数を作成したり、 補助的なメソッドを作成したり、)、 Reflectionクラスを拡張することができます。

Example#8 組み込みクラスを拡張する

```php
<?php
/**
 * 独自の Reflection_Method クラス
 */
class My_Reflection_Method extends ReflectionMethod
{
    public $visibility = array();

    public function __construct($o, $m)
    {
        parent::__construct($o, $m);
        $this->visibility = Reflection::getModifierNames($this->getModifiers());
    }
}

/**
 * デモクラス #1
 *
 */
class T {
    protected function x() {}
}

/**
 * デモクラス #2
 *
 */
class U extends T {
    function x() {}
}

// 基本情報を表示する
var_dump(new My_Reflection_Method('U', 'x'));
?>
```

注意: 注意: Iコンストラクタを上書きした場合、 挿入するコードの前に 親クラスのコンストラクタをコールしわすれないようにしてください。 これを怠ると、以下のようなエラーを発生します。 *Fatal error: Internal error: Failed to retrieve the reflection object*

タイプヒンティング

PHP 5では、タイプヒンティング(Type Hinting)が導入されました。 これにより、関数は、 (クラスの名前を関数プロトタイプの中に指定することにより) パラメータをオブジェクトもしくは配列 (PHP5.1以降) が必ず指定されるようにすることができるようになりました。

Example#1 タイプヒンティングの例

```php
<?php
// とあるクラス
class MyClass
{
    /**
     * テスト関数
     *
     * 第 1 引数は OtherClass 型のオブジェクトでなければならない
     */
    public function test(OtherClass $otherclass) {
        echo $otherclass->var;
```

```
    }

    /**
     * もう一つのテスト関数
     *
     * 第 1 引数は配列でなければならない
     */
    public function test_array(array $input_array) {
        print_r($input_array);
    }
}

// もう一つのサンプルクラス
class OtherClass {
    public $var = 'Hello World';
}
?>
```

タイプヒントの指定を満たさないとキャッチ可能な致命的エラーとなります。

```
<?php
// それぞれのクラスのインスタンス
$myclass = new MyClass;
$otherclass = new OtherClass;

// Fatal Error: Argument 1 must be an object of class OtherClass
$myclass->test('hello');

// Fatal Error: Argument 1 must be an instance of OtherClass
$foo = new stdClass;
$myclass->test($foo);

// Fatal Error: Argument 1 must not be null
$myclass->test(null);

// Works: Prints Hello World
$myclass->test($otherclass);

// Fatal Error: Argument 1 must be an array
$myclass->test_array('a string');

// 動作する: 配列の内容を表示する
$myclass->test_array(array('a', 'b', 'c'));
?>
```

タイプヒンティングは、関数でも使用できます。

```
<?php
// とあるクラス
class MyClass {
    public $var = 'Hello World';
}

/**
 * テスト関数
 *
 * 第 1 引数は MyClass 型のオブジェクトでなければならない
 */
function MyFunction (MyClass $foo) {
    echo $foo->var;
}

// 動作する
$myclass = new MyClass;
MyFunction($myclass);
?>
```

タイプヒントは、object型やarray型 (PHP5.1以降) でのみ使用できます。 int および stringのような 通常の型でのタイプヒンティングはサポートされません。

遅延静的束縛 (Late Static Bindings)

PHP 5.3.0 以降、PHP に遅延静的束縛と呼ばれる機能が搭載されます。 これを使用すると、静的継承のコンテキストで呼び出し元のクラスを参照できるようになります。

この "遅延静的束縛" という機能名は、内部動作を考慮してつけられたものです。 "遅延束縛 (Late binding)" の由来は、メソッドを定義しているクラス名を使用しても *static::* の解決ができなくなったことによります。 その代わりに、実行時情報をもとに解決するようになります。 "静的束縛 (static binding)" の由来は、 静的メソッドのコールに使用できることによります (ただし、静的メソッド以外でも使用可能です)。

self:: の制限

self:: あるいは *__CLASS__* による現在のクラスへの静的参照は、 そのメソッドが属するクラス (つまり、 そのメソッドが定義されているクラス) に解決されます。

Example#1 *self::* の使用例

```
<?php
class A {
```

```
     public static function who() {
         echo __CLASS__;
     }
     public static function test() {
         self::who();
     }
}

class B extends A {
     public static function who() {
          echo __CLASS__;
     }
}

B::test();
?>
```

上の例の出力は以下となります。

```
A
```

遅延静的束縛の使用法

遅延静的束縛は、この制限を解決するためのキーワードを導入し、 実行時に最初にコールされたクラスを参照するようにしています。 このキーワードを使用すると、先ほどの例における *test()* から *B* を参照できるようになります。 このキーワードは新たに追加したものではなく、すでに予約済みである *static* を使用しています。

Example#2 *static::* のシンプルな使用法

```
<?php
class A {
     public static function who() {
         echo __CLASS__;
     }
     public static function test() {
         static::who(); //  これで、遅延静的束縛が行われます
     }
}

class B extends A {
     public static function who() {
          echo __CLASS__;
     }
}

B::test();
?>
```

上の例の出力は以下となります。

```
B
```

> **注意**: *static::* の動作は、静的メソッドにおいては *$this* と異なります! *$this->* は継承規則に従いますが、 *static::* は従いません。 この違いについては、後ほど詳しく説明します。

Example#3 非静的コンテキストにおける *static::* の使用法

```
<?php
class TestChild extends TestParent {
     public function __construct() {
         static::who();
     }

     public function test() {
         $o = new TestParent();
     }

     public static function who() {
         echo __CLASS__."¥n";
     }
}

class TestParent {
     public function __construct() {
         static::who();
     }

     public static function who() {
         echo __CLASS__."¥n";
     }
}
$o = new TestChild;
$o->test();

?>
```

上の例の出力は以下となります。

```
TestChild
TestParent
```

注意: 遅延静的束縛の解決は、静的コールが代替なしに完全に解決された時点で終了します。

Example#4 完全に解決された静的コール

```
<?php
class A {
      public static function foo() {
          static::who();
      }

      public static function who() {
          echo __CLASS__."¥n";
      }
}

class B extends A {
      public static function test() {
          A::foo();
      }

      public static function who() {
          echo __CLASS__."¥n";
      }
}

B::test();
?>
```

上の例の出力は以下となります。

```
A
```

特殊な場合

PHP でメソッドをコールするには、さまざまな方法があります。 たとえばコールバックやマジックメソッドなどもそのひとつです。 遅延静的束縛は実行時の情報にもとづいて解決を行うので、 このように特殊な場合には予期せぬ結果となる可能性があります。

Example#5 マジックメソッド内における遅延静的束縛

```
<?php
class A {

    protected static function who() {
        echo __CLASS__."¥n";
    }

    public function __get($var) {
        return static::who();
    }
}

class B extends A {

    protected static function who() {
        echo __CLASS__."¥n";
    }
}

$b = new B;
$b->foo;
?>
```

上の例の出力は以下となります。

```
B
```

名前空間

目次

- 名前解決の規則

名前空間の概要

PHP における名前空間は、PHP のライブラリが巨大化したときの スコープの問題を解決するために設計されています。 PHP では、すべてのクラス定義はグローバルです。 したがって、ライブラリの作者がさまざまな公開 API を作成する際には、 他のライブラリの同様の機能との競合に注意する必要があります。 お互いに名前が重複しないよう、一意な名前を選択する必要があるのです。 このとき、通常はクラス名を先頭に付加することで一意な名前を作成します。 たとえばデータベースのクラスなら、その先頭に My_Library_DB を付加するなどといったことです。 ライブラリが巨大化するにつれて、このプレフィックスもどんどん追加され、 非常に長ったらしい名前になってしまいます。

名前空間を使用すると、そのクラスを参照する際に毎回長い名前を使用する必要がなくなります。 コードの可読性を保ったまま、グローバル空間の共用の問題を解決することができます。

名前空間は、PHP 5.3.0 以降で使用可能です。 このセクションの内容は実験的なものであり、変更される可能性があります。

名前空間の定義

名前空間の宣言は、*namespace* キーワードを用いて行います。これを、ファイルの先頭に記述しなければなりません。 たとえば次のようになります。

Example#1 名前空間の定義

```
<?php
      namespace MyProject::DB;

     const CONNECT_OK  = 1;

     class Connection  { /* ... */ }

     function connect() { /* ... */  }

?>
```

同一の名前空間を、複数のファイルで使用することもできます。

名前空間の中にはクラスや定数、関数定義を含めることができます。 ただしそれら以外のコードを含めることはできません。

名前空間の定義は次のようなものです。

- 名前空間の内部では、すべてのクラスや関数、 定数名には自動的に名前空間名のプレフィックスが付加されます。 クラス名は常にフルネームとなります。つまり、 上の例のクラスをコールする際は MyProject::DB::Connection とします。
- 定数を定義すると、名前空間名と定数名を組み合わせた定数を作成します。 クラス定数と同様、名前空間定数にも静的な値しか保持できません。
- 修飾されていないクラス名 (:: を含まない名前) は、実行時に次の手順で解決されます。
    1. そのクラスを、現在の名前空間から (つまり現在の名前空間名を先頭につけて) 探します。 その際には autoload を試みません。
    2. グローバル名前空間から、autoload を試みずにそのクラスを探します。
    3. 現在の名前空間で autoload を試みます。
    4. 以上がすべて失敗した場合は、クラスの検索が失敗します。
- 修飾されていない関数名 (:: を含まない名前) は、実行時にまず現在の名前空間で探され、 次にグローバル空間で探します。
- 修飾されていない定数名は、 まず現在の名前空間で探され、次にグローバル空間で探します。

完全な 名前解決の規則 も参照ください。

名前空間の使用法

名前空間内のすべてのクラスや関数は、どこからでも MyProject::DB::Connection や MyProject::DB::connect のようにフルネームで参照することができます。

Example#1 名前空間内の名前の使用法

```
<?php
    require 'MyProject/Db/Connection.php';
    $x = new MyProject::DB::Connection;
    MyProject::DB::connect();
?>
```

名前空間を現在のコンテキスト (グローバル空間あるいは別の名前空間) にインポートするには *use* 演算子を使用します。 この演算子の構文は、次のようになります。

```
<?php
/* ... */
use Some::Name as Othername;
```

```
// シンプルな使用法
use Foo::Bar;
// これは、さきほどと同じ意味になります
use Foo::Bar as Bar;
?>
```

インポートされた名前の働きは、次のようになります。 コンパイラがローカル名 *Othername* (それ単体か、あるいは :: で区切られた長い名前へのプレフィックス) に遭遇すると、インポートされた名前 *Some::Name* でそれを置き換えます。

use はグローバルスコープでのみ使用可能です。 関数やクラスの内部では使用できません。 インポートされた名前が有効なのは、 インポートした箇所からそのファイルの最後までの間です。 混乱を避けるため、インポートはファイルの先頭で行うようにしましょう。

Example#2 名前空間のインポート、名前空間へのアクセス

```
<?php
    require 'MyProject/Db/Connection.php';
    use MyProject::DB;
    use MyProject::DB::Connection as DbConnection;

    $x = new MyProject::DB::Connection();
    $y = new DB::connection();
    $z = new DbConnection();
    DB::connect();
?>
```

> **注意**: インポート処理はコンパイル時にのみ行われ、 すべてのローカル名はコンパイラによって完全な名前に変換されます。 この変換は文字列で行うわけではないので、 コールバック関数の指定がインポートの影響を受けることはありません。

グローバル空間

名前空間を定義しない場合は、すべてのクラスや関数の定義はグローバル空間に配置されます。 これは、名前空間をサポートする前のバージョンの PHP と同じ状態です。 名前の先頭に :: をつけると、 名前空間のコンテキストでも明示的にグローバル空間を指定することができます。

Example#1 グローバル空間の指定

```
<?php
     namespace A::B::C;

 /* この関数は A::B::C::fopen です */
    function fopen() {
         /* ... */
        $f = ::fopen(...); // グローバル空間の fopen をコールします
         return $f;
    }
?>
```

__NAMESPACE__

コンパイル時に、現在の名前空間を表す定数 **__NAMESPACE__** が定義されます。 名前空間の外部では、この定数の値は空文字列となります。 この定数は、名前空間内のローカル名から完全な名前を作成する際に便利です。

Example#1 __NAMESPACE__ の使用法

```
<?php
namespace A::B::C;

function foo() {
// 何かの処理
}

set_error_handler(__NAMESPACE__ . "::foo");
?>
```

名前解決の規則

名前解決は、以下の規則に基づいて行います。

1. すべての修飾名が、コンパイル時に import の指定にあわせて変換されます。 たとえば、名前空間 A::B::C をインポートした環境で `C::D::e()` をコールすると、それが `A::B::C::D::e()` に変換されます。
2. 修飾されていないクラス名が、コンパイル時に import の指定にあわせて変換されます (短い名前を完全な名前に変換します)。 たとえば、名前空間 *A::B::C* をインポートした環境では `new C()` が `new A::B::C()` に変換されます。
3. 名前空間の中で、修飾されていない関数コールのうち、現在の名前空間で定義されているもの (そしてその関数コールをパースする時点で既知のもの) については、 現在の名前空間の関数に変換されます。
4. 名前空間 (たとえば A::B) の中で、修飾されていない関数コールのうち、 現在の名前空間では定義されていないものについては実行時に解決されます。 関数 *foo()* のコールは、次のように解決されます。
    1. 現在の名前空間の関数 *A::B::foo()* を探します。

    2. 内部関数 *foo()* を探します。

   グローバル名前空間にあるユーザ定義の関数をコールするには、 *::foo()* とする必要があります。

5. 名前空間 (たとえば A::B) の中で、 修飾されていないクラス名については実行時に解決されます。 `new C()` のコールは、次のように解決されます。
    1. 現在の名前空間のクラス *A::B::C()* を探します。
    2. 内部クラス *C()* を探します。
    3. *A::B::C()*、 *C()* の autoload を試みます。

   グローバル名前空間にあるユーザ定義のクラスを参照するには `new ::C()` とする必要があります。
6. 修飾されている関数コールを実行時に解決します。 *A::B::foo()* のコールは、次のように解決されます。
    1. 名前空間 *A::B* の関数 *foo()* を探します。
    2. クラス *A::B* を探し、その静的メソッド *foo()* をコールします。 必要に応じてクラスを autoload します。
7. 修飾されているクラス名を、コンパイル時にその名前空間のクラスとして解決します。 たとえば `new A::B::C()` は、名前空間 *A::B* のクラス C を指します。

Example#1 名前解決の解説

```
<?php
namespace A;

// 関数コール

foo();      // まずは名前空間 "A" の関数 "foo" のコールを試み、
            // それがなければ内部関数 "foo" をコールします

::foo();    // グローバルスコープで定義されている関数 "foo" をコールします

// クラスの参照

new B();    // まずは名前空間 "A" で定義されているクラス "B" のオブジェクトの作成を試み、
            // それがなければ内部クラス "B" のオブジェクトを作成します

new ::B();  // グローバルスコープで定義されているクラス "B" のオブジェクトをします

// 他の名前空間の静的メソッド/関数

B::foo();   // まずは名前空間 "A::B" の関数 "foo" のコールを試み、
            // 次に内部クラス "B" のメソッド "foo" をコールします

::B::foo(); // まずは名前空間 "B" の関数 "foo" のコールを試み、
            // 次にグローバルスコープのクラス "B" のメソッド "foo" をコールします

// 現在の名前空間の静的メソッド/関数

A::foo();   // まずは名前空間 "A::A" の関数 "foo" のコールを試み、
            // 次に名前空間 "A" のクラス "A" のメソッド "foo"、
            // それから名前空間 "A" の関数 "foo"、
            // さらにその次に内部クラス "A" のメソッド "foo" という順になります

::A::foo(); // まずは名前空間 "A" の関数 "foo" のコールを試み、次に
            // グローバルスコープのクラス "A" のメソッド "foo" をコールします
?>
```

例外(exceptions)

PHP 5 は、他のプログラミング言語に似た例外モデルを有しています。 PHP 内で例外が投げられ ("*throw*" され)、それが 捕捉され ("*catch*" され) ます。発生した例外を 捕捉するには、コードを *try* ブロックで囲みます。 各 *try* ブロックには、対応する *catch* ブロックが存在する必要があります。異なる型の例外を捕捉するために 複数の *catch* フロックを使用することができます。 通常の実行時 (*try* ブロック内で例外が投げられなかった 場合、あるいは投げられた例外に対応する *catch* ブロックが存在しなかった場合) は、*catch* ブロック内は処理されず、それ以降から処理が続けられます。 *catch* ブロックの中から例外を投げる (あるいは投げなおす) こともできます。

例外が投げられた場合、その命令に続くコードは実行されず、 PHP は最初にマッチする *catch* ブロックを探します。 例外が捕捉されない場合、PHP は "*Uncaught Exception ...*" というメッセージとともに 致命的なエラー(fatal error)を発行します。 ただし、set_exception_handler() でハンドラが定義されている場合を除きます。

Example#1 例外を投げるには

```
<?php
function inverse($x) {
    if (!$x) {
        throw new Exception('ゼロによる除算。');
    }
    else return 1/$x;
}

try {
    echo inverse(5) . "¥n";
    echo inverse(0) . "¥n";
} catch (Exception $e) {
    echo '捕捉した例外: ', $e->getMessage(), "¥n";
}

// 実行は継続される
echo 'Hello World';
```

```
?>
```

上の例の出力は以下となります。

```
0.2
捕捉した例外: ゼロによる除算。
Hello World
```

例外を拡張する

組み込みの Exception クラスを拡張することで、例外クラスをユーザーが 定義することが可能です。以下のメンバーおよびプロパティは、 組み込みの Exception クラスから派生した子クラスの中でアクセス可能です。

Example#2 例外クラスを構築する

```
<?php
class Exception
{
     protected $message = 'Unknown exception';   // exception message
    protected $code = 0;                         // user defined exception code
    protected $file;                             // source filename of exception
    protected $line;                             // source line of exception

    function __construct($message = null, $code = 0);

     final function getMessage();                // message of exception
    final function getCode();                    // code of exception
    final function getFile();                    // source filename
    final function getLine();                    // source line
    final function getTrace();                   // an array of the backtrace()
    final function getTraceAsString();           // formated string of trace

     /* Overrideable */
    function __toString();                        // formated string for display
}
?>
```

クラスが、組み込みの Exception クラスを拡張し、 コンストラクタを再定義した場合、 全ての利用可能なデータが正しく代入されることを保証するために parent::__construct() もコールすることが強く推奨されます。 __toString() メソッドは、 オブジェクトが文字列として表された際に独自の出力を行うために 上書きすることができます。

Example#3 例外クラスを拡張する

```
<?php
/**
 * カスタム例外クラスを定義する
 */
class MyException extends Exception
{
    // 例外を再定義し、メッセージをオプションではなくする
    public function __construct($message, $code = 0) {
        // なんらかのコード

        // 全てを正しく確実に代入する
        parent::__construct($message, $code);
    }

    // オブジェクトの文字列表現を独自に定義する
    public function __toString() {
        return __CLASS__ . ": [{$this->code}]: {$this->message}¥n";
    }

     public function customFunction() {
        echo "A Custom function for this type of exception¥n";
    }
}


/**
 * 例外をテストするためのクラスを作成
 */
class TestException
{
     public $var;

     const THROW_NONE     = 0;
     const THROW_CUSTOM   = 1;
     const THROW_DEFAULT  = 2;

     function __construct($avalue = self::THROW_NONE) {

         switch ($avalue) {
             case self::THROW_CUSTOM:
                //  カスタム例外を投げる
                 throw new MyException('1 is an invalid parameter', 5);
                 break;

             case self::THROW_DEFAULT:
                //  デフォルト例外を投げる
                 throw new Exception('2 isnt allowed as a parameter', 6);
                 break;
```

```
            default:
              // 例外なし。オブジェクトが生成される
                $this->var = $avalue;
                 break;
        }
    }
}

// 例1
try {
    $o = new TestException(TestException::THROW_CUSTOM);
} catch (MyException $e) {       // Will be caught
    echo "Caught my exception¥n", $e;
    $e->customFunction();
} catch (Exception $e) {         // Skipped
    echo "Caught Default Exception¥n", $e;
}

// 実行を継続する
var_dump($o);
echo "¥n¥n";


// 例2
try {
    $o = new TestException(TestException::THROW_DEFAULT);
} catch (MyException $e) {       // この型にはマッチしない
    echo "Caught my exception¥n", $e;
    $e->customFunction();
} catch (Exception $e) {         // キャッチされる
    echo "Caught Default Exception¥n", $e;
}

// 実行を継続する
var_dump($o);
echo "¥n¥n";


// 例3
try {
    $o = new TestException(TestException::THROW_CUSTOM);
} catch (Exception $e) {         // キャッチされる
    echo "Default Exception caught¥n", $e;
}

// 実行を継続する
var_dump($o);
echo "¥n¥n";


// 例4
try {
    $o = new TestException();
} catch (Exception $e) {         // スキップされる、例外なし
    echo "Default Exception caught¥n", $e;
}

// 実行を継続する
var_dump($o);
echo "¥n¥n";
?>
```

リファレンスの説明

目次

リファレンスとは?

PHP において、リファレンスとは同じ変数の内容を異なった名前で コールすることを意味します。これは C のポインタのようなものではなく、 シンボルテーブルのエイリアスです。PHP では、変数名と変数の内容は異なっており、 このため、同じ内容は異なった複数の名前を有する事が可能であることに 注意してください。最も良く似ているのは、Unix のファイル名とファイルの 関係です。この場合、変数名はディレクトリエントリ、変数の内容はファイル自体に対応します。リファレンスは、Unix ファイルシステムの ハードリンクのようなものであると考えられます。

リファレンスが行うことは何ですか?

PHP のリファレンスにより二つの変数が同じ内容を参照することが可能です。 つまり、以下のようなものを実行した場合です。

```
<?php
$a =& $b;
?>
```

この場合、*$a* と *$b* は同じ内容を 指します。

> **注意**: ここで、*$a* と *$b* は完全に 同じで、*$a* が *$b* を 指しているわけではなく、その逆でもありません。*$a* と *$b* は同じ場所を指しているのです。
>
> **注意**: リファレンスを含む配列をコピーする際に、そのリファレンスが解消される ことはありません。配列を関数に値渡しする場合も同様です。
>
> **注意**: 未定義の変数のリファレンスに対して代入したり 渡したり返したりすると、そこで変数が作成されます。
>
> **Example#1 未定義の変数のリファレンスの使用**
>
> ```
> <?php
> function foo(&$var) { }
>
> foo($a); // $a が作成され、null が代入されます
>
> $b = array();
> foo($b['b']);
> var_dump(array_key_exists('b', $b)); // bool(true)
>
> $c = new StdClass;
> foo($c->d);
> var_dump(property_exists($c, 'd')); // bool(true)
> ?>
> ```

リファレンスを返す関数や *new* 演算子でも 同じ構文が使用可能です（PHP 4.0.4 以降）。

```
<?php
$bar =& new fooclass();
$foo =& find_var($bar);
?>
```

PHP 5 以降、new は自動的にリファレンスを返すようになりました。そのため、この場面で *=&* を使用することは非推奨となり、 E_STRICT レベルのメッセージが表示されるようになりました。

> **注意**: *&* 演算子を使用しない場合は、オブジェクトのコピーが 作成されます。クラスの内部で *$this* を使用した場合、 それはクラスの現在のインスタンスに対する操作を表します。 *&* のない代入はインスタンス（オブジェクト）のコピーを 行い、*$this* はそのコピーに対する操作を表します。 これはお望みの動作と異なるかもしれません。パフォーマンスやメモリ使用量の 観点から、常に単一のインスタンスに対して操作を行いたくなることもあるでしょう。
> コンストラクタ内で *@new* のようにして *@* 演算子を使用すると、あらゆるエラーの表示を *見えなくすること*が可能ですが、 *&new* を使用する場合にはこの機能は動作しません。 Zend Engine の仕様により、これはパースエラーとなります。

警告

関数の内部で *global* 宣言された変数にリファレンスを 代入すると、そのリファレンスは関数の内部でのみ参照可能となります。 これを避けるには、*$GLOBALS* 配列を使用します。

Example#2 関数内でのグローバル変数の参照

```
<?php
$var1 = "Example variable";
$var2 = "";

function global_references($use_globals)
{
    global $var1, $var2;
    if (!$use_globals) {
        $var2 =& $var1; // 関数の内部でのみ参照可能
    } else {
        $GLOBALS["var2"] =& $var1; // 関数の外部でも参照可能
    }
}

global_references(false);
echo "var2 の値は '$var2'\n"; // var2 の値は ''
global_references(true);
echo "var2 の値は '$var2'\n"; // var2 の値は 'Example variable'
?>
```

global $var;* は、*$var =& $GLOBALS['var'];* の短縮版だと考えてください。 これにより、他のリファレンスを *$var に代入し、 ローカル変数のリファレンスのみを変更します。

> **注意**: foreach ステートメント の内部でリファレンス変数に値を代入すると、リファレンスも変更されます。
>
> **Example#3 リファレンスと foreach ステートメント**
>
> ```
> <?php
> $ref = 0;
> $row =& $ref;
> foreach (array(1, 2, 3) as $row) {
> ```

```
    // 何かを実行します
}
echo $ref; // 3 - 配列の最後の要素
?>
```

リファレンスの第 2 の使用法は、変数のリファレンス渡しです。この場合、 関数でローカル変数が作成され、コール側の変数が、それと同じ内容へのリファレンスとなります。例を示します。

```
<?php
function foo(&$var)
{
  $var++;
}

$a=5;
foo($a);
?>
```

この結果、*$a* は 6 となります。これは、関数 *foo* の中では、変数 *$var* は *$a* と同じ内容を指しているためです。 より詳細な説明は、 リファレンス渡し を参照ください。

リファレンスの第 3 の使用法は、 リファレンスによる返り値 です。

リファレンスが行わないこと

これまでに説明したように、リファレンスはポインタではありません。このため、 次の例は期待通りに動作しません。

```
<?php
function foo(&$var)
{
    $var =& $GLOBALS["baz"];
}
foo($bar);
?>
```

ここでの動作としては、関数 *foo* の *$var* はコール側の *$bar* と関連付けられますが、*$GLOBALS["baz"]* に再結合されるといったものになります。*$bar* は関数 *foo* で利用できないため、 リファレンス以外にはコール側の変数スコープにある *$bar* を何かに結合する手段はありません （この変数は *$var* として表されていますが、 *$var* はその変数の内容のみを有しており、 コール側のシンボルテーブルで名前と変数を結合したものではありません）。 関数内で指定した変数を参照するには、リファレンス返し が使用可能です。

リファレンス渡し

リファレンスにより関数に変数を渡すことが可能です。この場合、関数内で その引数を修正可能になります。構文は次のようになります。

```
<?php
function foo(&$var)
{
    $var++;
}

$a=5;
foo($a);
// $a はここでは 6 です
?>
```

関数コールの際には、リファレンス記号がないことに注意してください。 関数定義にのみリファレンス記号があります。リファレンスで正しく引数を渡すには、関数定義のみで十分です。以前のバージョンの PHP では *foo(&$a);* のような形式で & を利用すると "Call-time pass-by-reference" という警告が発生していましたが、 これは今では廃止されています。

次のものはリファレンスで渡すことが可能です。

- 変数、すなわち、*foo($a)*
- new 命令、すなわち、*foo(new foobar())*

- 関数から返されるリファレンスは、次のようになります。

  ```
  <?php
  function &bar()
  {
      $a = 5;
      return $a;
  }
  foo(bar());
  ?>
  ```

  リファレンスによる 返り値 に関する説明も参照ください。

他の式は、結果が未定義となるため、リファレンスで渡すべきではありません。 例えば、リファレンスで渡す次の例は、無効です。

```
<?php
function bar() // & がないことに注意
{
    $a = 5;
```

```
    return $a;
}
foo(bar()); // PHP 5.0.5 以降、致命的なエラーが発生する

foo($a = 5); // 式、変数ではない
foo(5); // 致命的なエラーが発生する
?>
```

以上の説明は、PHP 4.0.4 以降用です。

リファレンスを返す

リファレンスを返すことは、結合する変数を見付けるために関数を使用し たい場合に便利です。パフォーマンスを向上させるためだけの目的で この機能を用いることはやめてください。 そのようなことをしなくても、PHP エンジンが自動的に最適化を行います。 リファレンスを返すのは、そうすべき妥当な理由がある場合に限られます! リファレンスを返す場合、次の構文を使用して下さい。

```
<?php
class foo {
    public $value = 42;

    public function &getValue() {
        return $this->value;
    }
}

$obj = new foo;
$myValue = &$obj->getValue(); // $myValue は $obj->value へのリファレンス、つまり 42 となります
$obj->value = 2;
echo $myValue;                // $obj->value の新しい値である 2 を表示します
?>
```

この例では、関数 *getValue* により返された オブジェクトのプロパティが、設定されます。リファレンス構文を 使用しない場合のようにコピーとなるわけではありません。

> **注意**: パラメータを渡す場合と異なり、ここでは、通常のようにコピーでは なくリファレンスで返り値を指定し、リファレンス結合を指定するために 両方の場所で *&* を使用する必要があります。 *$myValue* について行われたのは、通常の代入ではありません。

> **注意**: 以下のような形式で関数からリファレンスを返そうとした場合、 *return ($this->value);* これは、あなたが望んでいるように 式 の結果を返してくれることは*ありません*。 可能なことは、値へのリファレンスを返すことができるということだけで、 それ以外の何者でもありません。 PHP 4.4.0 および PHP 5.1.0 以降では、式の結果や *new* 演算子の結果をそのまま返そうとした場合に **E_NOTICE** エラーが発生します。

リファレンスの解除

リファレンスを解除することは、ちょうど変数名と変数の内容の結合を 解除することに相当します。これは、変数の内容が破棄されることを 意味しません。例えば、

```
<?php
$a = 1;
$b =& $a;
unset($a);
?>
```

は、*$b* を削除せず、*$a* のみを 削除します。

ここでも、Unix の **unlink** コールと類似したものと 考えると便利です。

リファレンスの適用範囲

PHP の多くの構文構造は、リファレンス機構を利用して実装されています。 このため、前記のリファレンス結合に関する事項はこれらの構造についても 適用されます。リファレンス渡しおよびリファレンスの返り値のような いくつかの構造について前節で記述されています。リファレンスを使用する他の構造には次のものがあります。

*global*リファレンス

変数を **global $var** として宣言した場合、実際には グローバル変数へのリファレンスを作成したことになります。この意味は、 次の例と同じです。

```
<?php
$var =& $GLOBALS["var"];
?>
```

これは、例えば、*$var* を削除してもグローバル変数は 削除されないことを意味します。

$this

オブジェクトのメソッドでは、*$this* は 常にコール側のオブジェクトへのリファレンスです。

セキュリティ

目次

はじめに

PHP は強力な言語そしてインタプリタであり、モジュールとして Web サー バーに組み込んだ場合でも、独立した CGI バイナリ として実行される場合でも、ファイルをアクセスしたり、コマンドを実行 したり、サーバーへのネットワーク接続を開くことができます。デフォル トでは、これらの機能を実行した場合、Webサーバー上でセキュリティ上の 問題を生じる可能性があります。PHP は、特に CGI プログラムを書く場合、 Perl や C より安全な言語となるように設計されています。コンパイル時 または実行時の設定オプションを正しく選び、適切なコードを書くことに より、必要な自由度とセキュリティの組み合わせを確実に提供することが できます。

PHP の使用法に多くの異なった手段があるように、PHP の動作を制御する 多くの設定オプションがあります。オプションの選択肢が広いため、PHP を様々な用途に使用することができます。しかし、このことは、これらの オプションとサーバー設定の組み合わせによっては、安全でない設定とな ることを意味します。

PHPの設定の自由度はそのコードの柔軟さにほぼ匹敵します。PHPは、シェ ルユーザコマンドを全て実行可能な完全なサーバーアプリケーションや 厳しく制御された環境で低リスクの簡単なサーバーサイドインクルードを 使用できるようなアプリケーションを構築する場合に使用することが可能 です。そうした環境の構築方法、セキュリティのレベルはPHPの開発者に大 きく依存しています。

本章は、安全に使用可能な異なった設定と条件の組み合わせについての説 明から始めます。続いて、複数のセキュリティレベルのコーディングにお ける複数の考慮事項について説明し、最後にいくつかの一般的なセキュリ ティ上のアドバイスを行います。

一般的な考慮事項

完全に安全なシステムは理想の産物でしかないため、セキュリティ業界で しばしば使用される手法は、リスクと利便性のバランスのとれた手法です。変数がユーザから投稿される度に(網膜スキャンと指紋のような)2種類の 生体認証が必要だとしたら、極端に高いレベルの説明義務を生じます。ま た、かなり複雑なフォームを埋めるために30分もかかるとすれば、ユーザ がセキュリティをバイパスする手段を探す気分にさせる傾向があります。

最善のセキュリティは、通常、ユーザによる業務の達成を防たげずに要求 を十分に達成できる程度にさしでがましくないものです。むしろ、いくつ かのセキュリティ攻撃は、単純にこの種の多げさに構築され、時間を浪費 しがちなセキュリティ機構を狙うものです。

記憶するに値する言葉として次のようなものがあります。「システムは鎖 の最も弱い輪と同程度に優れている」全てのトランザクションが時間、場所、トランザクションの型等に基づき大量に記録されているが、ユーザは 一つのクッキーのみにより認証されている場合、ユーザとそのトランザクションログの結び付きの確実性はかなり弱くなります。

テストの際に、最も簡単なページに関してでさえ、全ての可能性をテスト することは不可能であるということを頭に入れておいてください。期待する入力は、不機嫌な社員、経験のあるクラッカー、キーボードの上を歩く家 の猫による入力とは全く無関係でしょう。これが、想定外のデータが入力 される可能性がある場所を見分けるために論理的な視点からコードを見て、 その後、修正、減少、または詳細に調べるというのが、最善であるという 理由です。

インターネットにはあなたのコードを壊したり、システムを破壊したり、 不適切な内容を投稿したり、その他あなたの一日を不快にするようなこと により自分の名を馳せたいと思う人がたくさんいます。サイトの規模の大 小によらず、単にオンラインであり、接続できるサーバを有しているだけ で攻撃目標となりえます。多くのクラック用プログラムはサイトの大きさ を考慮せず、犠牲者を探しつつ大きなIPブロックで網を張っています。 その犠牲者の一人にならないようにしてください。

CGI バイナリとしてインストール

目次

受ける可能性がある攻撃

PHP を CGI バイナリとして使用するのは、PHP を モジュールとして(Apache のような)サーバーソフトウエアに組み込み たくない何らかの理由がある場合や安全な chroot と setuid 環境をス クリプトに提供する他の CGI ラッパーと組み合わせて PHP を使用する 場合の設定オプションです。セットアップ時には、通常、PHP 実行バイ ナリを Web サーバーの cgi-bin ディレクトリにインストールします。 CERT 勧告 » CA-96.11は、いかなるインタプリタを cgi-bin に置くことにも反対しています。 PHP バイナリをスタンドアロンのインタプリタとして使用することが できる場合でも、PHP は、セットアップにより生じる可能性がある 次のような攻撃を防ぐように設計されています。

- システムファイルへのアクセス: *http://my.host/cgi-bin/php?/etc/passwd* URL において疑問符 (?) の後のクエリー情報は、CGI インターフェー スにより、インタプリタにコマンドライン引数として渡されます。通 常、インタプリタは、コマンドライン上の最初の引数に指定されたファ イルを開き、実行します。 CGI バイナリとして実行された場合、PHP は、コマンドライン引数の 解釈を拒否します。
- サーバー上の Web ドキュメントへのアクセス: *http://my.host/cgi-bin/php/secret/doc.html* URL の PHP バイナリ名の後のパス情報の部分、つまり*/secret/doc.html* は、 CGI プログラムによりオープンされて実行される ファイルの名前を指定するために従来より使用されています。 *http://my.host/secret/script.php* のようなドキュメントへの要求を PHP インタプリタにリダイレクト するために、通常、何らかの Web サーバー設定用命令(Apache では Action) が使用されます。この設定により、Web サーバーは、まずディレクトリ */secret* へのアクセス権をチェックし、 リダイレクト要求 *http://my.host/cgi-bin/php/secret/script.php* を生成します。残念なことに、リクエストが最初からこの形式で与え られた場合、Web サーバーによるアクセスチェックは、 */secret/script.php* ファイル ではなく、*/cgi-bin/php* ファイル に対して行われます。この手法により、*/cgi-bin/php* にアクセス可能なユーザーは、 Web サーバー上の全ての保護されたドキュメントにアクセスできてし まいます。 PHP では、サーバードキュメントツリーにアクセス制限付きのディレ クトリがある場合、コンパイル時の設定オプション --enable-force-cgi-redirect および実行時の設定命令 doc_root と user_dir をこの攻撃を防止す るために使用することができます。 これらを組み合わせたいくつか の手法について以下に詳細な説明を示します。

ケース 1: 配布制限がないファイルのみを配布

サーバー上にパスワードまたは IP アドレスを元にしたアクセス制限に よる制約を課すコンテンツがない場合、この設定オプションを使用す る必要はありません。使用する Web サーバーがリダイレクトを許可しな い場合やサーバーがリダイレクト要求を安全に処理しつつPHP バイナリ と通信できる手段を有していない場合、オプション --enable-force-cgi-redirectを configure スクリプトに指定することができます。この場合でも、直接 的な方法 *http://my.host/cgi-bin/php/dir/script.php* でもなくリダイレクション *http://my.host/dir/script.php*でもない他の やり方で PHP スクリプトを呼び出せるようになっていないかどうか確認 する必要があります。

リダイレクションは、例えば Apache では命令 AddHandler および Action で設定することができます。(以下を参照してください。)

ケース 2: --enable-force-cgi-redirect を使用

このコンパイル時のオプションは、 *http://my.host/cgi-bin/php/secretdir/script.php* のように URL から直接 PHP を呼び出すことを禁止します。代わりに、 Web サーバーのリダイレクションにより処理された場合は、 PHP はこのモードでのみ処理を行います。

通常、Apache 用設定でのリダイレクションは、 次の命令を使用して行います。

```
Action  php-script  /cgi-bin/php
AddHandler  php-script  .php
```

このオプションは、Apache Web サーバーでのみテストされており、リク エストのリダイレクト時に Apache が標準ではないCGI 環境変数 REDIRECT_STATUS をセットすることを前提にしています。 リクエストが直接のものであるか間接のものであるかを示す手段をWeb サーバーが全くサポートしていない場合は、このオプションを使用する ことはできません。この場合、ここで記した CGI 版を実行する他の方法 の内の一つを使用する必要があります。

ケース 3: doc_root または user_dir を設定

Web サーバー上のドキュメントディレクトリに スクリプトや実行ファイルのようなアクティブな内容を読み込むのは、 往々にして危険な行為であるとみなされることがあります。 何らかの設定ミスによりスクリプトが実行されず、通常の HTML ドキュメント として表示されてしまう場合には、知的著作物またはパスワードのような セキュリティ情報が漏洩する可能性があります。 このため、多くのシステム管理者は、スクリプトを PHP CGI を通じて

のみ アクセス可能な他のディレクトリ構造にセットアップしたいと思うこと でしょう。 この場合、常にインタープリタに処理されるため、上記のように表示されること はありません。

前節で記したようなリクエストがリダイレクトされたものでないことを 確かめる方法が利用可能でない場合、 スクリプト用の doc_root を Web ドキュメント用ルートとは別に セットアップする必要があります。

設定用命令 doc_root により 設定ファイル ファイル中で PHP スクリプト用ドキュメントルートを設定することができます。 または、環境変数 PHP_DOCUMENT_ROOT でも設定する ことができます。 これを設定した場合、CGI 版の PHP は、 常に開くファイルの名前をこの *doc_root* リクエストのパス情報を用いて作成し、 (以下の *user_dir* を除き、)確実に このディレクトリの外側でスクリプトが実行されないようにします。

ここで利用可能な別のオプションは、 user_dir です。user_dir が設定されていない場合、 開かれるファイル名を制御するのは、*doc_root* のみです。*http://my.host/~user/doc.php* のような URL は、ユーザーホームディレクトリ以下のファイルを開かず、 doc_root 以下の *~user/doc.php* というファイルを開くことになります。 (ディレクトリ名がチルダ [~] で始まっている ということになります)

user_dir が例えば、*public_php*に 設定されていた場合、 *http://my.host/~user/doc.php* の ようなリクエストは、そのユーザー user のホームディレクトリにある *public_php* 以下の *doc.php* という名前のファイルをオープンしま す。ユーザーのホームディレクトリが、 */home/user* である場合、 実行されるファイルは、 */home/user/public_php/doc.php* となります。

user_dir の展開は、 *doc_root* の設定によらず行われます。 このため、ドキュメントルートおよびユーザーディレクトリへの アクセスを別々に制御することができます。

ケース 4: Webツリーの外にPHPパーサを置く

非常に安全性の高いオプションとしてPHP パーサのバイナリをファイル 用 Web ツリーの外側、例えば */usr/local/bin*に置くことが考えられます。このオプションの唯一の欠点は、PHP タグを有する全てのファイルの先頭 行に次のような一行を加える必要があることです。

```
#!/usr/local/bin/php
```

また、ファイルを実行可能にしておく必要があります。この場合、実行 時にシェルエスケープ機能 *#!* を使用する Perl や sh や他のスクリプト言語で書かれた CGI スクリプトを処理するのと全 く同様に処理を行います。

この設定で PATH_INFO および PATH_TRANSLATED 情報を正しく処理するためには、 PHP パーサを設定オプション --enable-discard-path を付けてコンパイルする必要があります。

Apache モジュールとしてインストール

PHP が Apache モジュールとして使用された場合、PHP は、Apache ユー ザーの許可属性(通常はユーザー "nobody" の許可属性)を継承します。 これは、セキュリティと認証に数々の影響を与えます。例えば、データベー スと接続するためにPHPを使用している場合、データベースが組込みのア クセス制御機能を有していない限り、そのデータベースを "nobody"ユー ザからアクセス可能とする必要が生じます。これは、悪意のあるスクリプ トが、ユーザ名とパスワードなしにデータベースにアクセスし、修正する ことができることを意味します。Webスパイダがデータベース管理用Webペー ジを回って、データベースを全て削除することも可能です。Apache認証に よりこの攻撃に対して防衛することが可能であり、また、LDAPや .htaccess ファイル等を使用して固有のアクセスモデルを設計し、PHPスク リプトの一部としてそのコードを読み込むことも可能です。

しばしば、PHPユーザ(この場合はApacheユーザ)が非常に小さなリスクを 有する場所に一度セキュリティが確立されると、PHPはユーザディレクト リにウイルスファイルを書き込んだりすることができなくなります。もし くは、データベースにアクセスしたり変更したりといったことが出来なく なります。この場合、良いファイルおよび悪いファイルの書き込み、また は、良いデータベーストランザクションと悪いデータベーストランザクシ ョンに関して等しく安全性が確保されていると言えます。

この観点からしばしば行われるセキュリティ上の失敗としてApacheにルー ト権限を与えたり、他の何らかの手段でApacheの権限を昇格させるというものがあります。

Apacheユーザの権限をルートに昇格させることは非常に危険であり、シ ステム全体を危険にさらす可能性があります。よって、sudoやchrootの実行、ルート権限で実行を行う他の手段は、セキュリティに精通した人以外 は、考慮するべきではありません。

いくつかのより簡単な解決策があります。open_basedir を使用することによ り、PHPに使用を許可するディレクトリを制御したり制限したりすること が可能です。また、全てのWebベースの作業をユーザファイル、システム ファイル以外のファイルに制限するために、Apache専用エリアを設定する ことも可能です。

ファイルシステムのセキュリティ

PHP は、ファイルおよびディレクトリ毎に権限を設定する多くのサーバシ ステム上に組み込まれたセキュリティを提供します。これにより、ファイ ルシステム内のファイルを読み込み可能に制御することが可能になります。 全てのファイルは世界中から読み込み可能であり、このファイルシステム にアクセスした全てのユーザから読み込まれても安全であることを確認す る必要があります。

PHPは、ファイルシステムにユーザレベルのアクセスを許可するように設 計されているため、PHPスクリプトから/etc/password のようなシステム ファイルを読み込み可能としたり、イーサネット接続を修正したり、巨大 なプリンタジョブを出力したりすることができます。これから明らかにわ かることですが、読み書きするファイルを適切に設定する必要があります。

各自のホームディレクトリにあるファイルを削除する次のスクリプトを見 てみましょう。これは、ファイル管理用にWebインターフェースを使用す る場合に通常生じるような設定を仮定しています。この場合、Apacheユー ザはそのユーザのホームディレクトリにあるファイルを削除可能です。

Example#1 甘い変数の確認から生じるリスク

```
<?php
// ユーザのホームディレクトリからファイルを削除する
$username  = $_POST['user_submitted_name'];
$userfile  = $_POST['user_submitted_filename'];
$homedir   = "/home/$username";

unlink("$homedir/$userfile");

echo "ファイルは削除されました!";
?>
```

username および filename はユーザフォームから投稿可能であるため、別の username および filename を投稿して 削除すべきではない削除することが可能となります。この場合、 他の何らかの形式の認証を使用するべきです。投稿された変数が、 "../etc/" と "passwd " であった場合について考えてみましょう。簡単 なコードを以下に示します。

Example#2 ... ファイルシステムへの攻撃

```
<?php
// 外部からPHPユーザがアクセス可能なハードドライブを削除します。PHPが
// ルートのアクセス権限を有している場合、
$username  = $_POST['user_submitted_name']; //  "../etc"
$userfile  = $_POST['user_submitted_filename']; //  "passwd"
$homedir   = "/home/$username"; //  "/home/../etc"

unlink("$homedir/$userfile"); //  "/home/../etc/passwd"

echo "ファイルは削除されました!";
?>
```

こうした問題を防止するために必要な重要なチェック手段として以下の2 種類のものがあります。

- PHP Webユーザバイナリに制限された権限のみを許可する。
- 投稿された全ての変数を確認する。

以下に改良されたスクリプトを示します。

Example#3 より安全なファイル名の確認

```
<?php
// PHPユーザがアクセス可能なハードドライブからファイルを削除する。
$username  = $_SERVER['REMOTE_USER']; //  認証機構を使用する
$userfile  = basename($_POST['user_submitted_filename']);
$homedir   = "/home/$username";

$filepath  = "$homedir/$userfile";

if (file_exists($filepath) && unlink($filepath)) {
    $logstring  = "$filepath  を削除しました\n";
} else {
    $logstring  = "$filepath  の削除に失敗しました\n";
}
$fp  = fopen("/home/logging/filedelete.log", "a");
fwrite($fp, $lo  gstring);
fclose($fp);

echo htmlentities($logstring, ENT_QUOTES);

?>
```

しかし、これでも、傷口を塞いだことにはなりません。 ユーザが自分用のユーザログインを作成することをあなたの認証システムが 許可しており、ユーザが"../etc/"へのログインを選択した場合、システム はまたも公開されてしまいます。このため、よりカスタマイズされたチェッ クを行なう方がよいでしょう。

Example#4 より安全なファイル名の確認

```
<?php
$username      = $_SERVER['REMOTE_USER']; //  認証機構を使用する
$userfile       = $_POST['user_submitted_filename'];
$homedir        = "/home/$username";

$filepath       = "$homedir/$userfile";

if (!ctype_alnum($username) || !preg_match('/^(?:[a-z0-9_-]|¥.(?!¥.))+$/iD', $userfile)) {
     die("Bad  username/filename");
}

//etc...
?>
```

オペレーティングシステムにより、注意するべきファイルは大きく変化し ます。これらには、デバイスエントリ(/dev/ または COM1)、設定ファイ ル(/etc/ ファイルおよび .ini ファイル)、よく知られたファイル保存領 域 (/home/、 My Documents)等が含まれます。このため、明示的に許可す るもの以外の全てを禁止する方針とする方が通常はより簡単です。

Null バイト関連の問題

PHP はファイルシステム関連の操作に C 言語の関数を使用しているので、 null バイトの処理を予期せぬかたちで行うことがあります。 C 言語では null バイトは文字列の終端を表すので、 null バイトを含む文字列があった場合に null バイト以降の内容は文字列として処理されません。 以下に、この問題に関する脆弱性を含むコード例を示します。

Example#5 null バイトに対して脆弱なスクリプト

```
<?php
$file = $_GET['file']; //  ここで  "../../etc/passwd\0"  が渡されたとします
if (file_exists('/home/wwwrun/'.$file.'.php')) {
    // file_exists は true を返します。これは、ファイル /home/wwwrun/../../etc/passwd が存在するからです
    include '/home/wwwrun/'.$file.'.php';
    // ファイル /etc/passwd がインクルードされてしまいます
}
?>
```

したがって、ファイルシステム操作で使用する「汚染された」文字列は、 つねに適切に検証しなければなりません。 先ほどの例を改良したものを示します。

Example#6 入力を適切に検証する例

```
<?php
$file = $_GET['file'];

// 値として与えられる可能性のある、有効な値の一覧を作成します
switch ($file) {
    case 'main':
    case 'foo':
    case 'bar':
        include '/home/wwwrun/include/'.$file.'.php';
        break;
    default:
        include '/home/wwwrun/include/main.php';
}
?>
```

データベースのセキュリティ

目次

今日、ダイナミックなコンテンツを提供するウェブアプリケーションに おいてはデータベースは欠く事のできなコンポーネントとなっています。 そういったデータベースには重要な、そして秘密にすべき情報が格納 されることになるので、それらをいかにして保護するかについて十分に 考慮する必要があります。

情報を取り出したり格納するためにはデータベースに接続する必要があります。 そして適切なクエリを送信し、結果を受け取り、切断します。クエリに使用される言語はStructured Query Language (SQL)が一般的です。アタッカー がどのようにSQLに 干渉するかについて参照してください。

皆さんがお気づきの様に、PHPそれ自体は貴方のデータベースを保護することは ありません。以下のセクションはPHPスクリプトからどのようにデータベースに アクセスし操作すればいいのか、とういことに関する非常に基本的な導入です。

このシンプルなルールを覚えて置いてください：それは「多重防衛」です。 より多くの箇所で、より多くの保護を行うことにより、アタッカーが攻撃に成功して機密情報が漏洩する可能性は減っていきます。データベースと アプリケーションを正しくデザインすることで貴方の心配を取り除くことが できます。

データベースのデザイン

他人が用意した既存のものを使用するのでない限り、最初に行うのはデータベースの作成です。 データベースが作成されると、そのデータベースのオーナーは作成コマンドを 実行したユーザになります。通常、オーナー(とスーパーユーザー)のみが そのデータベースに対して操作を行うことが出来ます。他のユーザがデータベースを 使用するには適切な権利が与えられている必要があります。

アプリケーションはデータベースにオーナー、もしくはスーパーユーザーとして 接続しては絶対にいけません。なぜならこれらのユーザは 例えばスキーマの変更(テーブルの削除等)や全コンテンツの削除、といった あらゆるクエリーを実行することが出来るからです。

貴方が作成するアプリケーションがデータベースに対して行う操作の各方面ごとに、 操作対象となるオブジェクトに対して、出来る限り少ない権限を持った複数の ユーザを作成した方が良いでしょう。ユーザに対しては、最低限必要な権限のみを 与え、関係の無いデータへのアクセスを許可しないようにします。これは、 万が一侵入者がそのユーザの権限を以ってデータベースにアクセスした際に、 アプリケーションと関係の無いデータにまでアクセスされることを防ぐためです。

全てのビジネスロジックをウェブアプリケーション(つまり貴方のスクリプト) で実装することは推奨されません。代わりに、ビュー、トリガー、ルール

といった データベースの機能を活用した方が良いでしょう。システムが更新され、 新しい機能がデータベースへのアクセスすることになった場合、個々のデータベース クライアントごとに再度同様のロジックを実装しなければならなくなります。 さらに、トリガーは透過的に、そして自動的にフィールドを扱うことが出来るので、 デバッグ時や、トランザクションのロールバック時に役立ちます。

データベースへの接続

更なるセキュリティのために、クライアント/サーバ間の通信においてSSLを用いた 暗号化を行った方が良いでしょう。もしくはsshを使用することも出来ます。 どちらかの手段を講じた後、トラフィックをモニタリングしてみれば ここから何らかの情報を得ることが困難だという事が分かると思います。

ストレージの暗号化

SSL/SSHによってクライアント/サーバ間で通信されるデータは保護されますが、 データベースに保存されたデータは保護されません。SSLはあくまで通信上の プロトコルなのです。

一旦アタッカーがデータベースへ(ウェブサーバを通さずに)アクセスできてしまうと、 そこに格納されているデータ自体が暗号化されていない限り、自由に閲覧され、 使用されてしまいます。データを暗号化することによって、この脅威を減らすことが できますが、この機能をサポートしているデータベースは僅かです。

この問題への最も簡単な対応策は、まず自分専用の暗号化パッケージを作成し、 それをあなたのPHPスクリプトから使用することです。PHPの Mcrypt, Mhash といった幾つかの拡張モジュールは、様々な暗号化アルゴリズムをサポート しているので役に立つでしょう。データ格納時に暗号化を行い、取得時に 復号化します。この方法についてはリファレンスを参照してください。

もし完全にデータを隠したい場合や、元のデータ自体は必要ない場合(つまり 表示されない場合)は、ハッシュも考慮に入れたほうが良いでしょう。 ハッシュの良く知られた使用方法は、パスワードをそのまま格納せずに、 そのMD5ハッシュ値を格納する方法です。crypt()や md5()も参照してください。

Example#1 ハッシュされたパスワードフィールドを使う

```
// ハッシュされたパスワードを格納する
$query    = sprintf("INSERT INTO users(name,pwd) VALUES('%s','%s');",
                    pg_escape_string($username), md5($password));
$result = pg_query($connection, $query);

// パスワードが正しいかどうか問い合わせる
$query = sprintf("SELECT 1 FROM users WHERE name='%s' AND pwd='%s';",
                    pg_escape_string($username), md5($password));
$result = pg_query($connection, $query);

if (pg_num_rows($result) > 0) {
      echo 'Welcome, $username!';
} else {
      echo "$username の認証が失敗しました。";
}
```

SQLインジェクション

多くの開発者はSQLクエリがどのように改竄されるかということを余り 気にかけておらず、またSQLクエリは信用できるものと考えているようです。実際にはSQLクエリはアクセス制限を回避することが可能で、従って 通常の認証や権限のチェックを無視することができます。時には、 OSレベルのコマンドを実行できてしまうこともあります。

ダイレクトSQLコマンドインジェクション(SQLコマンドの直接実行)という手法は、 攻撃者がSQLコマンドを生成もしくは既存のコマンドを変更することで 隠蔽すべきデータを公開したり、重要なデータを書き換えたり、データベース ホストで危険なシステムレベルのコマンドを実行したりするものの事です。 この手法は、ユーザからの入力をスタティックなパラメータと組み合わせて SQLクエリを生成するアプリケーションにおいて使用されます。以下の例は 不幸なことに実際の事例に基づいたものです。

入力のチェックを怠っており、スーパーユーザもしくはデータベース作成権限を 持つユーザ以外のユーザでデータベースに接続していない ために、攻撃者はデータベースにスーパーユーザを作成することが出来ます。

Example#1 表示するデータを分割し ... そしてスーパーユーザを作成します。 (PostgreSQLの例)

```
$offset = $argv[0]; // 入力チェックが行われていません！
$query   = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";
$result = pg_query($conn, $query);
```

通常のユーザは、*$offset*がURL埋め込まれている '次へ'または'前へ'リンクをクリックします。スクリプトは、受け取った *$offset*が数字であることを期待します。しかしながら、 攻撃者はurlencode()された以下のようなURLを追加 することで攻撃を試みます。

```
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
      select 'crack', usesysid, 't','t','crack'
      from pg_shadow where usename='postgres';
--
```

このようなことが行われると、スクリプトは攻撃者にスーパーユーザ権限での アクセスを提供してしまいます。*0;*が正しいオフセット 指していると同

時に、クエリをそこで終端させていることに気をつけてください。

> **注意**: SQLパーサにクエリの残りの部分を無視させるために開発者によく使わ れる技法として、SQLのコメント記号である--があ ります。

パスワードを取得する恐るべき手段に、サイトの検索結果のページを欺く というものがあります。攻撃する者が必要とするものは、投稿された変数 の中でSQL命令で使用される際に正しく扱われていないものがあるかどう かを確かめるだけです。これらのフィルタは、通常、 *SELECT*文の*WHERE, ORDER BY, LIMIT*及び*OFFSET*句をカスタマイズするた めに前に置かれる形で設定されます。使用するデータベースが *UNION*構造をサポートしている場合、 攻撃者は元のクエリに任意のテーブルからパスワードのリストを取得する クエリを追加しようとするかもしれません。 暗号化されたパスワードフィールドを使用することが強く推奨されます。

Example#2 記事...そして(全てのデータベースサーバーの)いくつかのパスワード のリストを表示する

```
$query  = "SELECT id, name, inserted, size FROM products
                  WHERE size = '$size'
                    ORDER BY $order LIMIT $limit, $offset;";
$result = odbc_exec($conn, $query);
```

クエリの静的な部分は、以下のように全てのパスワードを外部にもらす別の *SELECT*文と組み合わせることができます。

```
'
union select '1', concat(uname||'-'||passwd) as name, '1971-01-01', '0' from usertable;
--
```

(及び--を使用する) このクエリが*$query*で使用される変数の 1 つに代入 された場合、この悪意のあるクエリが実行されることになります。

SQL UPDATE もデータベースを攻撃するために使用されます。これらのク エリも切捨てたり新しいクエリを元のクエリに追加することによる攻撃 を受けます。しかし、攻撃者は*SET*句を使用する可 能性があります。この場合、クエリを成功させるためにいくつかのスキー マ情報を保有する必要があります。これは、フォームの変数名や総当た り法により調べることができます。パスワードまたはユーザ名を保存す るフィールド用の命名記法はそう多くはありません。

Example#3 パスワードのリセットから ... (全てのデータベースサーバーで)より多 くの権限を得るまで

```
$query = "UPDATE usertable SET pwd='$pwd' WHERE uid='$uid';";
```

しかし、悪意のあるユーザは、管理者のパスワードを変更するために 値 *' or uid like'%admin%';* -- を *$uid* に代入するか、または、より多くの権限を得 るために、単純に*$pwd* に(後ろに空白を付けて) *"hehehe', admin='yes', trusted=100 "*と設定する 可能性があります。この場合、このクエリは以下のように改謬されてし まいます。

```
// $uid == ' or uid like'%admin%'; --
$query = "UPDATE usertable SET pwd='...' WHERE uid='' or uid like '%admin%'; --";

// $pwd == "hehehe', admin='yes', trusted=100 "
$query = "UPDATE usertable SET pwd='hehehe', admin='yes', trusted=100 WHERE ...;"
```

恐ろしい例として、いくつかのデータベースホストのオペレーティン グシステムレベルのコマンドがアクセス可能となる方法を示します。

Example#4 データベースホストのオペレーティングシステムを攻撃する (MSSQLサーバー)

```
$query  = "SELECT * FROM products WHERE id LIKE '%$prod%'";
$result = mssql_query($query);
```

攻撃者が、値 *a%' exec master..xp_cmdshell 'net user test testpass /ADD' --* を*$prod*に投稿した場合、 *$query* は以下のようになります。

```
$query  = "SELECT * FROM products
                    WHERE id LIKE '%a%'
                       exec master..xp_cmdshell 'net user test testpass /ADD'--";
$result = mssql_query($query);
```

MSSQLサーバーは、新規ユーザをローカルアカウント用データベースに追 加するコマンドを含むSQL命令をバッチ実行します。 このアプリケーションが*sa*で実行され、 MSSQLSERVERサービスが充分な権限で実行されている場合、攻撃者は このマシンにアクセスする権限を有することになります。

> **注意**: 上記のいくつかの例は、データベースサーバの種類に依存しています。 これは、他の製品に対して同様な攻撃ができないことを意味するもので はありません。使用しているデータベースが他の手段で攻撃可能である 可能性もあります。

回避する方法

ほとんどの例では攻撃する者は特定のデータベースのスキーマに関して 若干の情報を保有している必要があると弁解されるかもしれません。 これは正しいですが、いつ何時これが外部にもれうるかを知ることはで きませんし、いったんこの情報がもれると、データベースが外部に開示 される可能性があります。オープンソースまたは一般的に入手可能なデー タベース処理パッケージを使用している場合(これはコンテンツ監理シス テムまたはフォーラムに含まれている可能性があります)、侵入者は簡単 に使用されているコードの一部を入手することができます。そのコード の設計が悪い場合にもセキュリティリスクを生じる可能性があります。

これらの攻撃は、セキュリティを考慮して書かれていないコードを攻撃 する方法です。特にクライアント側から入力されるあらゆる種類の入力 を決して信用しないでください。これは、selectボックスやhidden input フィールド、Cookieの場合も同様です。最初の例は、このような欠点の ないクエリが破滅をもたらしうることを示すものです。

- データベースにスーパーユーザーまたはデータベースの所有者として 接続しないでください。 非常に制限された権限を有するカスタマイズ されたユーザを常に使用してください。
- 指定された入力が期待するデータ型であることを確認してください。 PHPは、多くの種類の入力検証用関数を有しており、 変数関連の関数や 文字型関数にある簡単な関数 (例: それぞれ、is_numeric(), ctype_digit()) や、 Perl互換の正規表現のサポートま であります。
- アプリケーションが、数値入力を期待している場合、データを is_numeric()で検証するか、 settype()により暗黙の型変換を行うか、 sprintf()に

より数値表現を使用することを検討 してみてください。

Example#5 ページング用のクエリを構築するためのより安全な方法

```
settype($order,  'integer');
$query   = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";

// フォーマット文字列の%dに注意してください。%sを使用しても意味がありません。
$query = sprintf("SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET %d;",
                 $offset);
```

- データベースに渡される数値以外のユーザ入力を データベース固有の文字列エスケープ関数 (mysql_escape_string(), **sql_escape_string()**, など) でクオートしてください。 データベース固有の文字列エスケープ機能が利用できない場合、 addslashes() および str_replace()関数が利用できるでしょう。 (データベースの型に依存) 最初の例を参照 してください。前期の例が示すように、クエリの静的な部分をクオート するだけでは充分ではなく、簡単にクラックされてしまう可能性があ ります。
- データベース固有の情報、特にスキーマに関する情報は出力してはい きません。エラー出力およ びエラー処理およびログ関数 も参照ください。
- ユーザがテーブルまたはビューに直接アクセスできないように、 データアクセスを抽象化することを目的としてストアドプロシージャ 及び事前に定義したカーソルを使用することもできますが、このソリュー ションには、副作用があります。

これらのケースにおいて、スクリプトまたはサポートされている場合は データベース自体でクエリのログをとることが有益です。 明らかにログは破壊的な行為を防止することはできませんが、攻撃され たアプリケーションを追跡する際には有効です。ログ自体は有益ではあ りませんが、含まれている情報は有益です。通常、より詳細なログをと る方が良いでしょう。

エラーのレポート

PHPのセキュリティに関して、2種類のエラー出力があります。一つは、セ キュリティ向上に役立つものであり、もう一つは、セキュリティ上有害なものです。

標準的な攻撃手法の中に不完全なデータをシステムに送信し、返されるエ ラーの種類と内容を調べることにより、システムを調べるというものがあ ります。これにより、システムのクラッカーがありうる弱点を調査するた めにそのサーバに関する情報を調べることが可能になります。 例えば、ある攻撃者が事前のフォーム投稿の際にあるページに関して収集 した情報を持っている場合、変数を上書きしたり、修正したりしようとす るかもしれません。

Example#1 カスタムHTMLページにより変数を攻撃する

```
<form  method="POST"  action="attacktarget?username=badfoo&password=badfoo">
<input  type="hidden"  name="username"  value="badfoo">
<input  type="hidden"  name="password"  value="badfoo">
</form>
```

通常返されるPHPのエラーは、エラーを生じた関数やファイル、エラーを 発生したPHPファイル、エラーを発生した行番号のような情報が含まれて おり、スクリプトをデバッグする開発者に非常に有用です。これらが調べ ることができる情報の全てです。デバッグ目的でPHPの開発者が show_source(), highlight_string(), highlight_file() を使用することはまれではありま せん。しかし、実用サイトでは、これは秘密の変数、未確認の構文、その 他の危険な情報を公開することになってしまいます。特に危険なのは、 組み込みのデバッグハンドラを有する既知のソースからのコードを実行して いるか、一般的なデバッグ技法を使用している場合です。攻撃者が、使用し ている一般的な技法を特定できた場合、次のようにあるページに様々な一般 的なデバッグ用文字列を送信することによりしらみつぶしに攻撃しようとす るかもしれません。

Example#2 一般的なデバッグ変数を探す

```
<form  method="POST"  action="attacktarget?errors=Y&amp;showerrors=1"&debug=1">
<input  type="hidden"  name="errors"  value="Y">
<input  type="hidden"  name="showerrors"  value="1">
<input  type="hidden"  name="debug"  value="1">
</form>
```

エラー処理の方法の方法のいかんによらず、エラーを調べる機能は、攻撃 者により多くの情報を与えることにつながります。

例えば、多くの一般的なエラー形式では、システムはPHPを実行している ことを示します。攻撃者が .html ページを調べ、(システムの既知の弱点 を探すために)誤ったデータを送信することによりバックエンドを調べた いと思った場合、システムをPHPと共に構築されていることを知ることが 可能となる可能性があります。

関数エラーは、システムが特定のデータベースエンジンが実行しているこ と、または、Webページのプログラム内容や設計に関する鍵を示すことが あります。これにより、データベースポートをオープンしたり、Webペー ジに固有のバグや弱点を調べるといったより詳細な調査を行うことが可能 となります。例えば、異なった不正なデータを送信することにより、攻撃 者は、(エラー行番号から)そのスクリプトの異なる場所を調べることと同 時にそのスクリプトの認証の順番を定義することが可能です。

ファイルシステムまたは一般的なPHPエラーは、Webサーバが有する許可属 性やWebサーバのファイル構造を示すことがあります。エラーコードを書く開発者は、元は秘密の情報を容易に公開することにより、この問題を悪 化させる可能性があります。

この問題に対しては3種類の対策があります。最初の対策は、全ての関数 をよく調べ、大部分のエラーの修正を試みることです。2番目は、実行す るコードのエラーリポートを完全に無効にすることです。3番目は、 PHPのカスタムエラー処理関数を使用して独自のエラーハンドラを作成す ることです。システムのセキュリティポリシーに基づき、これらの3種類 の対策が適用可能かどうかを判定します。

この問題を事前に防止する手段の一つは、PHPに組み込まれている error_reporting()機能を使用することです。これにより、 コードを安全にするための手がかりが得られ、危険と思われる変数が使用 されている部分をみつけることが可能です。実使用の前にE_ALLを指定して コードをテストすることにより、変数が他の手段で汚染されたり、修正さ れたりする可能性がある部分を簡単に見つけることが可能です。 実使用の準備ができた際には、あなたのコードをプローブから保護するために error_reporting() を 0 に設定するか、*php.ini* のオプション *display_errors* をオフに設定する、 のいずれかでエラーリポートを完全に無効にすべきです。 もし後者を選択した場合、加えて *error_log* ini ディレクティブを使用する、あるいは *log_errors* をオンにしてログファイルのパスを定義すべきです。

Example#3 E_ALLで危険な変数を見つける

```
<?php
if ($username) {  //  使用前に初期化または確認されていない変数
    $good_login  = 1;
}
if ($good_login == 1) { //  上のテストが失敗した場合、使用前に初期化または確認されていない
    readfile  ("/highly/sensitive/data/index.html");
}
?>
```

グローバル変数の登録機能の使用法

警告

この機能は *非推奨* であり、PHP 6.0.0 で *削除* されます。この機能を使用しないことを強く推奨します。

PHPの変更点で最も議論の対象となったのは、おそらく、PHP » 4.2.0において PHPのディレクティブ register_globalsが デフォルトでONからOFFに変更された時でしょう。 当時、このディレクティブに依存することが一般的であり、多くの人は、 このパラメータの存在すら知らず、PHPの動作そのものであるというよう に考えていました。このページは、このディレクティブにより安全でな いコードを書く可能性があるということをこのページで説明しますが、 このディレクティブそのものが安全でないわけではなく、これを誤って使 用することが安全でないということを念頭においていてください。

register_globalsをonとした場合、この機能により、HTMLフォームから投 稿される変数と同時に、あらゆる種類の変数がスクリプトに注入される ことになります。これは、PHPにおいては変数の初期化が不 要であるということにも関係し、安全でないコードを書くことが極めて容 易になるということを意味します。困難な変更でしたが、PHPコミュニティ は、このディレクティブをデフォルトで無効とすることを決定しました。 onとした場合、どこから来たのかが不明であり、出処を仮定するしかない 変数を使用することになります。スクリプト自体で定義される内部変数は、 ユーザにより送信されたリクエストデータと混ざってしまいますが、 register_globals を無効とすることでこれを回避することができます。 以下にregister_globalsの誤った使用例を示しましょう。

Example#1 register_globals = on の誤った使用例

```
<?php
// ユーザが認証された場合のみ  $authorized  =  true  を定義する
if (authenticated_user()) {
    $authorized  = true;
}

//  最初に$authorizedをfalseとして初期化していないため、
//  以下のコードにより、GET  auth.php?authorized=1  のように
//  register_globals機能により定義される可能性があります。
// このため、誰でも認証されたようにみせることができます!
if ($authorized) {
     include "/highly/sensitive/data.php";
}
?>
```

register_globals = onとした場合、上記のロジックは破綻する可能性が あります。offの場合、*$authorized*はリクエストに より設定することはできず、正しく動作します。しかし、一般に良いプロ グラミング法は、変数を最初に初期化することです。例えば、上の例で *$authorized = false*を先頭に置いておくことができ ます。これにより、ユーザはデフォルトで認証されない状態となるため、 register_globalsのon/offによらず上記のコードは動作します。

別の例は、 セッションに関するも のです。register_globals = onとした場合、以下の例で *$username*を使用することもできますが、 (URLにより)GETのような他の手段によっても *$username* を設定することができることに留意する 必要があります。

Example#2 register_globals on またはoffの場合のセッションの使用例

```
<?php
//  $usernameの出処は不明だが、$_SESSIONがセッションデータであることは
// 既知です。
if  (isset($_SESSION['username'])) {

     echo "Hello  <b>{$_SESSION['username']}</b>";

} else {

     echo "Hello  <b>Guest</b><br  />";
     echo "Would  you  like  to  login?";

}
?>
```

偽造が試みられた時に警告するために予防的な計測を行うことも可能です。 ある変数の出処を前もって正確に知っている場合、 投稿されたデータが適切な投稿元からのものであるかを調べることができ ます。これは、データが偽造されたものでないことを保証するわけではあ りませんが、攻撃者による偽造の成立を限定的なものにすることができま す。リクエストデータの出処を気にかけない場合、 *$_REQUEST* を使用することができます。 この変数には、GET、POST、COOKIEが合わさって含まれています。 本マニュアルの PHPの外部変数の セクションを参照してください。

Example#3 簡単に変数汚染を検出する

```
<?php
if (isset($_COOKIE['MAGIC_COOKIE'])) {

    // MAGIC_COOKIE comes from a cookie.
     // Be sure to validate the cookie data!

} elseif (isset($_GET['MAGIC_COOKIE']) || isset($_POST['MAGIC_COOKIE'])) {

   mail("admin@example.com", "Possible breakin attempt", $_SERVER['REMOTE_ADDR']);
    echo "Security violation, admin has been alerted.";
    exit;

} else {

   // MAGIC_COOKIE isn't set through this REQUEST

}
?>
```

もちろん、register_globalsをoffにするだけでは、使用するコードが安 全であることを意味しません。投稿された全てのデータ毎に 他の手段で検証することも必要です。ユーザデータを常に検証し、 使用する変数を初期化してください! 初期化されていない変数を調べるには、 error_reporting()で **E_NOTICE**レベルのエラーを有効にするように してください。

register_globalsをOnまたはOffのエミュレートに関数情報に ついては、FAQを 参照してください。

> **注意**: **スーパーグローバル: 使用可能なバージョンに関する注意**PHP 4.1.0 以降、 *$_GET*, *$_POST*, *$_SERVER* 等のスーパーグローバル配列が使用可能となっています。詳細な情報については、マニュアルの superglobals のセクションを参照してください。

ユーザが投稿したデータ

多くの PHP のプログラムで最も脆弱な部分は、言語自体に起因するものではなく、 単にセキュリティを考慮して書かれていないコードの問題です。そのため、 指定したコードの部分の意味を常に時間をかけて吟味し、 予想外の変数が投稿された場合に有り得る損害を確かめる必要があります。

Example#1 危険な変数の使用

```
<?php
// ユーザのホームディレクトリからファイルを削除します... または他の誰
// かのディレクトリかも?
unlink ($evil_var);

// 彼らのアクセスのログを書き込む.. または違うかも?
fputs ($fp, $evil_var);

// 何かちょっとしたことを実行.. または rm -rf *?
system ($evil_var);
exec ($evil_var);

?>
```

常に注意してコードをテストし、Webブラウザから投稿された全ての変数 について次のような点を確認してください。

- このスクリプトは、意図したファイルのみを受け付けるか?
- 例外的なまたは意図したもの以外のデータにより実行することが可能 か?
- このスクリプトは意図した以外の方法で使用することが可能か?
- このスクリプトは、悪い意味で他のスクリプトと組み合わせて使用す ることが可能か?
- トランザクションは適切に記録されているか?

スクリプトを書いた後ではなく、書いている時にこれらの質問を適宜行う ことにより、セキュリティ改善のために不幸にして書き直しが必要になる ということを避けることができます。こうした考慮をまず行うことにより、 システムのセキュリティを保証できるわけではありませんが、改善の一助 にはなりえます。

register_globals,magic_quotes, または他の便利な設定は、有効性、発 信元、指定した変数の値について混乱を生じる可能性があるため、設定を オフにしたいと思うかもしれません。error_reporting(E_ALL) モードで PHPを動作させた場合、確認または初期化する前に使用された変数に関し て警告を発生させることも可能です。(これにより、処理時に通常とは異 なるデータを防止することが可能です)

マジッククオート

目次

警告

この機能は *非推奨* であり、PHP 6.0.0 で *削除* されます。この機能を使用しないことを強く推奨します。

マジッククオートは、PHPスクリプトに入力されるデータを 自動的にエスケープする機能です。 コードでは、マジッククオートをオフにして 実行する際必要な時にデータをエスケープすることが望まれます。

マジッククオートとは

オンの場合、全ての' (シングルクオート), " (ダブルクオート), ¥ (バックスラッシュ)および*NULL* 文字がバックスラッシュで自動的にエスケープされます。 これは、addslashes() の機能と同じです。

3種類のマジッククオートディレクティブを以下に示します。

- magic_quotes_gpc HTTPリクエストデータ(GET, POST, そして COOKIE)に作用します。 実行時に設定することはできません。 PHPのデフォルトは、*on*です。 get_magic_quotes_gpc()も参照してください。
- magic_quotes_runtime 有効な場合、データベースやテキストファイルを含む 外部ソースからデータを返す関数の多くは、 バックスラッシュをクオートでエスケープします。 実行時に設定することができ、PHPでのデフォルトは *off*です。 set_magic_quotes_runtime()および get_magic_quotes_runtime()も参照してください。
- magic_quotes_sybase 有効な場合、シングルクオートはバックスラッシュではなくシングルクオートで エスケープされます。 onの場合、magic_quotes_gpc の指定を完全に上書きします。 これら両方のディレクティブを有効にすると、シングルクオートは ''とエスケープされます。 ダブルクオートやNULLはそのままとなり、エスケープされません。 この値を取得するには、 ini_get()も参照してください。

なぜマジッククオートを使用するのか

- 初心者にとって便利 PHPに実装されたマジッククオートは、初心者により書かれたコードを危険から 守る手助けとなります。 マジッククオートをonにした場合でも SQLインジェクション は可能ですが、そのリスクは減少します。
- 便利 データベースにデータを挿入する際、マジッククオートは暗黙のうちに addslashes()を全てのGET/POST/Cookieデータに 行います。

なぜマジッククオートを使用しないのか

- 移植性 これがonであることを仮定すると、移植性に影響します。 この確認のためにget_magic_quotes_gpc()を 使用し、適切なコーディングを行ってください。
- 性能 エスケープされたデータが全てデータベースに挿入されるわけではないので、 このように全てのデータをエスケープすることは性能を低下させます。 単に(addslashes()のような)エスケープを行う関数を 実行時にコールする方がより効率的です。 *php.ini-dist*はこれらのディレクティブを デフォルトで有効にしていますが、 *php.ini-recommended*はこれを無効にしています。 この推奨は主に性能面によるものです。
- 不便 全てのデータをエスケープする必要はないため、しばしば、 エスケープするべきではないデータまでエスケープされてしまう問題に 悩まされることになります。例えば、フォームからメールを送信する際、 emailの中に多くの \' が含まれることになります。 これを修正するために、stripslashes()を大量に使用することが必要と なる可能性があります。

マジッククオートを無効にする

magic_quotes_gpc ディレクティブはシステムレベルでのみ無効にすることができ、 実行時に行うことはできません。 つまり、ini_set() では指定できません。

Example#1 マジッククオートをサーバ側で無効にする

このディレクティブを*php.ini*で *Off* にする 例を示します。 より詳細については、 設定を変更する方法というタイトルのマニュアルのセクションを 参照してください。

```
; Magic quotes
;

; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = Off

; Magic quotes for runtime-generated data, e.g. data from SQL, from exec(), etc.
magic_quotes_runtime = Off

; Use Sybase-style magic quotes (escape ' with '' instead of ¥').
magic_quotes_sybase = Off
```

サーバの設定を変更できない場合には、 *.htaccess*も使用できます。 例えば、

```
php_flag  magic_quotes_gpc  Off
```

サーバレベルの設定を変更できない場合に対応するといったように、 移植性の高いコード(あらゆる環境で動作するコード)を書く要望に 対して、以下に、 magic_quotes_gpcを実行時に無効にする例を示します。 この方法は非効率であるため、どこかでディレクティブを適切に 設定する方が良いでしょう。

Example#2 マジッククオートを実行時に無効にする

```
<?php
if (get_magic_quotes_gpc()) {
     function stripslashes_deep($value)
     {
         $value = is_array($value) ?
                     array_map('stripslashes_deep', $value) :
                     stripslashes($value);

          return $value;
     }

     $_POST = array_map('stripslashes_deep', $_POST);
     $_GET = array_map('stripslashes_deep', $_GET);
     $_COOKIE = array_map('stripslashes_deep', $_COOKIE);
     $_REQUEST = array_map('stripslashes_deep', $_REQUEST);
}
?>
```

PHPの隠蔽

一般に隠蔽という手段はセキュリティとしては弱いものだと言われています。 しかしこうした手法が望ましい場合もあります。

PHPを隠すための簡単な技法がいくつかあり、システムの弱点を見つけよ うとする攻撃を遅延させることができる可能性があります。*php.ini*ファ イルでexpose_php = offと設定することにより、攻撃者が利用可能な情 報を減らすことが可能です。

他の手段は、ApacheのようなWebサーバをPHPに異なるファイル形式をパー スさせるように設定することです。これは、.htaccessディレクティブまたはApacheの設定ファイル自体で指定します。これにより、紛らわしい ファイル拡張子を使用可能です。

Example#1 PHPを他の言語として隠す

```
# PHPコードを他のコード型のようにする
AddType  application/x-httpd-php  .asp  .py  .pl
```

または、次のように完全に隠すことも可能です。

Example#2 PHP拡張子用に未知の型を使用する

```
# Make PHP code look like unknown types
AddType  application/x-httpd-php  .bop  .foo  .133t
```

または、HTMLコードとして隠すことも可能です。この場合、全てのHTMLファ イルがPHPエンジンを通じてパースされることになるため、若干の性能上の 問題があります。

Example#3 PHP拡張子としてHTML型を使用する

```
# 全てのPHPコードをHTMLのように作成する
AddType  application/x-httpd-php  .htm  .html
```

効率的にこれを使用するには、全てのPHPファイルの名前を上の拡張子に変更 する必要があります。これは、あいまいさに基づく形式のセキュリティですが、 欠点の少ない簡単な防衛策です。

最新版を維持する

PHP は、他の大規模なシステムと同様に、セキュリティを確保しつつ 改良されています。新バージョンにはしばしば大規模あるいは小規模な 変更が含まれています。その内容は、セキュリティの機能追加であったり セキュリティ上の問題・設定の不備・その他システム全体のセキュリティや 安定性にかかわる問題の修正であったりします。

他のシステムレベルのスクリプト言語やプログラムと同様に、最善の アプローチは、頻繁に更新し、最新のバージョンとその変更を 注視し続けることです。

機能

目次

PHP による HTTP 認証

PHP による HTTP 認証のフックは、 Apache モジュールとして実行した時のみ 有効で、CGI 版では利用できません。Apache モジュール上の PHP スクリプトに おいては、header() 関数を使用して "Authentication Required" メッセージをクライアントブラウザに 送ることが可能です。 これにより、クライアントブラウザではユーザー名とパスワードの入力要求 ウインドウがポップアップ表示されます。一度、ユーザーがユーザー名と パスワードを入力すると、PHP スクリプトを含むその URL は、次回以降、 定義済みの変数 *PHP_AUTH_USER* と、 *PHP_AUTH_PW* と、 *PHP_AUTH_TYPE* にそれぞれユーザー名、 パスワード、認証型が代入された状態で呼ばれます。 定義済みの変数は、配列 $_SERVER および *$HTTP_SERVER_VARS* でアクセス可能です。 "Basic" 認証および "Digest" 認証 (PHP 5.1.0 以降) の両者がサポートされています。詳細は、 header()を参照ください。

> **注意**: **PHP バージョンに関する注意** $_SERVERのような スーパーグローバルは、 PHP » 4.1.0 以降で利用可能となりました。 *$HTTP_SERVER_VARS*は、PHP 3以降で利用可能です。

ページ上でクライアント認証を強制するスクリプトの例を以下に示します。

Example#1 Basic HTTP 認証の例

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header("WWW-Authenticate: Basic realm=¥"My Realm¥"");
    header("HTTP/1.0 401 Unauthorized");
    echo "ユーザーがキャンセルボタンを押した時に送信されるテキスト\n";
    exit;
} else {
    echo "<p>こんにちは、{$_SERVER['PHP_AUTH_USER']} さん。</p>";
    echo "<p>あなたは、{$_SERVER['PHP_AUTH_PW']} をパスワードとして入力しました。</p>";
}
?>
```

Example#2 Digest HTTP 認証の例

この例は、シンプルな Digest HTTP 認証スクリプトをどの様に実装するか を示しています。 その他情報については、» RFC 2617 を読んでください。

```
<?php
$realm = 'Restricted area';

//user => password
$users = array('admin' => 'mypass', 'guest' => 'guest');


if (empty($_SERVER['PHP_AUTH_DIGEST'])) {
    header('HTTP/1.1 401 Unauthorized');
    header('WWW-Authenticate: Digest realm="'.$realm.
           '",qop="auth",nonce="'.uniqid().'",opaque="'.md5($realm).'"');

    die('ユーザーがキャンセルボタンを押した時に送信されるテキスト');
}


// PHP_AUTH_DIGEST 変数を精査する
if (!($data = http_digest_parse($_SERVER['PHP_AUTH_DIGEST'])) ||
    !isset($users[$data['username']]))
    die('誤った証明書です!');


// 有効なレスポンスを生成する
$A1 = md5($data['username'] . ':' . $realm . ':' . $users[$data['username']]);
$A2 = md5($_SERVER['REQUEST_METHOD'].':'.$data['uri']);
$valid_response = md5($A1.':'.$data['nonce'].':'.$data['nc'].':'.$data['cnonce'].':'.$data['qop'].':'.$A2);

if ($data['response'] != $valid_response)
    die('誤った証明書です!');

// OK, 有効なユーザー名とパスワードだ
```

```
echo 'あなたは次のユーザーとしてログインしています: ' . $data['username'];

// http auth ヘッダをパースする関数
function http_digest_parse($txt)
{
    // データが失われている場合への対応
    $needed_parts = array('nonce'=>1, 'nc'=>1, 'cnonce'=>1, 'qop'=>1, 'username'=>1, 'uri'=>1, 'response'=>1);
    $data = array();

    preg_match_all('@(¥w+)=(?:([¥'"])([^¥2]+)¥2|([^¥s,]+))@', $txt, $matches, PREG_SET_ORDER);

    foreach ($matches as $m) {
        $data[$m[1]] = $m[3] ? $m[3] : $m[4];
        unset($needed_parts[$m[1]]);
    }

    return $needed_parts ? false : $data;
}
?>
```

> **注意**: **互換性に関する注意** HTTPヘッダ行をコーディングする際には注意を要します。全てのクライアントへの 互換性を最大限に保証するために、キーワード "Basic" には、 大文字の"B"を使用して書くべきです。realm文字列は(一重引用符ではなく) 二重引用符で括る必要があります。また、*HTTP/1.0 401* ヘッダ行のコード *401* の前には、 1つだけ空白を置く必要があります。 認証パラメータは、上のダイジェスト認証の例にあるように カンマ区切りで指定しなければなりません。

単に *PHP_AUTH_USER*および*PHP_AUTH_PW* を出力するのではなく、ユーザー名とパスワードの有効性をチェックしたいと 思うかもしれません。その場合、クエリーをデータベースに送るか、ある dbm ファイル中の ユーザーを調べるといったことをすることになるでしょう。

バグのある Internet Explorer ブラウザには注意してください。このブラ ウザは、ヘッダの順序に関してとてもうるさいようです。今のところ、*HTTP/1.0 401* ヘッダの前に *WWW-Authenticate* ヘッダを送るのが効果があるようです。

PHP 4.3.0 以降、誰かが従来の外部機構による認証を行ってきたページの パスワードを暴くようなスクリプトを書くことを防ぐために、 特定のページに関して外部認証が可能でかつ セーフモード が有効の場合、 PHP_AUTH 変数はセットされません。 この場合、外部認証されたユーザーかどうかを確認するために *REMOTE_USER* 変数、すなわち、 *$_SERVER['REMOTE_USER']* を使用することができます。

> **注意**: **設定上の注意** PHP は、外部認証が動作しているかどうかの判定を *AuthType* ディレクティブの有無で行います。

しかし、上記の機能も、認証を要求されないURLを管理する人が同じサーバー にある認証を要するURLからパスワードを盗むことを防ぐわけではありませ ん。

サーバーからレスポンスコード 401 を受けた際に、Netscape Navigatorおよび Internet Explorer は共にローカルブラウザのウインドウ上の認証キャッシュを 消去します。この機能により、簡単にユーザーを"ログアウト"させ、強制的に ユーザー名とパスワードを再入力させることができます。この機能は、 "タイムアウト" 付きのログインや、"ログアウト" ボタンに適用されています。

Example#3 新規に名前 / パスワードを入力させる HTTP 認証の例

```
<?php
function authenticate() {
    header('WWW-Authenticate: Basic realm="Test Authentication System"');
    header('HTTP/1.0 401 Unauthorized');
    echo "このリソースにアクセスする際には有効なログインIDとパスワードを入力する必要があります。\n";
    exit;
}

if (!isset($_SERVER['PHP_AUTH_USER']) ||
    ($_POST['SeenBefore'] == 1 && $_POST['OldAuth'] == $_SERVER['PHP_AUTH_USER'])) {
    authenticate();
} else {
    echo "<p>Welcome: {$_SERVER['PHP_AUTH_USER']}<br>";
    echo "Old: {$_REQUEST['OldAuth']}";
    echo "<form action='{$_SERVER['PHP_SELF']}' METHOD='POST'>¥n";
    echo "<input type='hidden' name='SeenBefore' value='1'>¥n";
    echo "<input type='hidden' name='OldAuth' value='{$_SERVER['PHP_AUTH_USER']}'>¥n";
    echo "<input type='submit' value='Re Authenticate'>¥n";
    echo "</form></p>¥n";
}
?>
```

この動作は、HTTP Basic 認証の標準に基づいていません。よって、この機能に 依存しないように注意する必要があります。Lynx によるテストの結果、 Lynx は、認証証明書を 401 サーバー応答によりクリアしないことが明らかに なっています。このため、back を押してから foward を再度押すことにより 証明書の要件が変更されない限りリソースをオープンすることができます。 しかし、ユーザは '_' キーを押すことにより認証情報をクリアすることが可能です。

PHP4.3.3 までは、Microsoft の IIS サーバーと CGI 版の PHP の組み合わせでは、 この機能は、IIS の制約により使用することができなかったことにも 注意してください。PHP 4.3.3 以降においてこの機能を使用するには、 IIS の設定の "ディレクトリセキュリティ" の "編集" ボタンを押して "匿名アクセス" のみをオンにしてください。その他のフィールドは オフのままにしてください。

他の制限としては、IIS モジュール (ISAPI) と PHP 4 を使用している場合に、 *PHP_AUTH_** 変数が使用できないことがあります。 しかし、代わりに *HTTP_AUTHORIZATION* を使うことができます。 例えば、次のようなコードを考慮してください。*list($user, $pw) = explode(':', base64_decode(substr($_SERVER['HTTP_AUTHORIZATION'], 6)));*

> **注意**: **IIS に関する注意:** IIS上 で HTTP 認証を使用する場合、PHP の cgi.rfc2616_headers ディレクティブは*0* (デフォルト値) にセットされて いなければなりません。

注意: セーフモード が有効の場合、 *WWW-Authenticate*ヘッダの *realm*部にスクリプトの uid が追加されます。

クッキー(Cookies)

PHP は、HTTP クッキー(Cookie)を完全にサポートします。 クッキーは、リモートブラウザに文字列データを保存したり、 再訪するユーザを特定したりする機構です。 setcookie() か setrawcookie() を使用してクッキーをセットすることができます。 クッキーは HTTP ヘッダの一部なので、 setcookie() はブラウザに何らかの出力を行う前にコールする必要があります。 この制約は、header() に課されているものと同じです。 また、 出力バッファ関数を使用して、 設定するクッキーや送信するヘッダの内容が決まるまで スクリプトからの出力を遅らせることが出来ます。

もし variables_order が "C" を含んでいる場合、クライアントから送られた全てのクッキーは 自動的に $_COOKIE という（常にグローバルな）配列に格納されます。 多数の値を一つのクッキーに割り付けたい場合は、*[]* をクッキー名に加えてください。

register_globals の設定に よっては、クッキーから通常の PHP 変数を作成することも可能です。 しかし、この設定に依存することは推奨されません。なぜなら、セキュリティの 観点からこの機能はオフにされていることが多いからです。 それ以前のバージョンでも、track_vars がセットされていれば *$HTTP_COOKIE_VARS* にクッキーの内容が格納されます。 (この設定は PHP 4.0.3 以降は常にオンになっています)

ちょっとした注意やブラウザのバグといった詳細に関しては、 setcookie() と setrawcookie() を参照してください。

セッション

PHPのセッションサポートは、連続するアクセスを通じて特定のデータを保持する 手段を構成します。この機能により、よりカスタマイズされたアプリケーション の構築が可能となり、Webサイトをより魅力的なものにすることができます。 詳細な情報は、 セッションに関するリファレンス のセクションにあります。

XFormsの処理

» XForms は、従来のWebフォームから 派生したもので、広範なプラットフォームやブラウザそしてPDFドキュメントのような 従来のメディア以外のものにすら使用することができます。

xformsのまず第一の違いは、クライアントへフォームを送信する方法です。 » HTML作成者のためのXForms には、XFormsの作成方法の詳細な解説がありますが、この手引きの趣旨に沿って、 ここでは簡単な例のみ示すことにします。

Example#1 簡単なXForms検索フォーム

```
<h:html  xmlns:h="http://www.w3.org/1999/xhtml"
           xmlns="http://www.w3.org/2002/xforms">
<h:head>
  <h:title>Search</h:title>
  <model>
    <submission  action="http://example.com/search"
                 method="post"  id="s"/>
  </model>
</h:head>
<h:body>
  <h:p>
    <input  ref="q"><label>Find</label></input>
    <submit  submission="s"><label>Go</label></submit>
  </h:p>
</h:body>
</h:html>
```

上のフォームは、(*q* という名前の),テキスト入力ボックス と投稿ボタンを表示します。投稿ボタンがクリックされた時、このフォームは *action*が参照するページに送信されます。

ここからが、Webアプリケーションの視点から異なって見え始めるところです。 通常のHTMLフォームでは、データは、*application/x-www-form-urlencoded* で送信されますが、XFormsでは、この情報はXML形式のデータで送信されます。

XFormsで処理することを選択した場合には、おそらくデータをXML形式で取得したい はずで、この場合、*$HTTP_RAW_POST_DATA* により、 ブラウザが生成したXML ドキュメントにアクセスすることができます。このデータは、使用するXSLTエンジン やドキュメントパーサに渡すことができます。

データ形式には関心がなく、データを従来の データ形式には関心がなく、データを従来の *$_POST*変数に ロードしたいだけの場合、 *method* 属性を *urlencoded-post* に変更することにより、 クライアントブラウザに *application/x-www-form-urlencoded* としてデータを送信するよう指示することができます。

Example#2 XFormにより*$_POST*を送信する

```
<h:html  xmlns:h="http://www.w3.org/1999/xhtml"
           xmlns="http://www.w3.org/2002/xforms">
```

```
<h:head>
  <h:title>Search</h:title>
  <model>
    <submission  action="http://example.com/search"
                 method="urlencoded-post"  id="s"/>
  </model>
</h:head>
<h:body>
  <h:p>
    <input  ref="q"><label>Find</label></input>
    <submit  submission="s"><label>Go</label></submit>
  </h:p>
</h:body>
</h:html>
```

> **注意**: 本稿執筆時点で、多くのブラウザはXFormsをサポートしていません。 上の例が失敗する場合、ブラウザのバージョンを確認してください。

ファイルアップロードの処理

目次

POST メソッドによるアップロード

この機能により、テキスト、バイナリファイルの両方をアップロードできるように なります。 PHP の認証機構およびファイル操作関数を用いて、アップロードを許可する ユーザーとアップロード後にそのファイルを使用して行う動作を完全に制御する ことが可能です。

PHP は、全ての RFC-1867 対応ブラウザ(Netscape Navigator 3 以上、 Microsoft からのパッチをあてた Microsoft Internet Explorer 3 またはパッチ無しのそれ以降の版を含みます)からファイルのアップロードを 受けることができます。

> **注意**: **関係する設定に関する注記** *php.ini*の file_uploads, upload_max_filesize, upload_tmp_dir, post_max_size, max_input_time ディレクティブも参照ください。

PHP は Netscape Composer および W3C の Amaya クライアントにより使用される PUT メソッドによるファイルアップロードもサポートしています。詳細は、PUT メソッドのサポート を参照ください。

Example#1 ファイルアップロード用のフォーム

ファイルアップロード画面は、次のような特別なフォームを作成すること により、作成することができます。

```
<!-- データのエンコード方式である enctype は、必ず以下のようにしなければなりません -->
<form  enctype="multipart/form-data"  action="__URL__"  method="POST">
     <!-- MAX_FILE_SIZE はファイルフィールド用に必須です -->
      <input  type="hidden"  name="MAX_FILE_SIZE"  value="30000" />
     <!-- input 要素の名前が $_FILES 配列での名前となります -->
      このファイルをアップロード: <input  name="userfile"  type="file" />
      <input  type="submit"  value="ファイルを送信" />
</form>
```

上の例の *__URL__* は、PHP ファイルを指すよう置換される 必要があります。

hidden フィールド*MAX_FILE_SIZE* は、 input フィールド file の前に置く必要があります。 この値は、PHP で取得可能なファイルの最大サイズを規定します。この値はバイト数で指定します。 ブラウザ側でこの最大値を出し抜くのは簡単なことなので、この機能を 信頼して、これより大きなサイズのファイルがブロックされることを前提にしては いけません。 しかし、PHP 側の最大サイズの設定を欺くことはできません。 しかしそれでも *MAX_FILE_SIZE* を指定すべきです。なぜなら、 巨大なファイルを転送しようとして、実はそれが大きすぎて 転送できないということを長時間待ったあとで知らされるのを 防げるからです。

> **注意**: アップロード用のフォームが *enctype="multipart/form-data"* 属性を有しているかを 確認してください。さもないと、ファイルのアップロードは動作しません。

オートグローバル $_FILES は、PHP 4.1.0 以降に存在します (*$HTTP_POST_FILES* は、これ以前のバージョンに存在します)。 これらの配列には、全てアップロードされたファイルの情報が 含まれています。

$_FILES の内容は次のようになります。ここでは、上の例のスクリプトで使われたように、 アップロードファイルの名前として *userfile* を 使用することを仮定していることに注意してください。 実際にはどんな名前にすることもできます。

$_FILES['userfile']['name']

> クライアントマシンの元のファイル名。

$_FILES['userfile']['type']

ファイルの MIME 型。ただし、ブラウザがこの情報を提供する場合。 例えば、*"image/gif"* のようになります。 この MIME 型は PHP 側ではチェックされません。そのため、 この値は信用できません。

$_FILES['userfile']['size']

アップロードされたファイルのバイト単位のサイズ。

$_FILES['userfile']['tmp_name']

アップロードされたファイルがサーバー上で保存されているテンポラ リファイルの名前。

$_FILES['userfile']['error']

このファイルアップロードに関する エラーコード *['error']*は、PHP 4.2.0 で追加されました。

php.ini の upload_tmp_dir ディレクティブで 他の場所を指定しない限り、ファイルはサーバーにおけるデフォルトの テンポラリディレクトリに保存されます。サーバーのデフォルトディレクトリは、 PHP を実行する環境において環境変数 TMPDIR を設定する ことにより変更することができます。しかし、PHP スクリプトの内部から putenv() 関数により設定しても上手くいきません。 この環境変数は、アップロードされたファイルに他の処理を行う際にも 同様に使用することが可能です。

Example#2 ファイルのアップロードを検証する

詳細については、is_uploaded_file() および move_uploaded_file() の関数のエントリも 参照ください。以下はフォームからファイルをアップロードするプロセスの例です。

```
<?php
//  4.1.0より前のPHPでは$FILESの代わりに$HTTP_POST_FILESを使用する必要
// があります。

$uploaddir  = '/var/www/uploads/';
$uploadfile  = $uploaddir  . basename($_FILES['userfile']['name']);

echo  '<pre>';
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
     echo "File  is  valid,  and  was  successfully  uploaded.¥n";
} else {
     echo "Possible  file  upload  attack!¥n";
}

echo 'Here  is  some  more  debugging  info:';
print_r($_FILES);

print  "</pre>";

?>
```

アップロードされたファイルを受け取る PHP スクリプトは、アップロード されたファイルを用いて何をするべきかを決めるために必要なロジックを 全て実装する必要があります。例えば、変数 *$_FILES['userfile']['size']* を使用して、小さすぎたり 大きすぎたりするファイルを捨てることができます。指定した型以外の ファイルを全て捨てるために変数 *$_FILES['userfile']['type']* を用いることができますが、 これはあくまでいくつかのチェックの中のひとつとしてのみ実行するように してください。なぜなら、この値を設定するのはあくまでもクライアントであり、 PHP 側では何もチェックしていないからです。 PHP 4.2.0 以降、 *$_FILES['userfile']['error']* を使用することができ、 エラーコード に基づき、ロジックを構成することができます。 何らかの方法により、テンポラリディレクトリからファイルを削除したり 他の場所に移動したりする必要があります。

フォームでアップロードされるファイルが選択されていない場合、PHP は *$_FILES['userfile']['size']* として 0 を返し、*$_FILES['userfile']['tmp_name']* には値を 設定しません。

移動または名前の変更が行われていない場合、リクエストの終了時に そのファイルはテンポラリディレクトリから削除されます。

Example#3 ファイルの配列をアップロードする

PHP はファイルについても HTML フォームで配列を使用すること をサポートしています。

```
<form  action=""  method="post"  enctype="multipart/form-data">
<p>Pictures:
<input  type="file"  name="pictures[]"  />
<input  type="file"  name="pictures[]"  />
<input  type="file"  name="pictures[]"  />
<input  type="submit"  value="Send"  />
</p>
</form>

<?php
foreach  ($_FILES["pictures"]["error"] as $key  => $error) {
     if ($error  == UPLOAD_ERR_OK) {
          $tmp_name  = $_FILES["pictures"]["tmp_name"][$key];
          $name  = $_FILES["pictures"]["name"][$key];
          move_uploaded_file($tmp_name, "data/$name");
     }
}
?>
```

エラーメッセージの説明

PHP 4.2.0 以降、PHP はファイル配列とともに適当なエラーコードを返します。 エラーコードは、PHP によるファイルアップロードの間に生成され、 ファイル配列の*['error']* 要素で アクセス可能です。言い換えると、エラーは、 *$_FILES['userfile']['error']* でアクセス可能でしょう。

UPLOAD_ERR_OK

値: 0; エラーはなく、ファイルアップロードは成功しています。

UPLOAD_ERR_INI_SIZE

値: 1; アップロードされたファイルは、*php.ini* の upload_max_filesize ディレクティブの値を超えています。

UPLOAD_ERR_FORM_SIZE

値: 2; アップロードされたファイルは、HTML フォームで指定された *MAX_FILE_SIZE* を超えています。

UPLOAD_ERR_PARTIAL

値: 3; アップロードされたファイルは一部のみしかアップロードされていません。

UPLOAD_ERR_NO_FILE

値: 4; ファイルはアップロードされませんでした。

UPLOAD_ERR_NO_TMP_DIR

値: 6; テンポラリフォルダがありません。PHP 4.3.10 と PHP 5.0.3 で導入されました。

UPLOAD_ERR_CANT_WRITE

値: 7; ディスクへの書き込みに失敗しました。PHP 5.1.0 で導入されました。

UPLOAD_ERR_EXTENSION

値: 8; ファイルのアップロードが拡張モジュールによって停止されました。 PHP 5.2.0 で導入されました。

注意: これらは PHP 4.3.0 で PHP 定数となりました。

陥りやすい落とし穴

MAX_FILE_SIZE に upload_max_filesize で指定されたファイルサイズより大きなファイルサイズを指定する ことはできません。デフォルトは、2 メガバイトです。

メモリ制限が有効な場合、memory_limit の値をより大きく 設定することが必要となる可能性があります。memory_limit に充分大きな 値を設定するようにしてください。

max_execution_time に設定した値が小さすぎた場合、スクリプトの実行時間がこの値を越える 可能性を生じます。 *max_execution_time* に充分大きな値を設定するように してください。

> **注意**: max_execution_time はスクリプト自身の実行時間にのみ影響します。 スクリプトの実行範囲の外側で発生する動作にかかる時間、つまり、 system() を使ったシステムコールや、 sleep() 関数、データベースに対するクエリー、 ファイルアップロードプロセス、などに費やされた時間はスクリプトの 総実行時間に含まれません。

post_max_size の設定値が 小さすぎた場合、大きなファイルをアップロードすることができなくなります。 *post_max_size* に充分大きな値を設定するように してください。

処理するファイルを検証しない場合、ユーザーが他のディレクトリにある 非公開情報にアクセスできる可能性を生じます。

CERN httpd は、クライアントから得た content-type MIME ヘッダにおいて最初が空白文字で始まるものを切り捨てるようですので注意してください。 このような動作をする限り、CERN httpdは、 ファイルアップロード機能をサポートしないでしょう。

大量のディレクトリー覧のスタイルのせいで、 風変わりな名前（空白を含んでいるとか）のファイルを適切に扱えることは 保証できません。

通常の入力フィールドとファイルアップロードフィールドを （*foo[]* を利用するなどして）同一のフォーム変数で 扱うことはできません。

複数ファイルのアップロード

input で異なった *name* を 使用することにより、複数のファイルをアップロードすることができます。

複数のファイルを一度にアップロードし、自動的にまとめられた情報を 配列で取得することが可能です。これを行うには、HTML フォームで 複数選択可能なセレクトやチェックボックスを指定する際と同様、 配列を用いた投稿用の構文を使用する必要があります。

> **注意**: 複数ファイルのアップロード機能は 3.0.10 以降サポートされました。

Example#1 複数ファイルのアップロード

```
<form  action="file-upload.php"  method="post"  enctype="multipart/form-data">
   Send  these  files:<br />
    <input  name="userfile[]"  type="file"  /><br />
    <input  name="userfile[]"  type="file"  /><br />
    <input  type="submit"  value="Send  files"  />
</form>
```

上記のフォームで投稿された場合、配列 *$_FILES['userfile']*, *$_FILES['userfile']['name']*, *$_FILES['userfile']['size']* (4.1.0 より前のバージョンの PHP の場合は $HTTP_POST_FILES)が設定されま す。 *register_globals* が on の場合、 アップロードファイルに関してグローバル変数も設定されます。これらの各々は、投稿されたファイルに 関する適当な値を有する数値を添字とする配列となります。

例えば、ファイル名が */home/test/review.html* および */home/test/xwp.out* のファイルが 投稿されたとしましょう。この場合、*$_FILES['userfile']['name'][0]* の値が *review.html* となり、 *$_FILES['userfile']['name'][1]* の値が *xwp.out* となります。 同様に、*$_FILES['userfile']['size'][0]* の値が *review.html* のファイルサイズといったように なります。

$_FILES['userfile']['name'][0], *$_FILES['userfile']['tmp_name'][0]*, *$_FILES['userfile']['size'][0]*, *$_FILES['userfile']['type'][0]* も設定されます。

PUT メソッドのサポート

PHP は、クライアントがサーバにファイルを保存するための HTTP PUT メソッドのサポートを提供します。 PUT リクエストは POST リクエストによるファイルアップロードよりもシンプルであり、 次のような形になります。

```
PUT  /path/filename.html  HTTP/1.1
```

これは、通常、リモートクライアントが */path/filename.html* が指す内容を Web ツリーに保存したいことを意味します。 Apache または PHP に おいて全ての人が Web ツリー上の任意のファイルを自動的に上書きできる ようにするというのは明らかに良い発想ではありません。よって、このようなリクエストを処理する際には、まずそのリクエストを処理する特定の PHP スクリプトが必要なことを web サーバに通知する必要があります。 Apache においては、*Script* ディレクティブにより これを行うことができます。これは、Apache 設定ファイルのほぼ任意の 場所に置くことができます。一般的なのは、<Directory> ブロック または <VirtualHost> ブロックの中です。次のように指定します。

```
Script  PUT  /put.php
```

これにより、この行を指定したコンテキストにマッチする URI を有する すべての PUT リクエストが put.php スクリプトに送られるよう Apache に指定します。 もちろん、拡張子 .php により PHP が実行されるよう設定され、 PHP がアクティブであることが必要です。 このスクリプトに対するすべての PUT リクエストの対象リソースは、 スクリプト自身とします。アップロードされるファイルのファイル名ではありません。

PHP 4 以降では、put.php の中で以下のようなことを行います。 これは、アップロードされたファイルをサーバ上のファイル *myputfile.ext* にコピーします。 ファイルコピーを行う前には、 何らかの確認やユーザ認証を行いたくなるかもしれません。

Example#1 PHP 4 で、HTTP PUT されたファイルを保存する

```
<?php
/* PUT されたデータは標準入力からやってきます */
$putdata  = fopen("php://input", "r");

/* 書き込みモードでファイルをオープンします */
$fp = fopen("myputfile.ext", "w");

/* データを 1 KB 単位で読み込み、
   ファイルに書き込みます */
while  ($data  = fread($putdata, 1024))
  fwrite($fp, $data);

/* ストリームを閉じます */
fclose($fp);
fclose($putdata);
?>
```

> **注意**: これ以降のドキュメントは PHP 3 にのみ適用されます。

Example#2 PHP 3 で、HTTP PUT されたファイルを保存する

```
<?php  copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

上記のスクリプトでは、PHP が PUT メソッドのリクエストを受けた際に POST メソッド と全く同様にテンポラリファイルにアップロードされたファイルを保存すると いうことを利用しています。リクエストが終了した際に、テンポラリファイルは 削除されます。よって、PUT を処理する PHP スクリプトは、この ファイルをどこかにコピーする必要があります。このテンポラリファイルの ファイル名は *$PHP_PUT_FILENAME* 変数に保持されており、 *$REQUEST_URI*(Apache 以外の Web サーバでは変わる可能性があります) で指定された送信先ファイル名を得ることができます。送信先ファイル名は、 リモートクライアントが指定したものです。このクライアントの指定を 必ずしも受ける必要はありません。例えば、アップロードされた全ての ファイルを特別なアップロード用ディレクトリにコピーすることも可能です。

リモートファイルの使用

*php.ini*で*allow_url_fopen* を有効にした場合、 ファイル名をパラメータとする関数の多くで HTTP および FTP のURL を使用することができます。 加えて、include(), include_once(), require(), require_once() 命令でURLを使用することができます (PHP 5.2.0 以降では、これらで使用するためには allow_url_include を有効にする必要があります)。 PHPがサポートしているプロトコルに関する詳細は サポートされるプロトコル/ラッパーを参照してください。

> **注意**: PHP 4.0.3以前のバージョンにおいては、URLラッパーを使用するために、 configureオプション --enable-url-fopen-wrapper を使用してPHPをconfigureを行なう必要があります。

> **注意**: PHP 4.3未満のWindows版PHPは次の関数については リモートファイルアクセスをサポートしてません: include(), include_once(), require(), require_once(), そしてイメージ拡張によるimagecreatefromXXX関数。

例えば、リモートWebサーバーにファイルをオープンし、データを出力、デー タベースクエリーに使用するか、単にWebサイトのスタイルに合わせて出力 を行うことが可能です。

Example#1 リモートページのタイトルを得る

```
<?php
$file = fopen("http://www.php.net/", "r");
if (!$file) {
   echo "<p>Unable to open remote file.¥n";
   exit;
}
while (!feof($file)) {
  $line = fgets($file, 1024);
  /* タイトルとタグが同じ行にある場合のみ動作します。 */
  if (eregi("<title>(.*)</title>", $line, $out)) {
    $title = $out[1];
    break;
  }
}
fclose($file);
?>
```

(正しいアクセス権限を有するユーザとして接続した場合には) FTPサーバにファイルを書き込むこともできます。 この方法では、新規ファイルを作成することのみができます。 既存のファイルを上書きしようとした場合には、 fopen()の処理は失敗します。

'anonymous'以外のユーザーで接続を行う場合、URLの中で 'ftp://user:password@ftp.example.com/path/to/file' のように ユーザー名(そして多分パスワードも)指定する必要があります。 (Basic認証を要求された際にHTTP経由でファイルをアクセスする場合と 同じ種類の構文を使用することができます。)

Example#2 リモートサーバーにデータを保存する

```
<?php
$file = fopen ("ftp://ftp.example.com/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Unable to open remote file for writing.¥n";
    exit;
}
/* データをここに書きます。 */
fputs ($file, $_SERVER['HTTP_USER_AGENT'] . "¥n");
fclose ($file);
?>
```

> **注意**: 上の例からリモートログに書きこむためにこの手法を使用することを考えるかも しれません。 しかし残念ながら、リモート上のファイルが既に存在する状態では fopen()をコールすることができないため、 それはできません。 分散ロギングのようなことを行うには、syslog() の使用を考えてみてください。

接続処理

> **注意**: 以下は、PHP 3.0.7 版以降に適用されます。

PHP 内部で、接続ステータスが保持されます。 これは、次の 3 つの状態をとりえます。

- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT

通常 PHP が実行されている場合、NORMAL 状態がアクティブになります。 リモートクライアントが接続を切った場合、ABORTED 状態フラグが有効になります。 通常、リモートクライアントの接続断は、STOP ボタンを押すことにより発生します。 PHP 側の時間制限 (set_time_limit() 参照) にかかった場合、 TIMEOUT 状態フラグが有効になります。

スクリプトを終了させるためにクライアントとの接続を切断するかどうかを 決めることが出来ます。 時々、出力がどのリモートブラウザにも受信され

ない場合でも、 常にスクリプトの実行完了まで実行する方が便利であることがあります。 しかし、デフォルトの動作はリモートクライアントとの接続が断となった際にスクリプト の実行は破棄されます。 この動作は、*php.ini* ディレクティブ ignore_user_abort にて 設定できます。 同様に同じ意味を有する Apache .conf ディレクティブ "php_value ignore_user_abort" または ignore_user_abort() 関数にて設定することも可能です。 PHP にユーザーによる破棄を無視するように設定していない場合、 ユーザが破棄した場合、スクリプトの実行は終了します。 一つの例外は、register_shutdown_function() を用いて シャットダウン関数を定義した場合です。 シャットダウン関数を定義した場合、リモートユーザーが STOP ボタンを押した後、 次にスクリプトが何か出力を行おうとした場合、PHP は接続が破棄されたことを検知し、 シャットダウン関数がコールされます。このシャットダウン関数は、スクリプトが正常に終了した 際にもコールされます。 このため、クライアントが接続を切断した場合に別の動作をさせたい場合に は、connection_aborted() 関数を使用することが 可能です。 この関数は、接続が破棄された場合に、**TRUE** を返します。

スクリプトは、組み込みのスクリプトタイマーによっても終了することが できます。デフォルトのタイムアウトは、30 秒です。 これは、*php.ini*ディレクティブ max_execution_time または同義の Apache .confディレクティブ *php_value max_execution_time* を set_time_limit() 関数と同様に用いることにより 変更することが可能です。タイマーが切れた時、スクリプトは中断されます。 上記のクライアントが接続を切るケースのように シャットダウン関数が登録されている場合、この関数がコールされます。 このシャットダウン関数の中では、 connection_status() 関数のコールにより タイムアウトがシャットダウン関数のコールを生じさせるかどうかを 確認することができます。 この関数は、タイムアウトによりシャットダウン関数がコールされた 場合に 2 を返します。

もう一つ注意すべき点は、状態 ABORTED および TIMEOUT は同時にアクティブにできるということです。 これは、PHP をユーザーによる中断を無視するよう設定した場合に 可能です。 PHP は、ユーザーが接続を破棄しているが、スクリプトは 実行しつづけるということがある可能性があることに注意する 必要があります。 時間制限にかかって中断される場合、もしあればシャットダウン関数が コールされます。 ここで、connection_status() は 3 を返します。

持続的データベース接続

持続的接続は、スクリプトの実行終了時にも閉じられないリンクです。 持続的接続が要求された時、PHPは(前もってオープンされたままになって いる)同じ持続的接続が既にオープンされていないかどうかを確認します。 そして、存在する場合には、それを使用します。存在しない場合には、そ のリンクを作成します。'同じ'接続とは、同じホスト、同じユーザー名、 同じパスワード(利用可能な場合)でオープンされた接続のことを意味しま す。

Webサーバーの動作及び負荷の分散に関して熟知していない人は、持続的接 続において何が行われないかに関してミスを犯す可能性があります。特に、 持続的接続は、同じリンクで'ユーザーセッション'をオープンする機能 やトランザクションを効率的に確立する機能やその他のあらゆる機能を提 供*しません*。つまり、言いたいことを極めて簡単に述べると、持続的接続 は非持続的接続で使用できない いかなる 機能も提 供しません。

なぜ?

これは、Webサーバーの動作により行われるべきものです。Webページを生 成するためにPHPを利用するWebサーバーには、3種類の方法があります。

最初は、CGI "ラッパー"としてPHPを使用する方法です。このように実行し た場合、PHPインタプリタのインスタンスは、Webサーバーに(PHPページに 関する)ページがリクエストされる度に生成され、破棄されます。リクエス ト毎に破棄されるために、(SQLデータベースサーバーへのリンクのような) 必要な全てのリソースは破棄される際にクローズされます。この場合、持 続的接続を使用することから得るものは何もありません。持続的接続は持続しないのです。

2番目は、最も一般的ですが、PHPをマルチプロセスWebサーバー(現在は Apacheのみが含まれます)のモジュールとして実行する方法です。マルチプロセスサーバーは、通常、実際にWebページを送信する複数のプロセス(子) を管理するプロセス(親)を有しています。リクエストがクライアントから 来ると、親プロセスは、他のクライアントにすでに送信を行っていないク ライアントの一つに渡します。このため、同じクライアントが2番目のリク エストをサーバーに送信した際に最初のではなく他の子プロセスにより送 信が行われる可能性があります。 持続的接続がオープンされているとき、SQLサービスにリクエストを行う それぞれのページは SQL サーバへの確立された接続を再利用することが できます。

最後のは、PHPをマルチスレッドWebサーバーのプラグインとして使用する 方法です。現在、PHP 4 は、ISAPI, WSAPI, NSAPI を(Windows上で)サポー トしており、Netscape FastTrack、Microsoftの Internet Information Server (IIS)、O'Reillyの WebSite Proのようなマルチスレッド型サーバのプラグインとしてPHPを使用することが可能です。この場合の動作は前記 のマルチプロセス型モデルと同様です。SAPIはPHP 3では使用できないこと に注意してください。

持続的接続が機能を全く付加しないとしたら、優れている点はなんでしょ う?

答えはかなり簡単です。効率です。持続的接続は、SQLサーバーへ接続する オーバーヘッドが大きい場合には有効です。このオーバーヘッドが実際に大きいがどうかは様々な要因に依存します。例えば、データベースの種類、 Webサーバーが動作するのと同じコンピューターで動作しているか、SQLサー バーを動作させているマシンの負荷、等となります。肝心なのは、接続の オーバーヘッドが高い場合、持続的接続は著しいということです。持続的 接続は、SQLサーバーへの接続を要求するページをリクエスト毎に処理する 代わりに子プロセスが動作中の間一回しかサーバーへの接続を行わないよ うにします。このことは、持続的接続をオープンしたプロセス毎にサーバー への持続的接続をオープンするということになります。例えば、20の異なっ た子プロセスがSQLサーバーへの持続的接続を行うスクリプトを実行した場 合、各子プロセス毎にSQLサーバーへの20の異なった接続が行われます。

しかし、気をつけなければならないことが一つあります。それはデータ ベースへの接続数を制限して使用している場合に、持続的な子プロセスの 接続数がその数を超えると問題が発生し得ることです。もしデータベース の同時接続数の制限が16だとして、サーバに多くのアクセスがあったため 17個の子プロセスが接続しようとするとそのうちの一つは接続に失敗しま す。もしスクリプトにコネクションをシャットダウンしないようなバグ(例えば無限ループ)があると16程度の同時接続しか許容しないデータベース はすぐにダメになってしまいます。使用しているデータベースが、中断さ れた、もし

くは使用されていないコネクションをどのように扱うかを確認 してみてください。

警告

持続的接続を使用する際にはまだいくつか心に留めておく必要がある注意 点があります。一つは持続的接続でテーブルをロックする場合にスクリプトが何らかの理由でロックを外し損ねると、それ以降に実行されるスクリプト がその接続を使用すると永久にブロックしつづけてしまい、ウェブサーバかデータベースサーバを再起動しなければならなくなるということです。もう 一つはトランザクションを使用している場合に、トランザクションブロックが終了する前にスクリプトが終了してしまうと、そのブロックは次に同じ接 続を使用して実行されるスクリプトに引き継がれる、ということです。 どちらの場合でも register_shutdown_function()を使用してテーブルの ロックを解除したりトランザクションをロールバックする簡単なクリーン アップ関数を登録することができます。しかしそれよりも良い方法は、テー ブルロックやトランザクションを使用するスクリプトでは持続的接続を使用 せず、問題を完全に避けて通ることです(他の箇所で使用する分には問題あ りません)。

重要なことをまとめます。持続的接続は、標準的な接続に1対1の割りつけ を行うように設計されています。このことは、*常に* 持続的接続を非持続的接続で置きかえ、かつ動作を変更しないということ ができることを意味します。持続的接続は、スクリプトの効率を変える *かもしれません*(おそらく変えます)が、動作は変更しません!

See also fbsql_pconnect(), ibase_pconnect(), ifx_pconnect(), ingres_pconnect(), msql_pconnect(), mssql_pconnect(), mysql_pconnect(), ociplogon(), odbc_pconnect(), ora_plogon(), pfsockopen(), pg_pconnect(), sybase_pconnect()も参照してください。

セーフモード

目次

PHP のセーフモードは、共有サーバでのセキュリティの問題を解決するための試みです。 この問題を PHP のレベルで解決しようとするのはアーキテクチャ上正しくありません。 しかし、Web サーバや OS レベルでの代替策はあまり現実的ではないため、 多くのユーザ、特に ISP ではセーフモードが現在使用されています。

警告

セーフモードは、PHP 6.0.0 で削除されます。

セキュリティとセーフモード

セキュリティとセーフモード設定ディレクティブ

名前	デフォルト	変更の可否	変更履歴
safe_mode	"0"	PHP_INI_SYSTEM	PHP 6.0.0 で削除。
safe_mode_gid	"0"	PHP_INI_SYSTEM	PHP 4.1.0 から利用可能。PHP 6.0.0 で削除。
safe_mode_include_dir	NULL	PHP_INI_SYSTEM	PHP 4.1.0 から利用可能。PHP 6.0.0 で削除。
safe_mode_exec_dir	""	PHP_INI_SYSTEM	PHP 6.0.0 で削除。
safe_mode_allowed_env_vars	"PHP_"	PHP_INI_SYSTEM	PHP 6.0.0 で削除。
safe_mode_protected_env_vars	"LD_LIBRARY_PATH"	PHP_INI_SYSTEM	PHP 6.0.0 で削除。
open_basedir	NULL	PHP_INI_ALL	PHP < 6 では PHP_INI_SYSTEM。
disable_functions	""	*php.ini* only	PHP 4.0.1 から利用可能
disable_classes	""	*php.ini* only	PHP 4.3.2 から利用可能

PHP_INI_*の定義と詳細については、 ini_set()を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

セーフモードの設定ディレクティブの簡単な説明を以下に示します。

safe_mode boolean

セーフモードを有効にするか否か。

safe_mode_gid boolean

デフォルトでは、セーフモードはオープンしようとするファイルの UIDの比較チェックを行います。GIDの比較にすることでこのチェックを 緩やかなものにしたい場合、safe_mode_gidをオンにしてください。 ファイルにアクセスする際に*UID* (**FALSE**)を使用するか *GID* (**TRUE**)を使用するか制御できます。

safe_mode_include_dir string

このディレクトリ（そのサブディレクトリも含む）の配下のファイルが インクルードされる場合、*UID/GID* のチェックはバイパスされます。（ディレクトリは include_pathの配下であるか あるいはフルパスで記述される必要があります）

PHP 4.2.0以降、このディレクティブは include_pathと同様に コロン(Windowsではセミコロン)で分けた形式で複数のパスを書くことができます。 ここで指定される制限は実はプレフィックスでありディレクトリ名ではありません。 つまり、"safe_mode_include_dir = /dir/incl" と書くと "/dir/include" と "/dir/incls" の両方へのアクセスが許可されます（もし ディレクトリが存在すれば）。指定したディレクトリのみを許可したい 場合には、最後にスラッシュを追加してください。 例："safe_mode_include_dir = /dir/incl/" PHP 4.2.3 と PHP 4.3.3 以降では、ディレクティブの値が空の場合、 異なる *UID/GID* を持つファイルを インクルードすることはできません。 以前のバージョンでは、全てのファイルをインクルード可能でした。

safe_mode_exec_dir string

PHPがセーフモードで動作する場合、system()や その他のプログラム実行関数を、 このディレクトリ以外で起動することは拒否されます。Windowsを含む全ての環境において ディレクトリのセパレータとして/を使用する必要があります。

safe_mode_allowed_env_vars string

ある種の環境変数の設定はセキュリティ上の潜在的な欠陥となりえます。 このディレクティブにはプレフィックスをカンマで区切って書くことができます。 セーフモードでは、ここに書かれたプレフィックスで始まる環境変数だけを ユーザーが変更できるようになります。デフォルトでは、ユーザーは PHP_で始まる名前の環境変数(e.g. PHP_FOO=BAR)だけをセットすることができます。

注意: このディレクティブが空の場合、PHPは全ての環境変数について ユーザーが変更することを許可してしまいます。

safe_mode_protected_env_vars string

putenv()を使ってエンドユーザーが変更するのを 防ぎたい環境変数をカンマ区切りで記述します。ここで設定された環境変数は もしもsafe_mode_allowed_env_varsでは許可されているものであっても 保護されます。

open_basedir string

PHPによってオープンされうるファイルを特定のディレクトリツリー に制限します。このディレクティブはセーフモードのオン/オフに *関わらず* 適用されます。

例えば fopen() や gzopen() を使ってスクリプトがファイルをオープンしようとすると、 そのファイルの位置がチェックされます。 指定されたディレクトリツリーの範囲外にあった場合、PHP はオープンを拒否します。 全てのシンボリックリンクは解決されるので、 シンボリックリンクを使ってこの制限を回避することは不可能です。 ファイルが存在しない場合はシンボリックリンクの解決は行われず、 ファイル名の比較は *open_basedir* に対して行われます。

.は特別な値で、 スクリプトが格納されているワーキングディレクトリをベースディレクトリとして 使用することを意味します。 これは、しかし、スクリプトのワーキングディレクトリが chdir()により容易に変更されるために やや危険です。

*httpd.conf*の中で、open_basedir はオフにすることができます。 (例: 仮想サーバの場合) 他の設定ディレクティブと 同様に "php_admin_value open_basedir none"とします。

Windows上では、ディレクトリをセミコロンで区切ってください。 その他全てのシステムでは、ディレクトリをコロンで区切ってください。Apacheモジュールでは、親ディレクトリからopen_basedirへのパス は自動的に継承されます。

ここで指定される制限は実はプレフィックスでありディレクトリ名ではありません。 つまり、"open_basedir = /dir/incl" と書くと "/dir/include" と "/dir/incls" の両方へのアクセスが許可されます（もし ディレクトリが存在すれば）。指定したディレクトリのみを許可したい場合には、最後にスラッシュを追加してください。 例："open_basedir = /dir/incl/"

注意: 複数のディレクトリへの対応は3.0.7で追加されました。

デフォルトでは全てのファイルのオープンが許可されます。

disable_functions string

セキュリティを考慮したい場合に このディレクティブを使って、特定の関数を無効にすることができます。 関数名をカンマ区切りで表記してください。disable_functionsは セーフモードの影響を受けません。 このディレクティブは *php.ini*上でセットされなければなりません。 例えば、*httpd.conf*等でセットすることはできません。

disable_classes string

セキュリティを考慮したい場合に このディレクティブを使って特定のクラスを無効にすることができます。 クラス名をカンマ区切りで表記してください。disable_classsは セーフモードの影響を受けません。 このディレクティブは *php.ini*上でセットされなければなりません。 例えば、*httpd.conf*等でセットすることはできません。

注意: **Availability note** このディレクティブはPHP4.3.2で追加されました。

register_globals, display_errors, log_errorsも参照してください。

セーフモードがonの場合、PHPは、 現在のスクリプトの所有者がファイル関数により処理されているファイルまたはディレクトリ の所有者に一致するかどうかを調べます。例えば、

```
-rw-rw-r--     1 rasmus    rasmus         33 Jul   1 19:20 script.php
-rw-r--r--     1 root      root         1116 May  26 18:01 /etc/passwd
```

script.php を実行すると、

```
<?php
  readfile('/etc/passwd');
?>
```

セーフモードが有効な場合、以下のようなエラーが出力されます。

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

UID checking. しかし、多くの環境において、厳密な*UID*チェックは 適切ではなく、より緩やかな*GID*チェックで十分です。 これはsafe_mode_gidスイッチで サポートされます。これを*On*にすると制限の緩い *GID*チェックに、*Off*（デフォルト） にすると*UID*チェックになります。

safe_modeの代わりに、 open_basedirディレクトリを セットすると、全てのファイル操作は特定のディレクトリ配下のみに制限されます。 例えば (Apacheのhttpd.confの例):

```
<Directory  /docroot>
   php_admin_value  open_basedir  /docroot
</Directory>
```

open_basedirでセットしたのと 同じscript.phpを実行すると、以下のような結果になります:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

特定の関数を無効にすることもできます。 disable_functionsディレクティブは *php.ini*以外では使用できないことに注意してください。 つまり、*httpd.conf*上のバーチャルホスト毎あるいはディレクトリ毎に 関数を無効にすることはできない、ということになります。 もし*php.ini*ファイルに以下を追加した場合:

```
disable_functions = readfile,system
```

以下のような結果になります:

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

警告

もちろん、これらの PHP の制限はバイナリを実行した場合は有効になりません。

セーフモードにより制限を受けるか無効となる関数

safe-modeにより制限される 関数のリストを示します。ただし、まだ、不完全で、不正確である可能性 があります。

セーフモードで制限される関数

関数	制限
dbmopen()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
dbase_open()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
filepro()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
filepro_rowcount()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
filepro_retrieve()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
ifx_*()	sql_safe_mode restrictions, (!= safe mode)
ingres_*()	sql_safe_mode restrictions, (!= safe mode)
mysql_*()	sql_safe_mode restrictions, (!= safe mode)
pg_lo_import()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
posix_mkfifo()	処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
putenv()	iniディレクティブのsafe_mode_protected_env_vars および safe_mode_allowed_env_varsに依存します。 putenv()のドキュメントも参照ください。
move_uploaded_file()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
chdir()	処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
dl()	この関数は、safe-mode では無効となります。
backtick operator	この関数は、safe-mode では無効となります。
shell_exec() (functional equivalent of backticks)	この関数は、safe-mode では無効となります。
exec()	safe_mode_exec_dir の中でのみ実行可能です。現実的な理由により、現在、実行パスに ..を含めることは許可されていません。 escapeshellcmd()はこの関数の引数にたいして 実行できます。
system()	safe_mode_exec_dir の中でのみ実行可能です。現実的な理由により、現在、実行パスに ..を含めることは許可されていません。 escapeshellcmd()はこの関数の引数にたいして 実行できます。
passthru()	safe_mode_exec_dir の中でのみ実行可能です。現実的な理由により、現在、実行パスに ..を含めることは許可されていません。 escapeshellcmd()はこの関数の引数にたいして 実行できます。
popen()	safe_mode_exec_dir の中でのみ実行可能です。現実的な理由により、現在、実行パスに ..を含めることは許可されていません。 escapeshellcmd()はこの関数の引数にたいして 実行できます。
fopen()	処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
mkdir()	処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
rmdir()	処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
rename()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。 処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
unlink()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。 処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
copy()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。 処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。 (*source* および *target* において)
chgrp()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
chown()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。
chmod()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。 加えて、SUID, SGID, スティキービットを設定する ことはできません
touch()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。 処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。
symlink()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。 処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。 (注意: ターゲットのみが 確認されます)
link()	処理を行うファイル/ディレクトリが実行するスクリプトと同じUIDを有しているかどうかを確認します。 処理を行うディレクトリが実行するスクリプトと同じ UID を有しているかどうかを確認します。 (注意: ターゲットのみが 確認されます)
apache_request_headers()	セーフモードでは、'authorization'で始まるヘッダ(大文字小文字は 区別されません)は返されません。
header()	セーフモードでは、*WWW-Authenticate* (HTTP認証)ヘッダをセットする場合に *realm*パートに スクリプトのUIDがセットされます。
PHP_AUTH variables	セーフモードでは、*PHP_AUTH_USER*, *PHP_AUTH_PW*, *AUTH_TYPE*は *$_SERVER*に含まれません。 にも関わらず、USERに *REMOTE_USER*を使うことはまだ可能です。 (PHP4.3.0以降でのみ影響)

関数	制限

PHP をコマンドラインから使用する

4.3 以降で PHP は *Command Line Interface* を意味する *CLI* という名前の新しい *SAPI* 型 (Server Application Programming Interface)をサポートします。 名前から分かるように、この *SAPI* 型は、 PHP によるシェル(またはデスクトップ)アプリケーションの開発を 主な対象としています。 *CLI SAPI* と他の *SAPI* の間には、 いくつかの違いがあります。本章では、これらについて詳細を説明します。 *CLI* と *CGI* はその振る舞いの多くが 共通であるにもかかわらず、違う SAPI であることに留意してください。

CLI SAPI は、当初 PHP 4.2.0 でリリースされましたが、 この時点では実験的なステータスにあったため、 *./configure* を実行する際に、明示的に --enable-cli を指定することにより、 有効とする必要がありました。PHP 4.3.0 以降、 *CLI SAPI* はもはや実験的なステータスではなくなり、 --enable-cli はデフォルトでオンとなりました。 --disable-cli によりこれを無効とする ことも可能です。

PHP 4.3.0 以降、システムにPHPがどのようにインストールされているかによって CLI/CGI バイナリの名前、位置、存在が異なります。デフォルトでは **make** を実行すると CGI と CLI の両方がビルドされ ソースディレクトリの *sapi/cgi/php* と *sapi/cli/php* にそれぞれ配置されます。両方とも php という名前であることに注意してください **make install** でどうなるかは configure に因ります。 configure で例えば apxs のような SAPI モジュールが選択された場合、または --disable-cgi が指定された場合、 **make install** によって CLI が *{PREFIX}/bin/php* にコピーされます。さもなければ CGI がそこにコピーされます。 既にインストールされている CGI バイナリを上書きしたい場合には、 **make install** の後に **make install-cli** を実行してください。あるいは configure で --disable-cgi を指定することもできます。

> **注意**: --enable-cli と --enable-cgi の両方がデフォルトで有効になっています。そのため、configure で --enable-cli を指定したからといって、**make install** で *{PREFIX}/bin/php* にコピーされるのが必ずしも CLI になるとは限りません。

PHP 4.2.0 から PHP 4.2.3 までの Windows パッケージでは CLI は CGI *php.exe* と同じフォルダに *php-cli.exe* として配布されていました。 PHP 4.3.0 からは Windows パッケージでは CLI は *cli* という 別のフォルダに *php.exe* として配布されます。 したがって *cli/php.exe* となります。 PHP 5 以降、CLI はメインフォルダ内で *php.exe* という名前で配布されます。CGI バージョンは、*php-cgi.exe* として配布されます。

PHP 5 では、新しく *php-win.exe* というファイルが 配布されます。これは CLI バージョンとほぼ同じですが、php-win は何も 出力しないため、コンソールを必要としません（画面に「DOS 窓」が表示 されません）。この振る舞いは、php-gtk と似ています。このためには --enable-cli-win32 オプションを 含めて configure を行います。

> **注意**: **自分の SAPI は何か?** シェルで **php -v** をタイプすると、 *php* が CGI なのか CLI なのかわかります。 php_sapi_name() と定数 **PHP_SAPI** も参照してください。

> **注意**: Unix の *man* ページが PHP 4.3.2 で追加されました。 シェル環境から **man php** とすることで見ることができます。

CLI SAPI を他の *SAPI* と比べた時の 大きな違いを以下に示します。

- *CGI SAPI* と異なり、ヘッダが出力されません。

  CGI SAPI は HTTP ヘッダの出力を抑制する機能を 提供していますが、等価な機能は *CLI SAPI* では サポートされていません。

  デフォルトでは CLI は静寂モードで起動されます。古い CGI スクリプトと互換性を 保って使えるように -q スイッチが残されています。

  動作するディレクトリはスクリプトの場所に変更されることはありません （-C および --no-chdir スイッチは 互換性のために残されています）。

  エラーメッセージはプレーンテキストで表示されます （HTML でフォーマットされません）。

- 以下に示すいくつかの *php.ini* ディレクティブは、*CLI SAPI* により上書きされます。これは、シェル環境では意味が ないためです。

上書きされる *php.ini* のディレクティブ

ディレクティブ	*CLI SAPI* のデフォルト値	コメント
html_errors	**FALSE**	エラーメッセージに含まれる *HTML* タグは シェル上では意味がなく、可読性をかなり低下させるため、この ディレクティブはデフォルトで **FALSE** となっています。
implicit_flush	**TRUE**	print(), echo() および 関連するものによる全ての出力は、直ちに出力され、バッファに キャッシュされないことが望ましいと言えます。この場合でも、 標準出力を保留または操作したい場合には、 output buffering を 使用することが可能です。
max_execution_time	0 (unlimited)	シェル環境では、PHP を際限なく使用できる ようにするために、最大実行時間の制限は無しに設定されています。 Web 用アプリケーションは数秒単位で実行されるよう作られて いますが、シェルアプリケーションの実行時間は、これよりかなり 長くなる傾向があります。
register_argc_argv	**TRUE**	*CLI SAPI* を使用している場合、グローバル PHP 変数 *$argc* (アプリケーションに渡される引数の数)と *$argv* (引数の値の配列)は常に登録され、 適切な値が代入されます。 PHP 4.3.0以降、*CLI SAPI* を使用するときに PHP の *$argc* 変数と *$argv* 変数が登録され、適切な値がセットされます。 このバージョンより前では、*CGI* や *MODULE* がこれらの変数を生成するには PHP の register_globals ディレクティブがオンになっている必要がありました。 バージョンや register_globals の設定がどうであろうと、 $_SERVER または *$HTTP_SERVER_VARS* は常に使用可能です。 例：*$_SERVER['argv']*

注意: これらのディレクティブは、設定ファイル *php.ini* またはカスタム 設定ファイル(指定した場合のみ)で他の値に初期化できません。 この制限は、これらのデフォルト値が全ての設定ファイルをパースした後に 適用されるためです。しかし、これらの値は実行時に変更することが 可能です (上記のディレクティブの全てについてこれが当てはまるわけでは ありません。例えば、register_argc_argv)。

- シェル環境での動作を容易とするために、以下の定数が定義されています。

CLI 固有の定数

定数	説明

STDIN	*stdin* へのオープン済みのストリーム。 これにより、以下のようにオープンする必要がなくなります。
	`<?php`
	`$stdin = fopen('php://stdin', 'r');`
	`?>`
	stdin から1行読み込みたい場合、以下のようにします。
	`<?php`
	`$line = trim(fgets(STDIN)); // STDIN から 1 行読み込む`
	`fscanf(STDIN, "%d¥n", $number); // STDIN から数値を読み込む`
	`?>`

STDOUT	*stdout* へのオープン済みのストリーム。 これにより、以下のようにオープンする必要がなくなります。
	`<?php`
	`$stdout = fopen('php://stdout', 'w');`
	`?>`

STDERR	*stderr* へのオープン済みのストリーム。 これにより、以下のようにオープンする必要がなくなります。
	`<?php`
	`$stderr = fopen('php://stderr', 'w');`
	`?>`

上記のように、*stderr* のようなストリームを自分で オープンする必要はなく、以下のようにストリームリソースの代わりに 定数を使用するだけでかまいません。

```
php -r 'fwrite(STDERR, "stderr¥n");'
```

これらのストリームを明示的に閉じる必要はありません。これは、 PHP により自動的に行われます。

注意: これらの定数は、PHP スクリプトを *stdin* から読み込んだ場合は使用できません。

- *CLI SAPI* は、実行されるスクリプトのディレクトリに カレントディレクトリを変更*しません*！

CGI SAPI との違いを示す例を以下に示します。

```
<?php
// シンプルなテストアプリケーション
echo getcwd(), "¥n";
?>
```

CGI 版により実行した場合、出力は以下のようになります。

```
$ pwd
/tmp
```

```
$ php-cgi -f another_directory/test.php
/tmp/another_directory
```

これは、PHP が実行するスクリプトのディレクトリに カレントディレクトリを変更することを明らかに示しています。

CLI SAPI を使用した場合の出力は次のようになります。

```
$ pwd
/tmp

$ php -f another_directory/test.php
/tmp
```

これにより、PHP でシェルツールを書く際の柔軟性をより大きくすることができます。

注意: *CGI SAPI* は、この *CLI SAPI* の動作をコマンドライン実行時のスイッチ *-C* によりサポートしています。

PHP バイナリにより提供されるコマンドラインオプションの一覧は、 -h スイッチを指定して PHP を実行することにより いつでも調べることができます。

```
Usage: php [options] [-f] <file> [--] [args...]
         php [options] -r <code> [--] [args...]
          php [options] [-B <begin_code>] -R <code> [-E <end_code>] [--] [args...]
          php [options] [-B <begin_code>] -F <file> [-E <end_code>] [--] [args...]
         php [options] -- [args...]
        php [options] -a

  -a                 Run interactively
  -c <path>|<file> Look for php.ini file in this directory
  -n                 No php.ini file will be used
  -d foo[=bar]       Define INI entry foo with value 'bar'
  -e                   Generate extended information for debugger/profiler
  -f <file>          Parse and execute <file>.
  -h                 This help
  -i                 PHP information
  -l                 Syntax check only (lint)
  -m                 Show compiled in modules
  -r <code>          Run PHP <code> without using script tags <?..?>
  -B <begin_code>   Run PHP <begin_code> before processing input lines
  -R <code>          Run PHP <code> for every input line
  -F <file>          Parse and execute <file> for every input line
  -E <end_code>     Run PHP <end_code> after processing all input lines
  -H                   Hide any passed arguments from external tools.
  -s                   Display colour syntax highlighted source.
  -v                 Version number
  -w                   Display source with stripped comments and whitespace.
  -z <file>          Load Zend extension <file>.

  args...             Arguments passed to script. Use -- args when first argument
                      starts with - or script is read from stdin

  --ini              Show configuration file names

  --rf <name>        Show information about function <name>.
  --rc <name>        Show information about class <name>.
  --re <name>        Show information about extension <name>.
  --ri <name>        Show configuration for extension <name>.
```

CLI SAPI は、実行する PHP コードを 取得するために三種類の異なる手段をサポートしています。

1. PHP に特定のファイルの実行を指示する。

   ```
   php my_script.php

   php -f my_script.php
   ```

   上記の方法は共に(-f スイッチの使用の如何に関らず) 指定したファイル *my_script.php* を実行します。 実行ファイルとしてあらゆるファイルを指定することができ、 PHP スクリプトは拡張子 *.php* で終わる必要がなく、任意の名前や拡張子を 使用することができます。

   注意: -f スイッチを使用している時にスクリプトに引数を渡したい場合は、 最初の引数として -- を渡す必要があります。

2. 実行する PHP コードをコマンドラインで直接指定する。

   ```
   php -r 'print_r(get_defined_constants());'
   ```

   シェル変数の置換と引用符の使用については特に注意してください。

注意: この例をよくみてください。開始/終了タグがありません! -r スイッチを使用した場合、これらのタグは不要と なります。これらのタグを使用するとパーサエラーを発生します。

3. 実行する PHP コードを標準入力 (*stdin*)で指定する。

   これは強力な機能で、以下の(仮想的な)例に示すように、動的に PHP コードを生成し、実行バイナリに入力すること ができます。

```
$ some_application | some_filter | php | sort -u >final_output.txt
```

これらのコードを実行する三種類の方法を組み合わせて使用することは できません。

他のシェルアプリケーションのように、PHP バイナリに 引数を指定することができるだけでなく、PHP スクリプトが この引数を取得することも可能です。スクリプトに指定できる 引数の数は PHP による制限を受けません (シェルは指定可能な文字数の最大値を設定しています。通常、この制限値を越えることはできません)。スクリプトに指定した引数は、グローバル配列 *$argv* でアクセス可能です。 添字 0 は、常にスクリプト名が含まれています (PHP コードが標準入力またはコマンドラインスイッチ *-r* により指定された場合、スクリプト名は - となります)。 登録される第 2 のグローバル変数は *$argc* で (スクリプトに指定された引数の数ではなく)、配列 *$argv* の要素数が含まれます。

スクリプトに指定する引数が文字 - で始まっていない 限り、特に留意すべきことはありません。スクリプトに指定する引数が文字 - で始まる場合、PHP 自体がこれを パースする必要があるとみなすため、問題を発生します。 これを防止するため、引数リストセパレータ -- を 使用してください。PHP にパースされる引数の後に このセパレータを置くと、その後の全ての引数はそのままパースされずに スクリプトに渡されます。

```
# これは、指定したコードを実行せずに PHP の使用法を示します
$ php -r 'var_dump($argv);' -h
Usage: php [options] [-f] <file> [args...]
[...]

# これは '-h' を引数として解釈し、PHP の使用法を表示しません
$ php -r 'var_dump($argv);' -- -h
array(2) {
  [0]=>
  string(1) "-"
  [1]=>
  string(2) "-h"
}
```

また、PHP をシェルスクリプトとして使用する他の手段があります。 最初の行が *#!/usr/bin/php* で始まり、 PHP の開始/終了タグの中に通常の PHP コードが続くスクリプトを書き、適当なファイル 実行属性を設定する(例: **chmod +x test**)ことが可能です。 この方法は、通常のシェル/Perl スクリプトと同様に実行することができます。

Example#1 シェルスクリプトとしての PHP スクリプトの実行

```
#!/usr/bin/php
<?php
var_dump($argv);
?>
```

このファイルの名前が *test* で、カレントディレクトリに あるとすると、以下のように実行することができます。

```
$ chmod +x test
$ ./test -h -- foo
array(4) {
  [0]=>
  string(6) "./test"
  [1]=>
  string(2) "-h"
  [2]=>
  string(2) "--"
  [3]=>
  string(3) "foo"
}
```

見て分かるように、- で始まるスクリプトのパラメータを 指定する際に、特に注意する必要はありません。

長いオプション指定は、PHP 4.3.3 以降で利用可能です。

コマンドラインオプション

オプション	長いオプション	説明
-a	--interactive	PHP を対話的に実行します。PHP のコンパイル時に Readline 拡張モジュール（Windows 版では利用できません）を含めた場合、補完（例: 変数名の一部を 入力して TAB キーを押すと、PHP が完全な変数名に変換する）・ 矢印キーによるコマンド履歴 などの高度な機能が利用できます。コマンドの履歴は *~/.php_history* ファイルに保存されます。 **注意**: auto_prepend_file および auto_append_file で インクルードされたファイルはこのモードでもパースされますが、 いくつかの制限があります。例えば、関数はそれがコールされる前に 定義されていなければなりません。 **注意**: オートローディング は、PHP を CLI 対話シェルで実行している場合は使用できません。

-c	--php-ini	このオプションを使用することにより、*php.ini* を探すディレクトリを 指定したり、カスタマイズされた *INI* ファイル (*php.ini* という名前である必要はありません)を直接指定する ことが可能です。例：
		
		`$ php -c /custom/directory/ my_script.php`
		`$ php -c /custom/directory/custom-file.ini my_script.php`
		
		このオプションを指定しない場合、ファイルは、 デフォルトの位置 で探索されます。

-n	--no-php-ini	*php.ini* を完全に無視します。このスイッチは、PHP 4.3.0 以降で 利用可能です。

-d	--define	このオプションにより *php.ini* で指定できる設定ディレクティブに カスタム値を設定することができます。構文は以下のようになります。
		
		`-d configuration_directive[=value]`
		
		例 (レイアウト上の理由により、長い行が折りたたまれています):
		
		`# 値の部分を省略すると、設定ディレクティブに"1"を指定します`
		`$ php -d max_execution_time`
		`-r '$foo = ini_get("max_execution_time"); var_dump($foo);'`
		`string(1) "1"`
		
		`# 空の値を渡すと設定ディレクティブに""を設定します`
		`php -d max_execution_time=`
		`-r '$foo = ini_get("max_execution_time"); var_dump($foo);'`
		`string(0) ""`
		
		`# 設定ディレクティブは文字'='の後に指定したものを設定します`
		`$ php -d max_execution_time=20`
		`-r '$foo = ini_get("max_execution_time"); var_dump($foo);'`
		`string(2) "20"`
		`$ php`
		`-d max_execution_time=doesntmakesense`
		`-r '$foo = ini_get("max_execution_time"); var_dump($foo);'`
		`string(15) "doesntmakesense"`

-e	--profile-info	デバッガ/プロファイラ用の拡張情報を出力します。
-f	--file	-f オプションに指定したファイル名をパースし、 実行します。このスイッチはオプションで省略することもできます。 実行するスクリプトを指定するだけで充分です。 **注意**: 引数をスクリプトに渡すには、最初の引数を -- としなければなりません。 そうしないと、PHP は渡された内容を PHP のオプションとみなしてしまいます。
-h と -?	--help と --usage	このオプションを使用すると、実際の一連のコマンドラインオプションと 各1行の説明が情報を取得できます。
-i	--info	このコマンドラインオプションは、phpinfo() をコールし、 結果を出力します。PHP が正しく動作していない場合、 **php -i** を実行し、情報テーブルの前または中に 出力されるエラーメッセージを調べることをお勧めします。 出力は、*HTML* 形式で行なわれるため、かなり 量が多くなることに注意してください。
-l	--syntax-check	このオプションにより、指定した PHP コードの 構文チェックのみを簡単に行なうことができます。成功した場合、 テキスト *No syntax errors detected in <filename>* が標準出力に書き込まれ、リターンコードは *0* となります。失敗した場合、テキスト *Errors parsing <filename>* に加え、内部パーサエラーメッセージ が標準出力に書き込まれ、シェルリターンコードは、 *255* となります。 このオプションは、(未定義の関数のような)致命的なエラー(fatal error) はみつけません。致命的なエラーについても調べ

オプション	長いオプション	説明

PHP 実行バイナリは、Web サーバから完全に独立して PHP スクリプトを 実行するために使用することができます。Unix システムを使用している場合、 実行可能とするためには PHP スクリプトの先頭に特別な一行を追加する必要が あります。これにより、システムがそのスクリプトを実行するプログラムを 知ることができます。 Windows 環境では、*.php* ファイルのダブルクリック オプションに *php.exe* を関連づけることができます。 または、PHP によりスクリプトを実行するバッチファイルを作成することも 可能です。Unix 上で動作させるためにスクリプトに追加された先頭行は、Windows 環境での動作に悪影響を与えません。このため、この手法により、 クロスプラットフォーム環境で動作するプログラムを書くことができます。 コマンドライン PHP プログラムの書方の簡単な例を以下に示します。

Example#8 コマンドラインから実行されることを意図したスクリプト(script.php)

```
#!/usr/bin/php
<?php

if ($argc != 2 || in_array($argv[1], array('--help', '-help', '-h', '-?'))) {
?>
```

これは、ひとつのオプションをとるコマンドラインの PHP スクリプトです。

```
  使用法:
  <?php echo $argv[0]; ?> <option>

  <option> は出力したい単語です。
  --help, -help, -h, あるいは -? を指定すると、
  このヘルプが表示されます。

<?php
} else {
    echo $argv[1];
}
?>
```

上のスクリプトでは、特殊な先頭行が用いられており、このファイルが PHP により実行されることを示しています。ここでは CLI 版を使用しているため、 HTTP ヘッダは出力されません。PHP で コマンドラインアプリケーションを使用する際には、2 つの変数 *$argc* および *$argv* を使用することができます。 最初の変数は、引数の数に 1 (実行中のスクリプトの名前)を加えたものです。 2 番目の変数は、引数を保持する配列で、スクリプト名を有する 要素 0 (*$argv[0]*) から始まっています。

上のプログラムでは、引数が 1 より少ないかまたは多いかを調べています。 また、引数が --help, -help, -h または -? の場合、 ヘルプメッセージを出力し、動的にスクリプト名を出力します。 他の引数を受け取った場合、これを出力します。

上のスクリプトを Unix で実行する場合、実行可能とした後、 **script.php echothis** または **script.php -h** とする必要があります。 Windows では、この処理を行なう以下のようなバッチファイルを作成することが できます。

Example#9 コマンドライン PHP スクリプトを実行するバッチファイル(script.bat)

```
@C:¥php¥php.exe  script.php  %1  %2  %3  %4
```

上のプログラムが *script.php* という名前であるとし、 *c:¥php¥php.exe* に *php.exe* があるとすると、このバッチファイルは、追加オプション **script.bat echothis** または **script.bat -h** を指定して、スクリプトを実行します。

PHP のコマンドラインアプリケーションを拡張するために使用できる その他の関数については、拡張モジュール Readline に関する ドキュメントも参照してください。

関数リファレンス

目次

- SCA 関数
- SDO 関数
- SDO XML データアクセスサービス関数
- SDO リレーショナルデータアクセスサービス関数
- セマフォ・共有メモリおよび IPC 関数(semaphore)
- SESAM データベース関数
- PostgreSQL セッション保存ハンドラ
- セッション処理関数(session)
- 共有メモリ関数(shmop)
- SimpleXML関数
- SNMP 関数
- SOAP関数
- ソケット関数
- Standard PHP Library (SPL) 関数
- SQLite 関数
- SQLite 関数 (PDO_SQLITE)
- Secure Shell2 関数
- 統計関数
- ストリーム関数
- Strings(文字列関数)
- Subversion 関数
- Shockwave Flash 関数
- Swish 関数
- Sybase 関数
- TCP ラッパ関数（TCP Wrappers）
- Tidy 関数
- Tokenizer 関数
- Unicode 関数
- URL 関数
- 変数操作関数(Variable Handling)
- Verisign Payflow Pro 関数
- vpopmail 関数
- W32api 関数
- WDDX 関数
- win32ps 関数
- win32service 関数
- xattr 関数
- xdiff 関数
- XML パーサ関数
- XML-RPC 関数
- XMLReader 関数
- XMLWriter 関数
- XSL 関数
- XSLT 関数
- YAZ 関数
- YP/NIS 関数
- Zip ファイル関数
- zlib 圧縮関数

See also 拡張モジュールの分類.

.NET 関数

導入

警告

この拡張モジュールは、 *実験的* なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

---

dotnet_load

(No version information available, might be only in CVS)

dotnet_load — DOTNET モジュールをロードする

説明

int **dotnet_load** (string `$assembly_name` [, string `$datatype_name` [, int `$codepage`]])

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

変更履歴

バージョン	説明
4.1.0	パラメータ *codepage* が追加されました。

Apache専用の関数

導入

以下の関数は、Apacheモジュール版のPHPを実行している場合のみ利用可能です。

> **注意**: PHP 4.3.2以降、PATH_TRANSLATED は、 Apache 2 SAPIでは暗黙のうちに設定されなく なりました。これは、Apacheにより設定されない場合に サーバ変数SCRIPT_FILENAMEと同じ値に設定される Apache 1とは異なります。この変更は、PATH_TRANSLATEDは PATH_INFOが定義されている場合のみ存在するべきであるという CGIの規定を満たすために行われました。Apache 2のユーザは、PATH_INFOを定義するために *httpd.conf*の中で *AcceptPathInfo = On*を使用してください。

インストール手順

PHPのApacheへのインストール方法については、インストールの章を参照してください。

実行時設定

Apache PHPモジュールの動作は、*php.ini*の設定により影響を受けます。 *php.ini*の設定は、サーバの設定ファイル内の php_flag の設定 またはローカルなファイル *.htaccess*により上書きすることができます。

Example#1 *.htaccess*によりあるディレクトリのPHPによるパー スを無効にする

```
php_flag  engine  off
```

Apache設定オプション

名前	デフォルト	変更の可否	変更履歴
engine	"1"	PHP_INI_ALL	PHP 4.0.5 から利用可能
child_terminate	"0"	PHP_INI_ALL	PHP 4.0.5 から利用可能
last_modified	"0"	PHP_INI_ALL	PHP 4.0.5 から利用可能
xbit_hack	"0"	PHP_INI_ALL	PHP 4.0.5 から利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

設定ディレクティブの短い説明を以下に示します。

`engine` boolean

> PHP によるパースのオン/オフを切り替えます。 このディレクティブは、Apacheモジュール版のPHPでのみ有効です。 このディレクティブは、ディレクティブ毎または仮想サーバ毎にPHPに よるパースを有効または無効にしたいサイトで使用されます。 *httpd.conf*ファイルの適当な場所に **`engine off`** を置くことにより、PHPを有効また は無効にすることができます。

`child_terminate` boolean

リクエストの終了時にPHPスクリプトが子プロセスの終了を指定するか どうかを指定します。 apache_child_terminate()も参照してください。

last_modified boolean

PHPスクリプトの修正日をこのリクエストのLast-Modified:ヘッダとし て送信します。

xbithack boolean

PHPがファイル終端を無視して実行ビットが設定されているファイルを パースするようにします。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

apache_child_terminate

(PHP 4 >= 4.0.5, PHP 5)

apache_child_terminate — このリクエストの後にApacheプロセスを終了する

説明

bool **apache_child_terminate** (void)

apache_child_terminate() は、カレントのPHPリク エストを実行しているApacheプロセスをリクエスト終了時点で終了しま す。この関数は、メモリ消費量が大きなスクリプトを実行した後プロセ スを使用するために使用することが可能です。これは、メモリは通常内 部的にのみ解放され、オペレーティングシステムに戻されないためです。

返り値

もし PHP が Apache 1 モジュールとして実行している場合、**TRUE** を返します。 この Apache バージョンはマルチスレッドバージョンではなく、child_terminate PHP ディレクティブは有効です (デフォルトは無効) 。 もしこれらの条件に適合しない場合 **FALSE** が返され、エラーレベル **E_WARNING** が発生します。

注意

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- exit()

---

apache_get_modules

(PHP 4 >= 4.3.2, PHP 5)

apache_get_modules — ロードされた Apache モジュールのリストを取得する

説明

array **apache_get_modules** (void)

ロードされた Apache モジュールのリストを取得します。

返り値

ロードされた Apache モジュールの配列 array を返します。

変更履歴

バージョ	説明

ン

5.0.0	Apache 1 を使用している、もしくは Apache 2 で フィルター API として使用している場合に利用可能です。 以前は、Apache 2 の ハンドラ API として使用している場合にのみ利用可能でした。

例

Example#1 apache_get_modules() の例

```
<?php
print_r(apache_get_modules());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => core
     [1] => http_core
     [2] => mod_so
     [3] => sapi_apache2
     [4] => mod_mime
     [5] => mod_rewrite
)
```

---

apache_get_version

(PHP 4 >= 4.3.2, PHP 5)

apache_get_version — Apache のバージョンを取得する

説明

string **apache_get_version** (void)

Apache のバージョンを取得します。

返り値

成功時は Apache のバージョン、失敗時は **FALSE** を返します。

変更履歴

バージョン	説明
4.3.4	Apache 1 で利用可能です。
5.0.0	Apache 2 の フィルター API で利用可能です。

例

Example#1 apache_get_version() の例

```
<?php
$version = apache_get_version();
echo "$version¥n";
?>
```

上の例の出力は、たとえば以下のようになります。

```
Apache/1.3.29 (Unix) PHP/4.3.4
```

参考

- phpinfo()

---

apache_getenv

(PHP 4 >= 4.3.0, PHP 5)

apache_getenv — Apache の subprocess_env 変数を取得する

説明

string **apache_getenv** (string `$variable` [, bool `$walk_to_top`])

variable で指定された Apache 環境変数を取得します。

この関数は Apache 2 を必要とします。それ以外では定義されていません。

パラメータ

variable

Apache の環境変数を指定します。

walk_to_top

全ての Apache レイヤについてのトップレベル変数を取得するかどうか を指定します。

返り値

成功時は Apache 環境変数の値、失敗時は **FALSE** を返します。

例

Example#1 apache_getenv() の例

この例は、Apache の環境変数 *SERVER_ADDR* をどの様に取得するかを示しています。

```
<?php
$ret = apache_getenv("SERVER_ADDR");
echo $ret;
?>
```

上の例の出力は、たとえば以下のようになります。

```
42.24.42.240
```

参考

- apache_setenv()
- getenv()
- スーパーグローバル

---

apache_lookup_uri

(PHP 4, PHP 5)

apache_lookup_uri — リクエストの一部を指定したURIに対して行い、全ての情報を返す

説明

object **apache_lookup_uri** (string `$filename`)

この関数は、URIにリクエストの一部を行います。このリクエストは指定した リソースに関する全ての重要な情報を得るのに十分なものです。

この関数は、 PHP が apache モジュールとしてインストールされた場合のみサポートされます。

パラメータ

filename

リクエストされているファイル名 (URI)。

返り値

URI に関する情報のオブジェクトです。返される object のプロパティは次のようなものです。

- status

- the_request
- status_line
- method
- content_type
- handler
- uri
- filename
- path_info
- args
- boundary
- no_cache
- no_local_copy
- allowed
- send_bodyct
- bytes_sent
- byterange
- clength
- unparsed_uri
- mtime
- request_time

例

Example#1 apache_lookup_uri() の例

```
<?php
$info = apache_lookup_uri('index.php?var=value');
print_r($info);

if (file_exists($info->filename)) {
    echo 'file exists!';
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
stdClass Object
(
    [status] => 200
    [the_request] => GET /dir/file.php HTTP/1.1
    [method] => GET
    [mtime] => 0
    [clength] => 0
    [chunked] => 0
    [content_type] => application/x-httpd-php
    [no_cache] => 0
    [no_local_copy] => 1
    [unparsed_uri] => /dir/index.php?var=value
    [uri] => /dir/index.php
    [filename] => /home/htdocs/dir/index.php
    [args] => var=value
    [allowed] => 0
    [sent_bodyct] => 0
    [bytes_sent] => 0
    [request_time] => 1074282764
)
file exists!
```

---

apache_note

(PHP 4, PHP 5)

apache_note — Apacheリクエスト記号(note)を取得/設定する

説明

string **apache_note** (string `$note_name` [, string `$note_value`])

apache_note()はApache専用の関数であり、 リクエストの*notes*テーブルから値を得たり、 値を設定します。

パラメータ

note_name

記号の名前

note_value

記号の値

返り値

引数が1つだけ指定されてコールされた場合、 現在の記号の値*note_name*が返されます。 引数が2つ指定されてコールされた場合、記号*note_name* の値を*note_value*にセットし、前の記号 *note_name*の値を返します。 もし記号が処理できない場合、**FALSE** が返されます。

---

apache_request_headers

(PHP 4 >= 4.3.0, PHP 5)

apache_request_headers — HTTPリクエストヘッダを全て取得する

説明

array **apache_request_headers** (void)

カレントのリクエストにおける全てのHTTPヘッダを取得します。

この関数は、 PHP が apache モジュールとしてインストールされた場合のみサポートされます。

返り値

カレントのリクエストにおける全てのHTTPヘッダの連想配列、もしくは 失敗時は **FALSE** を返します。

例

Example#1 apache_request_headers() の例

```
<?php
$headers = apache_request_headers();

foreach ($headers as $header => $value) {
    echo "$header: $value <br />¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: www.example.com
Connection: Keep-Alive
```

注意

注意: PHP 4.3.0より前では、**apache_request_headers()**は、 getallheaders()と呼ばれていました。PHP 4.3.0以 降、getallheaders() は、 **apache_request_headers()**へのエイリアスとなって います。

注意: 環境変数を読み込むことにより、共通のCGI変数の値を取得することも可 能です。これは、PHPがApacheモジュール として使用されているかどうかにはよらず動作します。 利用可能な環境変数 の一覧を見るには、phpinfo()を使用してください。

注意: PHP 4.3.3 以降、 Netscape/iPlanet/SunONE Web サーバの NSAPI サーバモジュール でもこの関数を使用できます。

参考

- apache_response_headers()

---

apache_reset_timeout

(PHP 5 >= 5.1.0)

apache_reset_timeout — Apache の書き込みタイマーをリセットする

説明

bool **apache_reset_timeout** (void)

apache_reset_timeout() は Apache の書き込みタイマーをリセットします。デフォルトは 300 秒です。 *set_time_limit(0);* *ignore_user_abort(true)* と定期的な **apache_reset_timeout()** をコールすることで、 理論的に Apache を永遠に実行することができます。

この関数は Apache 1 を必要とします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数は、safe-mode では無効となります。

---

apache_response_headers

(PHP 4 >= 4.3.0, PHP 5)

apache_response_headers — HTTPレスポンスヘッダを全て取得する

説明

array **apache_response_headers** (void)

全てのApacheレスポンスヘッダを配列として取得します。

返り値

全てのApacheレスポンスヘッダの配列、もしくは **FALSE** を返します。

例

Example#1 apache_response_headers() の例

```
<?php
print_r(apache_response_headers());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
      [Accept-Ranges] => bytes
      [X-Powered-By] => PHP/4.3.8
)
```

注意

> **注意**: PHP 4.3.3 以降、 Netscape/iPlanet/SunONE Web サーバの NSAPI サーバモジュール でもこの関数を使用できます。

参考

- apache_request_headers()
- headers_sent()

---

apache_setenv

(PHP 4 >= 4.2.0, PHP 5)

apache_setenv — Apacheサブプロセスの環境変数を設定する

説明

bool **apache_setenv** (string `$variable` , string `$value` [, bool `$walk_to_top`])

apache_setenv() は *variable* で指定された Apache 環境変数の値を設定します。

> **注意**: Apache 環境変数を設定しても、対応する $_SERVER の値は変更されません。

パラメータ

variable

> 設定する環境変数

value

> *variable* の新しい値

walk_to_top

> 全ての Apache レイヤに対して有効なトップレベルの変数を設定するかどうか

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apache_setenv() を使った Apache 環境変数の設定

```
<?php
apache_setenv("EXAMPLE_VAR", "Example  Value");
?>
```

注意

> **注意**: **apache_setenv()** は、分割されたページや PHP スクリプトによって取り込まれた Server Side Includes (.shtml) に渡すための変数を設定する場合、 apache_getenv() と対にして使用することができます。

参考

- apache_getenv()

---

ascii2ebcdic

(No version information available, might be only in CVS)

ascii2ebcdic — ASCIIからEBCDICに文字列を変換する

説明

int **ascii2ebcdic** (string $ascii_str)

ascii2ebcdic() は、EBCDICに基づくオペレーティング システム (OS/390, BS2000)でのみ使用可能なApache専用の関数です。 この関数は、ASCII エンコードされた文字列 *ascii_str* を等価なEBCDIC表現(バイナリ対応)に 変換し、結果を返します。

パラメータ

ascii_str

> 変換される ASCII 文字列

返り値

ASCII 文字列を表す EBCDIC を返します。

参考

- ebcdic2ascii()

---

ebcdic2ascii

(No version information available, might be only in CVS)

ebcdic2ascii — EBCDICからASCIIに文字列を変換する

説明

int **ebcdic2ascii** (string `$ebcdic_str`)

ebcdic2ascii() は、EBCDICに基づくオペレーティング システム (OS/390, BS2000)でのみ使用可能なApache専用の関数です。 この関数は、EBCDICンコードされた文字列 *`ebcdic_str`* を等価なASCII表現(バイナリ対応)に 変換し、結果を返します。

パラメータ

`ebcdic_str`

変換される EBCDIC 文字列

返り値

EBCDIC 文字列を表す ASCII を返します。

参考

- ascii2ebcdic()

---

getallheaders

(PHP 4, PHP 5)

getallheaders — 全てのHTTPリクエストヘッダを取得する

説明

array **getallheaders** (void)

全てのHTTPリクエストヘッダを取得します。

この関数は、 apache_request_headers()のエイリアスです。 この関数は、カレントのリクエストにおける全てのHTTPヘッダーを有する 連想配列を返します。この関数の動作に関する詳細については、 apache_request_headers()のドキュメントを参照して ください。

この関数は、 PHP が apache モジュールとしてインストールされた場合のみサポートされます。

返り値

全てのHTTPリクエストヘッダの連想配列、もしくは失敗時に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	apache_request_headers()のエイリアスとなりました。 これは、基本的に関数名の変更にあたります。この関数は、 Apache モジュールの場合のみ動作します。

注意

注意: PHP 4.3.3 以降、 Netscape/iPlanet/SunONE Web サーバの NSAPI サーバモジュール でもこの関数を使用できます。

参考

- apache_response_headers()

---

virtual

(PHP 4, PHP 5)

virtual — Apache サブリクエストを実行する

説明

bool **virtual** (string `$filename`)

virtual() は、mod_include の *<!--#include virtual...-->* と似ている Apache 用関数です。 この関数は、Apache サブリクエストを実行します。 CGI スクリプトまたは .shtml ファイル、Apache を通して解釈を行う 他のものがある場合にこの関数は有用です。 CGI スクリプトの場合、そのスクリプトは、有効な CGI ヘッダを生成する 必要があることに注意してください。 最低でも、Content-type ヘッダを生成する必要があります。

サブリクエストを実行するには、全てのバッファを終了、ブラウザへフラッシュし、 待機状態のヘッダも送信しておきます。

この関数は、 PHP が apache モジュールとしてインストールされた場合のみサポートされます。

パラメータ

`filename`

virtual コマンドが実行されるファイル

返り値

成功時は virtual コマンドの実行、失敗時は **`FALSE`** を返します。

変更履歴

バージョン	説明
4.0.6	この関数を PHP ファイルに使用することができます。 しかし、他の PHP ファイルをインクルードする必要があるのなら、 include() または require() を 使用するほうがベターです。

注意

警告

クエリ文字列をインクルードされるファイルに渡す事ができますが、 *$_GET* は親スクリプトからコピーされ、 *$_SERVER['QUERY_STRING']* は渡されたクエリ文字列に なります。クエリ文字列は、Apache 2 を使用している場合の見渡されます。 要求されたファイルは Apache アクセスログに出力されません。

> **注意**: 要求されたファイルの中で設定された環境変数は、呼び出し元の スクリプトからは見えません。
>
> **注意**: PHP 4.3.3 以降、 Netscape/iPlanet/SunONE Web サーバの NSAPI サーバモジュール でもこの関数を使用できます。

目次

Alternative PHP Cache (APC)

導入

Alternative PHP Cache (APC) は、PHP の実行コードをキャッシュする仕組みで、 フリーかつオープンに使用できます。PHP の中間コードのキャッシュ・最適化を 行うためのフリーでオープン、かつ堅牢なフレームワークを提供するという 考えのもとに作られています。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/apc.

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

> **注意**: Windows 版の APC では、temp パスが存在し、 Web サーバから書き込み可能になっていることが必要です。 APC は環境変数 TMP、TEMP、USERPROFILE の内容をこの順に調べ、 どれも設定されていない場合は WINDOWS ディレクトリを使用します。

> **注意**: さらに深く踏み込んだ、実装についての高度な技術情報は、 » developer-supplied TECHNOTES file を参照ください。

実行時設定

php.ini の設定により動作が変化します。

たいていの場合はデフォルトの APC 設定でうまく動作しますが、 きちんとチューニングをしたい場合は以下のパラメータを設定します。

あなたが決めなければいけないことは、以下の 2 つです。 まず APC にどれくらいの共有メモリを設定するかということ、そして、 ファイルの更新チェックをリクエストのたびに APC が行うかどうかということです。 これらに関連する ini ディレクティブが *apc.shm_size* および *apc.stat* です。 これらのディレクティブについて、 以下の説明を注意深くお読みください。

サーバを起動したら、この拡張モジュールに含まれているスクリプト *apc.php* をドキュメントルート以下に配置し、 ブラウザでアクセスしてください。 キャッシュの状態についての詳細な情報がここで得られます。 PHP で GD が使用可能になっている場合は、きれいなグラフも表示されます。 まず最初にチェックすべきなのは、当然、 実際にファイルがキャッシュされているかどうかでしょう。 実際に動作していることを確認したら、次は左側にある *Cache full count* の値に注目しましょう。 これは、キャッシュがいっぱいになったために強制削除が行われた (直近の *apc.ttl* 秒間にアクセスされなかったエントリが、 キャッシュから削除された) 回数を表します。 この値ができるだけ小さくなるようにキャッシュを設定しなければなりません。 キャッシュが絶えずいっぱいになっているようだと、 パフォーマンスに影響を及ぼします。 この場合は、APC に割り当てるメモリの量を増やすか、キャッシュするスクリプトを絞り込むために *apc.filters* を使用します。

APC の設定オプション

名前	デフォルト	変更の可否	変更履歴
apc.enabled	"1"	PHP_INI_SYSTEM	APC 2 で PHP_INI_SYSTEM。APC <= 3.0.12 で PHP_INI_ALL。
apc.shm_segments	"1"	PHP_INI_SYSTEM	
apc.shm_size	"30"	PHP_INI_SYSTEM	
apc.optimization	"0"	PHP_INI_ALL	
apc.num_files_hint	"1000"	PHP_INI_SYSTEM	
apc.user_entries_hint	"4096"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.ttl	"0"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.user_ttl	"0"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.gc_ttl	"3600"	PHP_INI_SYSTEM	
apc.cache_by_default	"1"	PHP_INI_ALL	APC <= 3.0.12 で PHP_INI_SYSTEM。APC 3.0.0 以降で利用可能。
apc.filters	NULL	PHP_INI_SYSTEM	
apc.mmap_file_mask	NULL	PHP_INI_SYSTEM	
apc.slam_defense	"0"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.file_update_protection	"2"	PHP_INI_SYSTEM	APC 3.0.6 以降で利用可能。
apc.enable_cli	"0"	PHP_INI_SYSTEM	APC 3.0.7 以降で利用可能。
apc.max_file_size	"1M"	PHP_INI_SYSTEM	APC 3.0.7 以降で利用可能。
apc.stat	"1"	PHP_INI_SYSTEM	APC 3.0.10 以降で利用可能。
apc.write_lock	"1"	PHP_INI_SYSTEM	APC 3.0.11 以降で利用可能。
apc.report_autofilter	"0"	PHP_INI_SYSTEM	APC 3.0.11 以降で利用可能。
apc.include_once_override	"0"	PHP_INI_SYSTEM	APC 3.0.12 以降で利用可能。
apc.rfc1867	"0"	PHP_INI_SYSTEM	APC 3.0.13 以降で利用可能。
apc.rfc1867_prefix	"upload_"	PHP_INI_SYSTEM	
apc.rfc1867_name	"APC_UPLOAD_PROGRESS"	PHP_INI_SYSTEM	
apc.rfc1867_freq	"0"	PHP_INI_SYSTEM	
apc.localcache	"0"	PHP_INI_SYSTEM	APC 3.0.14 以降で利用可能。
apc.localcache.size	"512"	PHP_INI_SYSTEM	APC 3.0.14 以降で利用可能。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

apc.enabled boolean

apc.enabled を 0 にすることで APC を無効にできます。 APC が静的にコンパイルされて PHP に組み込まれており、 他に無効にする手段がない場合などに有用です (DSO としてコンパイルされている場合は、 単に *php.ini* の中の *extension* という行をコメントアウトするだけで無効にできます)。

apc.shm_segments integer

コンパイラキャッシュのために割り当てる共有メモリセグメントの数。 APC が割り当て済みの共有メモリを使い切ってしまっているが、 すでにシステムが許す限り *apc.shm_size* を拡大しているといった場合に、この値を大きくすることを試みます。

apc.shm_size integer

個々の共有メモリセグメントの大きさを MB 単位で指定します。デフォルトで、 共有メモリセグメントの大きさが非常に小さく設定されているシステムもあります (大半の BSD 系システムがこれに含まれます)。

apc.optimization integer

最適化レベル。ゼロは最適化を無効にし、 値を大きくするほど最適化のレベルが高くなります。 わずかながら速度の向上が期待できます。この項目は実験的なものです。

apc.num_files_hint integer

Web サーバで読み込まれるソースファイルの総数についての 「ヒント」。よくわからない場合はゼロを指定するか、 単に無視してください。何千ものソースファイルを扱っているようなサイトで有用です。

apc.user_entries_hint integer

apc.num_files_hint と同様に、 保存するユーザキャッシュ変数の数についての「ヒント」。 よくわからない場合は、ゼロを設定するか無視してください。

apc.ttl integer

キャッシュされているエントリが、 他のエントリに割り当てられるまでスロットに残っていることの可能な秒数。 ゼロのままにしておくと、キャッシュの中身が古いエントリでいっぱいになってしまい、 新しいエントリがキャッシュできなくなります。

apc.user_ttl integer

ユーザキャッシュエントリが、 他のエントリに割り当てられるまでスロットに残っていることの可能な秒数。 ゼロのままにしておくと、キャッシュの中身が古いエントリでいっぱいになってしまい、 新しいエントリがキャッシュできなくなります。

apc.gc_ttl integer

キャッシュエントリがガベージコレクションのリストに残り続ける秒数。 ソースファイルのキャッシュ中にサーバプロセスが死んだ場合の安全装置となります。 ソースファイルが変更された場合、メモリに割り当てられている古いバージョンは、 この TTL に達するまで再読み込みされません。 この機能を無効にするにはゼロを設定します。

apc.cache_by_default boolean

デフォルトで On です。しかし、これを Off にして *+* で始まる *apc.filters* とともに使用することで、 フィルタに一致したファイルのみをキャッシュすることが可能です。

apc.filters string

カンマで区切られた、POSIX 拡張正規表現のリスト。 ソースファイル名がいずれかのパターンにマッチした場合、そのファイルはキャッシュされません。 マッチングに使用されるファイル名は include/require に渡される名前であり、 絶対パスではないことに注意しましょう。正規表現が *+* で始まっている場合、その条件にマッチするファイルはキャッシュされます。 また - で始まっている場合は、 条件にマッチするファイルはキャッシュされません。 デフォルトは - なので、これは省略可能です。

apc.mmap_file_mask string

--enable-mmap を用いて MMAP サポートつきでコンパイルされている場合、ここで mktemp 形式のファイルマスクを指定します。mmap モジュールは、 mmap されたメモリ領域をファイルに置くか共有メモリに置くかを、 これによって判断します。 ファイルベースの mmap を使用するには、この値を */tmp/apc.XXXXXX* （正確に 6 つの *X*）のように指定します。 POSIX 形式の shm_open/mmap を使用するには、*.shm* をマスクのどこかで指定します。例: */apc.shm.XXXXXX* 。 また、*/dev/zero* を指定することで、カーネルの */dev/zero* インターフェースを使用した anonymous mmap を使用することもできます。未定義の場合は、この方式が用いられます。

apc.slam_defense integer

非常にアクセスの多いサーバでは、 サーバを起動したりファイルを書き換えたりするたびに 「多くのプロセスが」「同時に」「同じファイルを」 キャッシュしようとすることになります。このオプションを指定すると、 ある一定のパーセンテージでファイルをキャッシュせずに利用するようにします。 あるいは、単一のプロセスがキャッシュ処理をスキップする確率と考えることもできます。 たとえば、*apc.slam_defense* を *75* に設定すると、プロセスがキャッシュされていないファイルをキャッシュする処理を 75% の確率で抑えられます。つまり、この値を大きく設定することで、 キャッシュ処理の混雑を防ぐことが可能です。値を *0* に設定すると、この機能が無効になります。

非推奨です。かわりに apc.write_lock を使用しましょう。

apc.file_update_protection integer

稼動中の Web サーバ上のファイルを書き換える場合、それを原始的（atomic） な手段で行うべきです。つまり、まずいったん一時ファイルに書き込み、 準備ができた時点でそれをリネーム（*mv*） して正しい位置に移動します。多くのテキストエディタや cp、tar その他のプログラムはこの方式ではありません。 ということは、ファイルの書き込み中にそのファイルがアクセスされる （そしてキャッシュされる）可能性があるわけです。 *apc.file_update_protection* は、 新しいファイルをキャッシュするまでの遅延を設定します。デフォルトは 2 秒で、ファイルの更新時刻（*mtime*）がアクセス時刻と 2 秒未満しか違わない場合はファイルをキャッシュしないという意味です。 更新の最中のファイルにアクセスしてしまった不幸な人には 変なデータが見えてしまいますが、 少なくともその変な状態がキャッシュされてしまうことはありません。 rsync などの原始的な更新を保証する方式を利用することがわかっている場合は、 値を 0 に設定することでこの機能を無効にできます。 更新処理に 2 秒以上かかるようなシステムを利用している場合は、 この値をもう少し大きくしたくなるかもしれません。

apc.enable_cli integer

たいていは、テストやデバッグ用に使用します。これを設定すると CLI バージョンの PHP で APC を有効にします。通常は、すべての CLI リクエストに対して APC キャッシュを作成/格納/削除したいとは思わないでしょう。 しかし、CLI バージョンの APC を簡単に作成できるようにしておくことは、 多くのテストシナリオで有用です。

apc.max_file_size integer

この値よも大きなサイズのファイルは、キャッシュされません。 デフォルトは 1M です。

apc.stat integer

この設定を変更する場合は十分注意してください。デフォルト設定は On で、 これは、ファイルが変更されていないかどうかを スクリプトの実行のたびに APC が調べ、 もし変更されていれば、再コンパイルして新しいバージョンをキャッシュします。 この設定を Off にすると、変更されているかどうかがチェックされません。 つまり、変更内容を有効にするには Web サーバを再起動する必要があるということです。 実運用環境ではコードを変更することはめったにないでしょうから、 この設定を Off にしておくことでパフォーマンスを大きく向上させられます。

included/require されたファイルについてもこのオプションは適用されますが、 もし相対パス (Unix の場合は / で始まらないすべてのパス) の include を使用している場合は、ファイルを一意に識別するために APC がチェックする必要があります。 絶対パスの includes を使用している場合、 APC 絶対パスをファイルの識別子として使用し、 チェックを飛ばすことができます。

apc.write_lock boolean

多くの処理が実行されるサーバでは、最初にサーバを立ち上げたときや 多くのファイルを変更した場合などに、 すべてのプロセスが同一のファイルをコンパイルしたりキャッシュしたりしてしまうことがあります。 write_lock を有効にすると、ひとつのプロセスのみが 未キャッシュのスクリプトをコンパイルするようになります。 その間、他のプロセスはロック待ちをするのではなく キャッシュされていない状態で実行します。

apc.report_autofilter boolean

バインド時の問題によりキャッシュから自動的に除外されたスクリプトを記録します。

apc.include_once_override boolean

include_once() および require_once() を最適化し、コストの高いシステムコールの使用を避けるようにします。

apc.rfc1867 boolean

RFC1867 のファイルアップロード進捗ハンドラが有効になるのは、 PHP 5.2.0 以降で APC をコンパイルした場合のみです。 これを有効にすると、ファイルアップロードフォームの file フィールドの前に *APC_UPLOAD_PROGRESS* というフィールドがある場合に APC が自動的にユーザキャッシュエントリ upload_*key* を作成します。ここで、*key* はフォームの *APC_UPLOAD_PROGRESS* エントリの値となります。

現時点では、ファイルアップロードの追跡はスレッドセーフではないことに注意しましょう。 前のアップロード処理が終わる前に別のアップロードを開始すると、 前のアップロードの追跡が無効になってしまいます。

Example#1 apc.rfc1867 の例

```
<?php
print_r(apc_fetch("upload_$_POST[APC_UPLOAD_PROGRESS]"));
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [total] => 1142543
     [current] => 1142543
     [rate] => 1828068.8
     [filename] => test
     [name] => file
      [temp_filename] => /tmp/php8F
     [cancel_upload] => 0
    [done] => 1
)
```

apc.rfc1867_prefix string

rfc1867 のアップロード進捗処理機能で作成するユーザキャッシュエントリの キーにつけるプレフィックスを指定します。

apc.rfc1867_name string

APC のアップロード進捗処理機能を有効にするフォームの hidden 項目名、そしてユーザキャッシュエントリのキーのサフィックスを指定します。

apc.rfc1867_freq string

アップロードの進捗を記録するユーザキャッシュエントリの更新頻度を指定します。 ファイルサイズに対するパーセンテージ、あるいはファイルサイズで指定します。 サイズを指定する場合は、最後に 'k'、'm' あるいは 'g' を指定することでそれぞれキロバイト、メガバイト、ギガバイトを指定できます (大文字小文字は区別しません)。 0 を指定すると、可能な限り進捗を更新するようにします。 これは、アップロードの速度を低下させてしまいます。

apc.localcache boolean

これは、ロックが不要なローカルプロセスのシャドウキャッシュを有効にします。 これにより、キャッシュが書き込まれる際のロックの競合を軽減します。

apc.localcache.size integer

ローカルプロセスのシャドウキャッシュの大きさ。 ある程度大きなな値を設定しておく必要があります。目安としては apc.num_files_hint の半分程度となります。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

apc_add

(PECL apc:3.0.13-3.0.14)

apc_add — 変数をデータ領域にキャッシュする (保存されていない場合のみ)

説明

bool **apc_add** (string $key , mixed $var [, int $ttl])

注意: PHP の他の多くの仕組みと異なり、**apc_add()** を用いて格納された変数はリクエストを超えて （その値がキャッシュから取り除かれるまで）持続します。

パラメータ

key

この名前を用いて変数を格納します。*key* は キャッシュ内で一意です。そのため、**apc_add()** を使用して同一の *key* で新しい値を格納しようとしても、それは保存されません。 かわりに **FALSE** が返されます (これが、**apc_add()** と apc_store() の唯一の相違点です)。

var

格納する変数。

ttl

有効期間。*var* は、キャッシュに *ttl* 秒間だけ格納されます。 *ttl* が経過すると、格納されている変数は （次のリクエスト時に）キャッシュから削除されます。 *ttl* が指定されていない（あるいは *ttl* が *0* の場合）は、 キャッシュから手動で削除される・あるいはキャッシュに存在できなくなる （clear, restart など）まで値が持続します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apc_add() の例

```
<?php
$bar = 'BAR';
apc_add('foo', $bar);
var_dump(apc_fetch('foo'));
echo "¥n";
$bar = 'NEVER GETS SET';
apc_add('foo', $bar);
var_dump(apc_fetch('foo'));
echo "¥n";
?>
```

上の例の出力は以下となります。

```
string(3) "BAR"
string(3) "BAR"
```

参考

- apc_store()

- apc_fetch()
- apc_delete()

---

apc_cache_info

(PECL apc:2.0-3.0.9)

apc_cache_info — APC のデータから、キャッシュされた情報（およびメタデータ）を取得する

説明

array **apc_cache_info** ([string $cache_type])

返り値

キャッシュされたデータ（およびメタデータ）の配列を返します。 失敗した場合には **FALSE** を返します。

> **注意**: もし APC キャッシュのデータを取得できなかった場合、 **apc_cache_info()** は警告を発生します。これが起こるのは、 一般的には APC が有効になっていない場合などです。

パラメータ

cache_type

> *cache_type* が "*user*" の場合はユーザキャッシュの情報が返されます。 それ以外の場合はシステムキャッシュ (キャッシュされたファイル) の情報が返されます。

limited

> *limited* が **TRUE** の場合は、 返り値にキャッシュエントリの個々の一覧が含まれません。 これは、統計情報の収集時に呼び出しを最適化したい場合などに有用です。

変更履歴

バージョン	説明
3.0.11	パラメータ *limited* が追加されました。

例

Example#1 apc_cache_info() の例

```
<?php
print_r(apc_cache_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [num_slots] => 2000
     [ttl] => 0
     [num_hits] => 9
     [num_misses] => 3
     [start_time] => 1123958803
     [cache_list] => Array
        (
             [0] => Array
                (
                       [filename] => /path/to/apc_test.php
                      [device] => 29954
                      [inode] => 1130511
                      [type] => file
                      [num_hits] => 1
                       [mtime] => 1123960686
                       [creation_time] => 1123960696
                      [deletion_time] => 0
                       [access_time] => 1123962864
                      [ref_count] => 1
                      [mem_size] => 677
                )
              [1] => Array (...iterates for each cached file)
)
```

参考

- APC 設定ディレクティブ

---

apc_clear_cache

(PECL apc:2.0-3.0.9)

apc_clear_cache — APC キャッシュをクリアする

説明

bool **apc_clear_cache** ([string `$cache_type`])

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

パラメータ

`cache_type`

> *cache_type* が "*user*" の場合はユーザキャッシュがクリアされます。 それ以外の場合はシステムキャッシュ (キャッシュされたファイル) がクリアされます。

---

apc_compile_file

(PECL apc:3.0.13-3.0.14)

apc_compile_file — ファイルをバイトコードキャッシュに保存し、すべてのフィルタをバイパスする

説明

bool **apc_compile_file** (string `$filename`)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

パラメータ

`filename`

> コンパイルし、バイトコードキャッシュに保存したい PHP ファイルへの完全パスあるいは相対パス。

---

apc_define_constants

(PECL apc:3.0.0-3.0.9)

apc_define_constants — 定数の組を定義し、それを取得あるいは一括定義する

説明

bool **apc_define_constants** (string `$key` , array `$constants` [, bool `$case_sensitive`])

ご存知のとおり、define() は非常に遅いです。 APC を使用する主な利点はスクリプト/アプリケーションのパフォーマンスの改善なので、 大量の定数を定義する手順を合理化するために、この仕組みが提供されています。 しかし、この関数は期待通りの動作をしません。

よりよい解決策として、PECL の » hidef 拡張モジュールを試してみましょう。

> **注意**: （キャッシュ全体をクリアすることなく）格納されている定数を削除するには、 *`constants`* に空の配列を渡します。そうすれば、そこに格納されていた値は事実上削除されます。

パラメータ

`key`

key は、格納される定数群の名前を表します。 この *key* は、格納されている定数を apc_load_constants() で取得するために使用されます。

constants

constant_name => value 形式の連想配列。 *constant_name* は、一般の 定数 の命名規則に従う 必要があります。 *value* は、スカラ値でなければ なりません。

case_sensitive

デフォルトでは、定数名の大文字・小文字は区別されます。すなわち、 *CONSTANT* と *Constant* は別の値を表します。このパラメータを **FALSE** にすると、 定数名の大文字・小文字は区別されなくなります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apc_define_constants() の例

```
<?php
$constants  =  array(
     'ONE'    => 1,
     'TWO'    => 2,
     'THREE'  => 3,
);
apc_define_constants('numbers', $constants);
echo ONE, TWO, THREE;
?>
```

上の例の出力は以下となります。

```
123
```

参考

- apc_load_constants()
- define()
- constant()
- あるいは PHP リファレンスの「定数」

---

apc_delete

(PECL apc:3.0.0-3.0.9)

apc_delete — 格納されている変数をキャッシュから取り除く

説明

bool **apc_delete** (string $key)

パラメータ

key

（apc_store() を用いて）値を格納する際に 使用された *key* 。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apc_delete() の例

```
<?php
$bar = 'BAR';
apc_store('foo', $bar);
apc_delete('foo');
// もちろん、このような使い方は無意味
?>
```

参考

- apc_store()
- apc_fetch()

---

apc_fetch

(PECL apc:3.0.0-3.0.9)

apc_fetch — 格納されている変数をキャッシュから取得する

説明

mixed **apc_fetch** (string $key)

パラメータ

key

（apc_store() を用いて）値を格納する際に 使用された *key* 。

返り値

成功した場合に格納されていた変数、失敗した場合に **FALSE** を返します。

例

Example#1 apc_fetch() の例

```
<?php
$bar = 'BAR';
apc_store('foo', $bar);
var_dump(apc_fetch('foo'));
?>
```

上の例の出力は以下となります。

```
string(3) "BAR"
```

参考

- apc_store()
- apc_delete()

---

apc_load_constants

(PECL apc:3.0.0-3.0.9)

apc_load_constants — 定数群をキャッシュから読み込む

説明

bool **apc_load_constants** (string $key [, bool $case_sensitive])

パラメータ

key

取得したい定数群（apc_define_constants() を使用して格納されたもの）。

case_sensitive

デフォルトでは、定数名の大文字・小文字は区別されます。すなわち、 *CONSTANT* と *Constant* は別の値を表します。このパラメータを **FALSE** にすると、 定数名の大文字・小文字は区別されなくなります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apc_load_constants() の例

```
<?php
$constants  =  array(
    'ONE'    => 1,
    'TWO'    => 2,
    'THREE'  => 3,
);
apc_define_constants('numbers', $constants);
apc_load_constants('numbers');
echo ONE, TWO, THREE;
?>
```

上の例の出力は以下となります。

```
123
```

参考

- apc_define_constants()
- define()
- constant()
- あるいは PHP リファレンスの「定数」

---

apc_sma_info

(PECL apc:2.0-3.0.9)

apc_sma_info — APC の共有メモリ割り当てに関する情報を取得する

説明

array **apc_sma_info** ([bool `$limited`])

パラメータ

limited

返り値

共有メモリ割り当てデータの配列を返します。失敗した場合は **FALSE** を返します。

例

Example#1 apc_sma_info() の例

```
<?php
print_r(apc_sma_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [num_seg] => 1
     [seg_size] => 31457280
     [avail_mem] => 31448408
     [block_lists] => Array
         (
               [0] => Array
                   (
                         [0] => Array
                             (
                                   [size] => 31448408
                                   [offset] => 8864
                             )

                   )

         )

)
```

参考

- APC 設定ディレクティブ

---

apc_store

(PECL apc:3.0.0-3.0.9)

apc_store — 変数をデータ領域にキャッシュする

説明

bool **apc_store** (string $key , mixed $var [, int $ttl])

> **注意**: PHP の他の多くの仕組みと異なり、**apc_store()** を用いて格納された変数はリクエストを超えて （その値がキャッシュから取り除かれるまで）持続します。

パラメータ

key

> この名前を用いて変数を格納します。*key* は キャッシュ内で一意です。そのため、同一の *key* で新しい値を格納すると、元の値は上書きされます。

var

> 格納する変数。

ttl

> 有効期間。*var* は、キャッシュに *ttl* 秒間だけ格納されます。 *ttl* が経過すると、格納されている変数は （次のリクエスト時に）キャッシュから削除されます。 *ttl* が指定されていない（あるいは *ttl* が *0* の場合）は、 キャッシュから手動で削除される・あるいはキャッシュに存在できなくなる （clear, restart など）まで値が持続します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 A apc_store() の例

```
<?php
$bar = 'BAR';
apc_store('foo', $bar);
var_dump(apc_fetch('foo'));
?>
```

上の例の出力は以下となります。

```
string(3) "BAR"
```

参考

- apc_add()
- apc_fetch()
- apc_delete()

目次

- apc_delete — 格納されている変数をキャッシュから取り除く
- apc_fetch — 格納されている変数をキャッシュから取得する
- apc_load_constants — 定数群をキャッシュから読み込む
- apc_sma_info — APC の共有メモリ割り当てに関する情報を取得する
- apc_store — 変数をデータ領域にキャッシュする

Advanced PHP Debugger (APD)

導入

APD は進化した PHP デバッガです。PHP コードのプロファイリングや デバッグの機能を提供すること、また完全なスタックトレースを出力する 機能を提供することを目的として作成されています。APD は対話形式の デバッグもサポートしていますが、デフォルトではデータをトレース ファイルに書き出すようになっています。また、さまざまなレベルの 情報（関数のコール・渡された引数・時間などを含む）をイベント発生時に 記録することができ、それを個々のスクリプト単位で有効にしたり無効に したりできます。

警告

APD は Zend 拡張モジュールで、PHP 内部関数のコール方法に手を加えます。 そのため、他の Zend 拡張モジュール（たとえば Zend Optimizer など）との 相性に注意が必要です。

インストール手順

現在 APD は PECL 拡張モジュールとして公開されており、 » http://pecl.php.net/package/apd で入手可能です。CGI 版の PHP がインストールされており、パスの通った場所に phpize スクリプトがあることを確かめてください。

APD の最新安定バージョンをダウンロード・ビルド・インストールするには 以下のコマンドを実行します。

```
pear install apd
```

これは、APD Zend モジュールを自動的に PHP の extension ディレクトリに インストールします。必ずこの場所でなければならないわけではありません。 zend_extension パラメータを適切に設定すれば、PHP が 読み込めるディレクトリならどこにでもインストール可能です。

Windows ユーザは » http://snaps.php.net/win32/PECL_STABLE/ から *php_apd.dll* をダウンロードできます。

INI ファイルに以下の行を追加します。

```
zend_extension = /absolute/path/to/apd.so
apd.dumpdir = /absolute/path/to/trace/directory
apd.statement_tracing = 0
```

PHP のビルド状況によって、zend_extension ディレクティブは 以下のうちのいずれかひとつとなります。

```
zend_extension                (非 ZTS, 非 debug ビルド)
zend_extension_ts             (   ZTS, 非 debug ビルド)
zend_extension_debug          (非 ZTS,     debug ビルド)
zend_extension_debug_ts       (   ZTS,     debug ビルド)
```

Win32 でのビルド

Windows で APD をビルドするには、http://php.net/ で述べられているような PHP コンパイル環境が必要です。 -- 基本的には、Microsoft Visual C++・ win32build.zip・bison/flex・そしてそれらをうまく動かすためのちょっとした コツが必要になります。また adp.dsp の改行コードは必ず DOS 形式に してください。Unix 形式の改行コードだと、Microsoft Visual C++ に文句を言われます。

実行時設定

php.ini の設定により動作が変化します。

APD の設定オプション

名前	デフォルト	変更の可否	変更履歴
apd.dumpdir	NULL	PHP_INI_ALL	
apd.statement_tracing	"0"	PHP_INI_ALL	apd 0.9 以降で利用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

apd.dumpdir string

APD がプロファイルのダンプファイルを書き出すディレクトリを設定します。 絶対パス・相対パスのどちらも指定可能です。

apd_set_pprof_trace() へ引数を渡すことで、 指定した以外の場所に書き出すことも可能です。

apd.statement_tracing boolean

行単位のトレースをするかしないかを設定します。これを on (1) にすると、アプリケーションのパフォーマンスに衝撃的な影響を与えます。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

APD の定数

定数	値	説明
FUNCTION_TRACE (integer)	1	
ARGS_TRACE (integer)	2	
ASSIGNMENT_TRACE (integer)	4	
STATEMENT_TRACE (integer)	8	
MEMORY_TRACE (integer)	16	
TIMING_TRACE (integer)	32	
SUMMARY_TRACE (integer)	64	
ERROR_TRACE (integer)	128	
PROF_TRACE (integer)	256	
APD_VERSION (string)	例: *1.0.2-dev*	

スクリプト中で PHP-APD を使用する方法

1. トレースを開始するには、PHP スクリプトの最初の行で apd_set_pprof_trace() 関数をコールします。

   ```
   apd_set_pprof_trace();
   ```

   この行はスクリプト内のどの場所にでも挿入可能ですが、もしスクリプトの 最初からトレースを始めなければ、あなたをパフォーマンスのボトルネックに 導いてくれるかもしれないデータを捨てることになってしまいます。

2. さあ、スクリプトを実行しましょう。ダンプ結果は *apd.dumpdir/pprof_pid.ext* に出力されます。

   ヒント

   CGI 版の PHP を使用している場合は、apd が正しく動作するように '-e' フラグつきで実行し、拡張情報を有効にしてください。たとえば **php -e -f script.php** のように実行します。

3. プロファイルデータを整形して表示するには、お好みの並べ替えオプション・ 表示オプションを指定して **pprofp** コマンドを実行 してください。整形された出力は以下のようになります。

   ```
   bash-2.05b$  pprofp  -R  /tmp/pprof.22141.0

   Trace  for  /home/dan/testapd.php
   Total  Elapsed  Time  =  0.00
   Total  System  Time   =  0.00
   Total  User  Time     =  0.00


   Real          User          System              secs/     cumm
   %Time  (excl/cumm)   (excl/cumm)   (excl/cumm)  Calls      call      s/call   Memory  Usage  Name
   ---------------------------------------------------------------------------------------
   100.0 0.00 0.00   0.00 0.00   0.00 0.00       1   0.0000    0.0009               0 main
   56.9  0.00 0.00   0.00 0.00   0.00 0.00       1   0.0005    0.0005               0 apd_set_pprof_trace
   28.0  0.00 0.00   0.00 0.00   0.00 0.00      10   0.0000    0.0000               0 preg_replace
   14.3  0.00 0.00   0.00 0.00   0.00 0.00      10   0.0000    0.0000               0 str_replace
   ```

   この例で使われている -R オプションは、その関数を実行するのにかかった時間の 順でプロファイルテーブルを並べ替えます。"cumm call" 列に

は個々の関数が 何回コールされたか、そして "s/call" 列には 1 回のコールあたりの 平均所要時間が表示されます。

4. KCacheGrind にインポートできる形式のファイルを作成するには、 **pprof2calltree** コマンドを実行してください。

連絡先の情報

コメント・バグフィックス・機能拡張・あるいは開発を手伝いたいなどの場合は、 メールを » apd@mail.communityconnect.com に送ってください。大歓迎します。

---

apd_breakpoint

(PECL apd:0.2-1.0.1)

apd_breakpoint — インタプリタの処理を停止し、ソケットからの CR を待つ

説明

bool **apd_breakpoint** (int `$debug_level`)

スクリプトの実行を停止し、接続しているソケットからの応答を待ち受ける ために使用します。プログラムのステップ実行を行うには、Enter キーを押す (空行を送る) あるいは実行したい PHP コマンドを入力します。

パラメータ

`debug_level`

XXX_TRACE 定数の組み合わせによる整数値。

MEMORY_TRACE を用いることは推奨しません。これは非常に低速で、またあまり正確ではないようだからです。 **ASSIGNMENT_TRACE** は、まだ実装されていません。

すべての機能 (TIMING, FUNCTIONS, ARGS SUMMARY (strace -c のようなもの)) のトレースを有効にするには、値 99 を指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 tcplisten を使用した典型的なセッション

```
bash#tcplisten localhost 7777

APD - Advanced PHP Debugger Trace File
-------------------------------------------------------------------------------
Process Pid (6118)
Trace Begun at Sun Mar 10 23:13:12 2002
-------------------------------------------------------------------------------
(   0.000000): apd_set_session_trace called at /home/alan/Projects/project2/test.
php:5
(   0.074824): apd_set_session_trace_socket() at /home/alan/Projects/project2/tes
t.php:5 returned.   Elapsed (0.074824)
(   0.074918): apd_breakpoint() /home/alan/Projects/project2/test.php:7
                ++ argv[0] $(??) = 9
apd_breakpoint() at /home/alan/Projects/project2/test.php:7 returned.   Elapsed (
-2089521468.1073275368)
>¥n
statement: /home/alan/Projects/project2/test.php:8
>¥n
statement: /home/alan/Projects/project2/test.php:8
>¥n
statement: /home/alan/Projects/project2/test.php:10
>apd_echo($i);
EXEC: apd_echo($i);
0
>apd_echo(serialize(apd_get_active_symbols()));
EXEC:   apd_echo(serialize(apd_get_active_symbols()));
a:47:{i:0;s:4:"PWD";i:1;s:10:"COLORFGBG";i:2;s:11:"XAUTHORITY";i:3;s:14:"
COLORTERM_BCE";i:4;s:9:"WINDOWID";i:5;s:14:"ETERM_VERSION";i:6;s:16:"SE
SSION_MANAGER";i:7;s:4:"PS1";i:8;s:11:"GDMSESSION";i:9;s:5:"USER";i:10;s:5:"
MAIL";i:11;s:7:"OLDPWD";i:12;s:5:"LANG";i:13;s:10:"COLORTERM";i:14;s:8:"DISP
LAY";i:15;s:8:"LOGNAME";i:16;s:6:"
>apd_echo(system('ls  /home/mydir'));
........
>apd_continue(0);
```

---

apd_callstack

(PECL apd:0.2-0.4)

apd_callstack — 現在のコールスタックを配列で返す

説明

array **apd_callstack** (void)

現在のコールスタックを配列形式で返します。

返り値

現在のコールスタックを含む配列を返します。

例

Example#1 apd_callstack() の例

```
<?php
print_r(apd_callstack());
?>
```

---

apd_clunk

(No version information available, might be only in CVS)

apd_clunk — 警告とコールスタックをスローする

説明

void **apd_clunk** (string `$warning` [, string `$delimiter`])

perl の Carp::cluck と同じように動作します。警告とコールバックをスローします。 デフォルトの行区切り文字は "
\n" です。

パラメータ

`warning`

スローする警告。

`delimiter`

区切り文字。デフォルトは *
*。

返り値

値を返しません。

例

Example#1 apd_clunk() の例

```
<?php
apd_clunk("Some  Warning", "<br/>");
?>
```

参考

- apd_croak()

---

apd_continue

(PECL apd:0.2-1.0.1)

apd_continue — インタプリタを再開する

説明

bool **apd_continue** (int `$debug_level`)

インタプリタを再開するために、一般的にはソケット経由で送信します。

パラメータ

debug_level

XXX_TRACE 定数の組み合わせによる整数値。

MEMORY_TRACE を用いることは推奨しません。これは非常に低速で、またあまり正確ではないようだからです。 **ASSIGNMENT_TRACE** は、まだ実装されていません。

すべての機能 (TIMING, FUNCTIONS, ARGS SUMMARY (strace -c のようなもの)) のトレースを有効にするには、値 99 を指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apd_continue() の例

```
<?php
apd_continue(0);
?>
```

---

apd_croak

(PECL apd:0.2-0.4)

apd_croak — エラーとコールスタックをスローし、終了する

説明

void **apd_croak** (string `$warning` [, string `$delimiter`])

perl の Carp::croak と同じように動作します。エラーとコールバックをスローし、 終了します。

パラメータ

warning

スローする警告。

delimiter

区切り文字。デフォルトは *
*。

返り値

値を返しません。

例

Example#1 apd_croak() の例

```
<?php
apd_croak("Some  Warning","<P>");
?>
```

参考

- apd_clunk()

---

apd_dump_function_table

(No version information available, might be only in CVS)

apd_dump_function_table — 現在の関数テーブルを出力する

説明

void **apd_dump_function_table** (void)

現在の関数テーブルを出力します。

返り値

値を返しません。

例

Example#1 apd_dump_function_table() の例

```
<?php
apd_dump_function_table();
?>
```

---

apd_dump_persistent_resources

(PECL apd:0.2-0.4)

apd_dump_persistent_resources — すべての持続的なリソースを配列で返す

説明

array **apd_dump_persistent_resources** (void)

すべての持続的なリソースを配列で返します。

返り値

現在のコールスタックを含む配列を返します。

例

Example#1 apd_dump_persistent_resources() の例

```
<?php
print_r(apd_dump_persistent_resources());
?>
```

参考

- **apd_dump_regular()**

---

apd_dump_regular_resources

(PECL apd:0.2-0.4)

apd_dump_regular_resources — 現在のすべての一般リソースを配列で返す

説明

array **apd_dump_regular_resources** (void)

現在のすべての一般リソースを配列で返します。

返り値

現在の一般リソースを含む配列を返します。

例

Example#1 apd_dump_regular_resources() の例

```
<?php
print_r(apd_dump_regular_resources());
?>
```

参考

- apd_dump_persistent_resources()

---

apd_echo

(PECL apd:0.2-1.0.1)

apd_echo — デバッグ用ソケットに表示する

説明

bool **apd_echo** (string `$output`)

実行中のスクリプトに関する情報を、 一般的にはソケット経由でリクエストします。

パラメータ

`output`

デバッグされた変数。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apd_echo() の例

```
<?php
apd_echo($i);
?>
```

---

apd_get_active_symbols

(PECL apd:0.2)

apd_get_active_symbols — ローカルスコープ内の現在の変数名を配列で取得する

説明

array **apd_get_active_symbols** (void)

アクティブなスコープ内で定義されているすべての変数名を返します (変数の値ではありません)。

返り値

すべての変数を含む多次元配列を返します。

例

Example#1 apd_get_active_symbols() の例

```
<?php
apd_echo(apd_get_active_symbols());
?>
```

---

apd_set_pprof_trace

(PECL apd:0.2-1.0.1)

apd_set_pprof_trace — セッションのデバッグを開始する

説明

string **apd_set_pprof_trace** ([string `$dump_directory` [, string `$fragment`]])

ダンプディレクトリの *pprof_{process_id}* へのデバッグを開始します。

パラメータ

dump_directory

プロファイルのダンプファイルを書き込むディレクトリ。 指定しない場合は、*php.ini* の設定 apd.dumpdir を使用します。

fragment

返り値

対象となるファイルへのパスを返します。

例

Example#1 apd_set_pprof_trace() の例

```
<?php
apd_set_pprof_trace();
?>
```

参考

- apd_set_session_trace()

---

apd_set_session_trace

(PECL apd:0.2-0.4)

apd_set_session_trace — セッションのデバッグを開始する

説明

void **apd_set_session_trace** (int `$debug_level` [, string `$dump_directory`])

ダンプディレクトリの *apd_dump_{process_id}* へのデバッグを開始します。

パラメータ

debug_level

XXX_TRACE 定数の組み合わせによる整数値。

MEMORY_TRACE を用いることは推奨しません。これは非常に低速で、またあまり正確ではないようだからです。 **ASSIGNMENT_TRACE** は、まだ実装されていません。

すべての機能 (TIMING, FUNCTIONS, ARGS SUMMARY (strace -c のようなもの)) のトレースを有効にするには、値 99 を指定します。

dump_directory

プロファイルのダンプファイルを書き込むディレクトリ。 指定しない場合は、*php.ini* の設定 apd.dumpdir を使用します。

返り値

値を返しません。

例

Example#1 apd_set_session_trace() の例

```
<?php
apd_set_session_trace(99);
?>
```

---

apd_set_session

(PECL apd:0.2-0.4)

apd_set_session — 現在のデバッグレベルを変更あるいは設定する

説明

void **apd_set_session** (int $debug_level)

アプリケーション内で、場所によってデバッグのレベルを増減する際に使用可能です。

パラメータ

debug_level

XXX_TRACE 定数の組み合わせによる整数値。

MEMORY_TRACE を用いることは推奨しません。これは非常に低速で、またあまり正確ではないようだからです。 **ASSIGNMENT_TRACE** は、まだ実装されていません。

すべての機能 (TIMING, FUNCTIONS, ARGS SUMMARY (strace -c のようなもの)) のトレースを有効にするには、値 99 を指定します。

返り値

値を返しません。

例

Example#1 apd_set_session() の例

```
<?php
apd_set_session(9);
?>
```

---

apd_set_socket_session_trace

(No version information available, might be only in CVS)

apd_set_socket_session_trace — リモートセッションのデバッグを開始する

説明

bool **apd_set_socket_session_trace** (string $tcp_server , int $socket_type , int $port , int $debug_level)

指定した *tcp_server* (例: tcplisten) に接続詞、デバッグデータをソケットに送信します。

パラメータ

tcp_server

TCP サーバの IP あるいは Unix ドメインソケット (ファイルなど)。

socket_type

ファイルベースのソケットの場合は **AF_UNIX**、 標準の tcp/ip の場合は **APD_AF_INET**。

port

任意のポートが使用できます。しかし大きめの番号にしておくことを推奨します。 小さめの番号はその他のシステムサービスが使用している可能性があります。

debug_level

XXX_TRACE 定数の組み合わせによる整数値。

MEMORY_TRACE を用いることは推奨しません。これは非常に低速で、またあまり正確ではないようだからです。 **ASSIGNMENT_TRACE** は、まだ実装されていません。

すべての機能 (TIMING, FUNCTIONS, ARGS SUMMARY (strace -c のようなもの)) のトレースを有効にするには、値 99 を指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 apd_set_socket_session_trace() の例

```
<?php
    apd_set_socket_session_trace("127.0.0.1",APD_AF_INET,7112,0);
?>
```

---

override_function

(PECL apd:0.2-1.0.1)

override_function — 組み込みの関数を上書きする

説明

bool **override_function** (string `$function_name` , string `$function_args` , string `$function_code`)

シンボルテーブルを書き換えることで、組み込みの関数を上書きします

パラメータ

`function_name`

上書きする関数。

`function_args`

関数への引数をカンマ区切りの文字列で指定します。

通常は、このパラメータだけでなく *`function_code`* パラメータも (シングルクォート区切りの文字列で) 指定することでしょう。シングルクォートで囲んだ文字列を使用する理由は、 変数名がパースされないようにするためです。 ダブルクォートを使用するなら、変数名をエスケープして ¥$*your_var* のようにしなければなりません。

`function_code`

関数の新しいコード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 override_function() の例

```
<?php
override_function('test', '$a,$b', 'echo "DOING TEST"; return $a * $b;');
?>
```

---

rename_function

(PECL apd:0.2-1.0.1)

rename_function — グローバルの関数テーブルで関数名を変更する

説明

bool **rename_function** (string `$original_name` , string `$new_name`)

グローバルの関数テーブルで関数名を変更します。 一時的に組み込み関数を上書きする際に有用です。

パラメータ

`original_name`

もとの関数名。

new_name

関数 *original_name* の新しい名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 rename_function() の例

```
<?php
rename_function('mysql_connect', 'debug_mysql_connect'  );
?>
```

目次

配列関数(array)

導入

これらの関数により様々な手法で配列にアクセスし、操作することが可能 になります。配列は、変数の組を保存、管理、操作する基本的な要素です。

通常の配列および多次元配列がサポートされており、ユーザが定義したり、 他の関数で作成することも可能です。いくつかのデータベース処理関数は、データベースのクエリから配列を返しますし、いくつかの関数は配列を返 します。

PHPでの配列の実装や使用方法の詳細については、マニュアルの 配列に関する節を参照下 さい。 その他の配列の操作方法については、 配列演算子も参照ください。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数は、PHP コアに含まれており、常に利用可能です。

CASE_LOWER (integer)
　　CASE_LOWERは、 array_change_key_case()で使用され、 配列のキーを小文字に変換するために使用されます。小文字は、array_change_key_case()のデフォルトのケースで もあります。
CASE_UPPER (integer)
　　CASE_UPPERは、 array_change_key_case()で使用され、配列のキー を大文字に変換するために使用されます。

ソース順のフラグ:

SORT_ASC (integer)
　　SORT_ASCは、 array_multisort()でソート順を昇順にするために 使用されます。
SORT_DESC (integer) (integer)
　　SORT_DESCは、 array_multisort()でソート順を降順にするために 使用されます。

ソート型のフラグ: 種々のソート関数で使用されます

SORT_REGULAR (integer)
　　SORT_REGULARは通常の比較するために使用され ます。
SORT_NUMERIC (integer)
　　SORT_NUMERICは数値で比較を行うために使用さ れます。
SORT_STRING (integer)
　　SORT_STRINGは文字列として比較を行うために使 用されます。
SORT_LOCALE_STRING (integer)
　　SORT_LOCALE_STRINGは現在のロケールに基づいた 文字列として比較を行うために使用されます。 PHP 4.4.0と5.0.2で追加 されました。

COUNT_NORMAL (integer)
COUNT_RECURSIVE (integer)
EXTR_OVERWRITE (integer)
EXTR_SKIP (integer)
EXTR_PREFIX_SAME (integer)
EXTR_PREFIX_ALL (integer)
EXTR_PREFIX_INVALID (integer)
EXTR_PREFIX_IF_EXISTS (integer)
EXTR_IF_EXISTS (integer)
EXTR_REFS (integer)

参考

is_array(), explode(), implode(), split(), preg_split(), および join() も参照してください。

---

array_change_key_case

(PHP 4 >= 4.2.0, PHP 5)

array_change_key_case — 配列のすべてのキーを変更する

説明

array **array_change_key_case** (array `$input` [, int `$case`])

input のすべてのキーを小文字あるいは大文字にした配列を返します。 数値添字はそのままとなります。

パラメータ

input

　　処理を行う配列。

case

　　CASE_UPPER あるいは **CASE_LOWER** (デフォルト)。

返り値

すべてのキーを小文字あるいは大文字にした配列を返します。 *input* が配列でない場合は false を返します。

エラー / 例外

input が配列でない場合は **E_WARNING** が発生します。

例

Example#1 array_change_key_case() の例

```
<?php
$input_array = array("FirSt" => 1, "SecOnd" => 4);
print_r(array_change_key_case($input_array, CASE_UPPER));
?>
```

上の例の出力は以下となります。

```
Array
(
    [FIRST] => 1
    [SECOND] => 4
)
```

注意

> **注意**: 配列の添字に、この関数を使うことによって同じになってしまうものがある場合 (例:"keY" と "kEY")、配列の後のほうにある値が他の値を上書きします。

---

array_chunk

(PHP 4 >= 4.2.0, PHP 5)

array_chunk — 配列を分割する

説明

array **array_chunk** (array $input , int $size [, bool $preserve_keys])

配列を、*size* 個ずつの要素に分割します。 最後の部分の要素数は *size* より小さくなることもあります。

パラメータ

input

> 処理を行う配列。

size

> 各部分のサイズ。

preserve_keys

> **TRUE** の場合はキーをそのまま保持します。 デフォルトは **FALSE** で、各部分のキーをあらためて数字で振りなおします。

返り値

数値添字の多次元配列を返します。添え字はゼロから始まり、 各次元の要素数が *size* となります。

エラー / 例外

size が 1 より小さい場合は **E_WARNING** が発生し、**NULL** を返します。

例

Example#1 array_chunk() の例

```
<?php
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, true));
?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => Array
         (
              [0] => a
              [1] => b
         )

     [1] => Array
         (
              [0] => c
              [1] => d
         )

     [2] => Array
         (
              [0] => e
         )

)
Array
(
     [0] => Array
         (
              [0] => a
              [1] => b
         )

     [1] => Array
         (
              [2] => c
              [3] => d
         )

     [2] => Array
         (
              [4] => e
         )

)
```

---

array_combine

(PHP 5)

array_combine — 一方の配列をキーとして、もう一方の配列を値として、ひとつの配列を生成する

説明

array **array_combine** (array $keys , array $values)

keys 配列の値をキーとして、また *values* 配列の値を対応する値として生成した 配列 を作成します。

パラメータ

keys

キーとして使用する配列。無効な値を渡すと文字列に変換されます。

values

値として使用する配列。

返り値

作成した配列を返します。 互いの配列の要素の数が合致しない場合や空の配列である場合に **FALSE** を返します。

エラー / 例外

keys および *values* のいずれかが空であったり要素数が一致しなかったりした場合は **E_WARNING** が発生します。

例

Example#1 array_combine()の簡単な例

```
<?php
$a = array('green', 'red', 'yellow');
$b = array('avocado', 'apple', 'banana');
```

```php
$c = array_combine($a, $b);

print_r($c);
?>
```

上の例の出力は以下となります。

```
Array
(
     [green]   => avocado
     [red]     => apple
     [yellow] => banana
)
```

参考

- array_merge()
- array_walk()
- array_values()

---

array_count_values

(PHP 4, PHP 5)

array_count_values — 配列の値の数を数える

説明

array **array_count_values** (array $input)

array_count_values() は、 配列 *input* の値をキーとし、 *input* におけるその値の出現回数を値とした配列を返します。

パラメータ

input

値を数える配列。

返り値

input のキーとその登場回数を組み合わせた連想配列を返します。

エラー / 例外

string あるいは integer 以外の型の要素が登場するたびに **E_WARNING** が発生します。

例

Example#1 array_count_values() の例

```php
<?php
$array = array(1, "hello", 1, "world", "hello");
print_r(array_count_values($array));
?>
```

上の例の出力は以下となります。

```
Array
(
     [1] => 2
     [hello] => 2
     [world] => 1
)
```

参考

- count()
- array_unique()
- array_values()
- count_chars()

---

array_diff_assoc

(PHP 4 >= 4.3.0, PHP 5)

array_diff_assoc — 追加された添字の確認を含めて配列の差を計算する

説明

array **array_diff_assoc** (array $array1 , array $array2 [, array $...])

array1 を *array2* と比較し、 その差を返します。array_diff() とは異なり、 配列のキーを用いて比較を行います。

パラメータ

array1

比較元の配列。

array2

比較する対象となる配列。

...

さらに比較する対象となる配列。

返り値

array1 の要素のうち、 その他の配列のいずれにも含まれないものだけを残した配列を返します。

例

Example#1 array_diff_assoc() の例

この例で、*"a" => "green"* の組が両方の配列に現れており、 このため、この関数の出力には含まれていません。 これとは異なり、*0 => "red"* は出力の中にありますが、 これは、二番目の引数において *"red"* が *1* というキーを有しているためです。

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "green", "yellow", "red");
$result = array_diff_assoc($array1, $array2);
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
    [b] => brown
    [c] => blue
    [0] => red
)
```

Example#2 array_diff_assoc() の例

key => value の組からの二つの値は、 *(string) $elem1 === (string) $elem2* が成り立つ場合のみ等しいと見なされます。 言い替えると、厳密なチェックが行われるため、 文字列表現が同じである必要があります。

```
<?php
$array1 = array(0, 1, 2);
$array2 = array("00", "01", "2");
$result = array_diff_assoc($array1, $array2);
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => 0
    [1] => 1
    )
```

注意

注意: この関数は、N 次元配列の一次元だけを調べます。もちろん、例えば *array_diff_assoc($array1[0], $array2[0]);* とすることにより、より深い次元でチェックを行うことも可能です。

参考

- array_diff()
- array_intersect()
- array_intersect_assoc()

---

array_diff_key

(PHP 5 >= 5.1.0)

array_diff_key — キーを基準にして配列の差を計算する

説明

array **array_diff_key** (array $array1 , array $array2 [, array $...])

array1 のキーを *array2* のキーと比較し、その差を返します。 この関数は array_diff() に似ていますが、 値ではなくキーを用いて比較するという点が異なります。

パラメータ

array1

比較元の配列。

array2

比較する対象となる配列。

...

さらに比較する対象となる配列。

返り値

array1 の要素のうち、 その他の配列のいずれにも含まれないものだけを残した配列を返します。

例

Example#1 array_diff_key() の例

ふたつの *key => value* のペアが等しいとみなされるのは、 *(string) $key1 === (string) $key2* である場合のみです。つまり、厳密な型チェックを行うということです。 文字列表現が一致しなければなりません。

```
<?php
$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan'   => 8);

var_dump(array_diff_key($array1, $array2));
?>
```

上の例の出力は以下となります。

```
array(2) {
  ["red"]=>
  int(2)
  ["purple"]=>
  int(4)
}
```

注意

注意: この関数は n 次元配列の一つの次元しかチェックしません。 もちろん、*array_diff_key($array1[0], $array2[0]);* のようにすることでより深い次元でのチェックもできます。

参考

- array_diff()

---

array_diff_uassoc

(PHP 5)

array_diff_uassoc — ユーザが指定したコールバック関数を利用し、 追加された添字の確認を含めて配列の差を計算する

説明

array **array_diff_uassoc** (array `$array1` , array `$array2` [, array `$...`], callback `$key_compare_func`)

array1 のキーを *array2* のキーと比較し、その差を返します。 この関数は array_diff() に似ていますが、 配列のキーを用いて比較するという点が異なります。

array_diff_assoc() とは異なり、 内部関数ではなくユーザが指定したコールバック関数を用いて添字を比較します。

パラメータ

array1

比較元の配列。

array2

比較する対象となる配列。

...

さらに比較する対象となる配列。

key_compare_func

使用するコールバック関数。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ 負の数 / ゼロ / 正の数 を返す必要があります。

返り値

array1 の要素のうち、 その他の配列のいずれにも含まれないものだけを残した配列を返します。

例

Example#1 array_diff_uassoc() の例

"a" => "green" の組み合わせが両方の配列に存在し、 関数の出力には存在しないことが確認できます。 これとは異なり、*0 => "red"* は出力されています。 なぜなら 2 つめの引数の *"red"* は、キーが *1* だからです。

```
<?php
function key_compare_func($a, $b)
{
    if ($a === $b) {
        return 0;
    }
    return ($a > $b)? 1:-1;
}

$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "green", "yellow", "red");
$result = array_diff_uassoc($array1, $array2, "key_compare_func");
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [b] => brown
     [c] => blue
     [0] => red
)
```

2 つのインデックスが等しいかどうかは、 ユーザが指定したコールバック関数で調べます。

注意

> **注意**: この関数は n 次元配列の一つの次元しかチェックしません。 もちろん、*array_diff_uassoc($array1[0], $array2[0], "key_compare_func");* のようにすることでより深い次元でのチェックもできます。

参考

- array_diff()
- array_diff_assoc()
- array_udiff()
- array_udiff_assoc()
- array_udiff_uassoc()
- array_intersect()
- array_intersect_assoc()
- array_uintersect()
- array_uintersect_assoc()
- array_uintersect_uassoc()

---

array_diff_ukey

(PHP 5 >= 5.1.0)

array_diff_ukey — キーを基準にし、コールバック関数を用いて配列の差を計算する

説明

array **array_diff_ukey** (array $array1 , array $array2 [, array $ ...], callback $key_compare_func)

array1 のキーを *array2* のキーと比較し、その差を返します。 この関数は array_diff() に似ていますが、 値ではなくキーを用いて比較するという点が異なります。

array_diff_key() とは異なり、 内部関数ではなくユーザが指定したコールバック関数を用いて添字を比較します。

パラメータ

array1

> 比較元の配列。

array2

> 比較する対象となる配列。

...

> さらに比較する対象となる配列。

key_compare_func

> 使用するコールバック関数。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ 負の数 / ゼロ / 正の数 を返す必要があります。

返り値

array1 の要素のうち、 その他の配列のいずれにも含まれないものだけを残した配列を返します。

例

Example#1 array_diff_ukey() の例

```php
<?php
function key_compare_func($key1, $key2)
{
    if ($key1 == $key2)
        return 0;
    else if ($key1 > $key2)
        return 1;
    else
        return -1;
}

$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan'   => 8);

var_dump(array_diff_ukey($array1, $array2, 'key_compare_func'));
?>
```

上の例の出力は以下となります。

```
array(2) {
  ["red"]=>
  int(2)
  ["purple"]=>
  int(4)
}
```

注意

> **注意**: この関数は n 次元配列の一つの次元しかチェックしません。 もちろん、*array_diff_ukey($array1[0], $array2[0], 'callback_func');* のようにすることでより深い次元でのチェックもできます。

参考

- array_diff()
- array_udiff()
- array_diff_assoc()
- array_diff_uassoc()
- array_udiff_assoc()
- array_udiff_uassoc()
- array_diff_key()
- array_intersect()
- array_intersect_assoc()
- array_intersect_uassoc()
- array_intersect_key()
- array_intersect_ukey()

---

array_diff

(PHP 4 >= 4.0.1, PHP 5)

array_diff — 配列の差を計算する

説明

array **array_diff** (array $array1 , array $array2 [, array $ ...])

array1 を *array2* と比較し、その差を返します。

例

Example#1 array_diff() の例

```php
<?php
$array1 = array("a" => "green", "red", "blue", "red");
$array2 = array("b" => "green", "yellow", "red");
$result = array_diff($array1, $array2);

print_r($result);
?>
```

$array1 に複数存在する場合でも全て同様に処理されます。 この出力は次の通りです。

```
Array
(
    [1] => blue
)
```

注意

注意: 二つの要素は、*(string) $elem1 === (string) $elem2* の場合のみ等しいと見直されます。言い換えると、文字列表現が同じ場合となります。

注意: この関数は n 次元配列の一つの次元しかチェックしません。 もちろん、*array_diff($array1[0], $array2[0]);* のようにすることでより深い次元でのチェックもできます。

警告

この関数は、PHP 4.0.4 では動作しません!

参考

- array_diff_assoc()
- array_intersect()
- array_intersect_assoc()

---

array_fill_keys

(PHP 5 >= 5.2.0)

array_fill_keys — キーを指定して、配列を値で埋める

説明

array **array_fill_keys** (array $keys , mixed $value)

パラメータ *value* で指定した値で配列を埋めます。 キーとして、配列 *keys* で指定した値を使用します。

パラメータ

keys

キーとして使用する値の配列。

value

文字列か、あるいは値の配列。

返り値

値を埋めた配列を返します。

例

Example#1 array_fill_keys() の例

```php
<?php
$keys = array('foo', 5, 10, 'bar');
$a = array_fill_keys($keys, 'banana');
print_r($a);
?>
```

上の例の出力は以下となります。

```
Array
(
    [foo] => banana
    [5] => banana
    [10] => banana
    [bar] => banana
)
```

参考

- array_fill()
- array_combine()

---

array_fill

(PHP 4 >= 4.2.0, PHP 5)

array_fill — 配列を指定した値で埋める

説明

array **array_fill** (int $start_index , int $num , mixed $value)

パラメータ *value* を値とする *num* 個のエントリからなる配列を埋めます。 この際、キーは、*start_index* パラメータから開始します。

パラメータ

start_index

返される配列の最初のインデックス。

num

挿入する要素数。

value

要素に使用する値。

返り値

値を埋めた配列を返します。

エラー / 例外

num が 1 より小さい場合に **E_WARNING** が発生します。

例

Example#1 array_fill() の例

```
<?php
$a = array_fill(5, 6, 'banana');
print_r($a);
?>
```

上の例の出力は以下となります。

```
Array
(
     [5]  => banana
     [6]  => banana
     [7]  => banana
     [8]  => banana
     [9]  => banana
     [10] => banana
)
```

参考

- str_repeat()
- range()

---

array_filter

(PHP 4 >= 4.0.6, PHP 5)

array_filter — コールバック関数を使用して、配列の要素をフィルタリングする

説明

array **array_filter** (array $input [, callback $callback])

callback 関数によりフィルタ処理が行われた *input* の全ての要素を含む配列を返します。 *callback* 関数が true を返した場合、 *input* の現在の値が結果の配列に入ります。 *input* が連想配列の場合、 キーは保存されます。

パラメータ

input

処理する配列。

callback

使用するコールバック関数。

コールバック関数が与えられなかった場合、 *input* のエントリの中で **FALSE** と等しいもの (boolean への変換 を参照ください) がすべて削除されます。

返り値

フィルタリングされた結果の配列を返します。

例

Example#1 array_filter() の例

```
<?php
function odd($var)
{
     return($var & 1);
}

function even($var)
{
     return(!($var & 1));
}

$array1 = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array(6, 7, 8, 9, 10, 11, 12);

echo "Odd :¥n";
print_r(array_filter($array1, "odd"));
echo "Even:¥n";
print_r(array_filter($array2, "even"));
?>
```

上の例の出力は以下となります。

```
Odd :
Array
(
     [a] => 1
     [c] => 3
     [e] => 5
)
Even:
Array
(
     [0] => 6
     [2] => 8
     [4] => 10
     [6] => 12
)
```

Example#2 *callback* がない array_filter()

```
<?php

$entry = array(
              0 => 'foo',
              1 => false,
              2 => -1,
              3 => null,
              4 => ''
          );

print_r(array_filter($entry));
?>
```

上の例の出力は以下となります。

```
Array
(
```

```
    [0] => foo
    [2] => -1
)
```

注意

警告

配列の内容がコールバック関数内で変更された場合 (たとえば要素が追加されたり削除されたりなど) のこの関数の挙動は未定義です。

参考

- array_map()
- array_reduce()
- array_walk()

---

array_flip

(PHP 4, PHP 5)

array_flip — 配列のキーと値を反転する

説明

array **array_flip** (array $trans)

array_flip() は、配列を反転して返します。 すなわち、*trans* のキーが値となり、 *trans* の値がキーとなります。

trans の値は有効なキーを必要とすることに注意してください。 すなわち、キーは、integer または string である必要があります。ある値が間違った型である場合、 警告が出力され、問題のこのキー/値の組は逆順にされません。

ある値が複数回出現した場合、最後のキーがその値として使用され、 その他の値は失われます。

パラメータ

trans

反転を行うキー/値の組。

返り値

成功した場合に反転した配列、失敗した場合に **FALSE** を返します。

例

Example#1 array_flip() の例

```php
<?php
$trans  = array_flip($trans);
$original  = strtr($str, $trans);
?>
```

Example#2 array_flip() の例 : 衝突

```php
<?php
$trans  = array("a" => 1, "b" => 1, "c" => 2);
$trans  = array_flip($trans);
print_r($trans);
?>
```

$trans は次のようになります:

```
Array
(
    [1] => b
    [2] => c
)
```

参考

- array_values()

- array_keys()
- array_reverse()

---

array_intersect_assoc

(PHP 4 >= 4.3.0, PHP 5)

array_intersect_assoc — 追加された添字の確認も含めて配列の共通項を確認する

説明

array **array_intersect_assoc** (array $array1 , array $array2 [, array $ ...])

array_intersect_assoc() は、全ての引数に現れる *array1* の全ての値を含む配列を返します。 array_intersect() と異なり、 キーが比較に使用されることに注意してください。

パラメータ

array1

値を調べるもととなる配列。

array2

値を比較する対象となる配列。

array

さらにそれ以外の配列。

返り値

array1 の値のうち、すべての引数に存在するものを含む連想配列を返します。

例

Example#1 array_intersect_assoc() の例

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "green", "yellow", "red");
$result_array = array_intersect_assoc($array1, $array2);
print_r($result_array);
?>
```

上の例の出力は以下となります。

```
Array
(
    [a] => green
)
```

上の例で、*"a" => "green"* の組のみが両方の配列に現れており、 よって配列として返されていることがわかります。 値 *"red"* は返されません。これは、*$array1* においてはそのキーが *0* であり、一方、*$array2* においてはキーが *1* であるためです。

key => value の組からの二つの値は、 *(string) $elem1 === (string) $elem2* が成り立つ場合にのみ等しいと見なされます。 言い替えると、厳密なチェックが行われるため、 文字列表現が同じである必要があります。

参考

- array_intersect()
- array_uintersect_assoc()
- array_intersect_uassoc()
- array_uintersect_uassoc()
- array_diff()
- array_diff_assoc()

---

array_intersect_key

(PHP 5 >= 5.1.0)

array_intersect_key — キーを基準にして配列の共通項を計算する

説明

array **array_intersect_key** (array $array1 , array $array2 [, array $ ...])

array_intersect_key() は、他の全ての引数に存在する *array1* の値を全て有する配列を返します。

パラメータ

array1

値を調べるもととなる配列。

array2

値を比較する対象となる配列。

array

さらにそれ以外の配列。

返り値

array1 の値のうち、 すべての引数に存在するキーのものを含む連想配列を返します。

例

Example#1 array_intersect_key() の例

```
<?php
$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan'   => 8);

var_dump(array_intersect_key($array1, $array2));
?>
```

上の例の出力は以下となります。

```
array(2) {
   ["blue"]=>
   int(1)
   ["green"]=>
   int(3)
}
```

この例では、両方の配列に存在するキーが *'blue'* と *'green'* だけであり、それが返されていることが わかります。また、ふたつの配列の間で *'blue'* と *'green'* に対応する値が違うことにも注意しましょう。 それでも一致していると判定されるのは、ただキーだけがチェックされているからです。 返される値は *array1* のものです。

二つの要素は、 *(string) $elem1 === (string) $elem2* の場合のみ等しいとみなされます。言い換えると、 文字列表現が同じ場合となります。

参考

- array_diff()
- array_udiff()
- array_diff_assoc()
- array_diff_uassoc()
- array_udiff_assoc()
- array_udiff_uassoc()
- array_diff_key()
- array_diff_ukey()
- array_intersect()
- array_intersect_assoc()
- array_intersect_uassoc()
- array_intersect_ukey()

---

array_intersect_uassoc

(PHP 5)

array_intersect_uassoc — 追加された添字の確認も含め、コールバック関数を用いて 配列の共通項を確認する

説明

array **array_intersect_uassoc** (array $array1 , array $array2 [, array $ ...], callback $key_compare_func)

array_intersect_uassoc() は、全ての引数に現れる *array1* の全ての値を含む配列を返します。 array_intersect() と異なり、 キーが比較に使用されることに注意してください。

比較は、ユーザが指定したコールバック関数を利用して行われます。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ負の数 / ゼロ / 正の数 を返す必要があります。

パラメータ

array1

比較元となる最初の配列。

array2

キーを比較する対象となる最初の配列。

array

キーを比較する対象となる配列の、可変リスト。

key_compare_func

比較に使用する、ユーザ定義のコールバック関数。

返り値

array1 の値のうち、 すべての引数に存在するもののみを返します。

例

Example#1 array_intersect_uassoc() の例

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

print_r(array_intersect_uassoc($array1, $array2, "strcasecmp"));
?>
```

上の例の出力は以下となります。

```
Array
(
    [b] => brown
)
```

参考

- array_intersect()
- array_intersect_assoc()
- array_uintersect_assoc()
- array_uintersect_uassoc()
- array_intersect_key()
- array_intersect_ukey()

---

array_intersect_ukey

(PHP 5 >= 5.1.0)

array_intersect_ukey — キーを基準にし、コールバック関数を用いて 配列の共通項を計算する

説明

array **array_intersect_ukey** (array $array1 , array $array2 [, array $...], callback $key_compare_func)

array_intersect_ukey() は、他の全ての引数に存在する *array1* の値を全て有する配列を返します。

比較は、ユーザが指定したコールバック関数を利用して行われます。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ負の数 / ゼロ / 正の数 を返す必要があります。

パラメータ

array1

比較元となる最初の配列。

array2

キーを比較する対象となる最初の配列。

array

キーを比較する対象となる配列の、可変リスト。

key_compare_func

比較に使用する、ユーザ定義のコールバック関数。

返り値

array1 の値のうち、 すべての引数に存在するキーのものを含む連想配列を返します。

例

Example#1 array_intersect_ukey() の例

```
<?php
function key_compare_func($key1, $key2)
{
    if ($key1 == $key2)
        return 0;
    else if ($key1 > $key2)
        return 1;
    else
        return -1;
}

$array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3, 'purple' => 4);
$array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7, 'cyan'   => 8);

var_dump(array_intersect_ukey($array1, $array2, 'key_compare_func'));
?>
```

上の例の出力は以下となります。

```
array(2) {
  ["blue"]=>
  int(1)
  ["green"]=>
  int(3)
}
```

この例では、両方の配列に存在するキーが *'blue'* と *'green'* だけであり、それが返されていることがわかります。 また、ふたつの配列の間で *'blue'* と *'green'* に対応する値が違うことにも注意しましょう。 それでも一致していると判定されるのは、ただキーだけがチェックされているからです。 返される値は *array1* のものです。

参考

- array_diff()
- array_udiff()
- array_diff_assoc()
- array_diff_uassoc()
- array_udiff_assoc()
- array_udiff_uassoc()
- array_diff_key()
- array_diff_ukey()
- array_intersect()
- array_intersect_assoc()
- array_intersect_uassoc()

- array_intersect_key()

---

array_intersect

(PHP 4 >= 4.0.1, PHP 5)

array_intersect — 配列の共通項を計算する

説明

array **array_intersect** (array $array1 , array $array2 [, array $ ...])

array_intersect() は、他の全ての引数に存在する *array1* の値を全て有する配列を返します。 キーと値の関係は維持されることに注意してください。

パラメータ

array1

値を調べるもととなる配列。

array2

値を比較する対象となる配列。

array

さらにそれ以外の配列。

返り値

array1 の値のうち、 すべての引数に存在する値のものを含む連想配列を返します。

例

Example#1 array_intersect() の例

```
<?php
$array1 = array("a" => "green", "red", "blue");
$array2 = array("b" => "green", "yellow", "red");
$result = array_intersect($array1, $array2);
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
    [a] => green
    [0] => red
)
```

注意

注意: 二つの要素は、 *(string) $elem1 === (string) $elem2* の場合のみ等しいとみなされます。言い換えると、文字列表現が同じ場合となります。

参考

- array_intersect_assoc()
- array_diff()
- array_diff_assoc()

---

array_key_exists

(PHP 4 >= 4.0.7, PHP 5)

array_key_exists — 指定したキーまたは添字が配列にあるかどうかを調べる

説明

bool **array_key_exists** (mixed $key , array $search)

指定した *key* が配列に設定されている場合、 **array_key_exists()** は **TRUE** を返します。 *key* は配列添字として使用できる全ての値を使用可能です。 **array_key_exists()** はオブジェクトに対しても動作します。

パラメータ

key

調べる値。

search

キーが存在するかどうかを調べたい配列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 array_key_exists() の例

```
<?php
$search_array = array('first' => 1, 'second' => 4);
if (array_key_exists('first', $search_array)) {
    echo "この配列には 'first' という要素が存在します";
}
?>
```

注意: この関数の名前は、PHP 4.0.6では **key_exists()** です。

Example#2 array_key_exists() 対 isset()

isset() は **NULL** 値を持つ配列キーに対して **TRUE** を返しません。一方、**array_key_exists()** は **TRUE** を返します。

```
<?php
$search_array = array('first' => null, 'second' => 4);

// false を返します
isset($search_array['first']);

// true を返します
array_key_exists('first', $search_array);
?>
```

参考

- isset()
- array_keys()
- in_array()

---

array_keys

(PHP 4, PHP 5)

array_keys — 配列のキーをすべて返す

説明

array **array_keys** (array $input [, mixed $search_value [, bool $strict]])

array_keys() は、配列 *input* から全てのキー (数値および文字列) を返します。

オプション *search_value* が指定された場合、 指定した値に関するキーのみが返されます。指定されない場合は、 *input* から全てのキーが返されます。 PHP 5 では、*strict* パラメータを使って、 比較に (===) のタイプを含めることができます。

パラメータ

input

返すキーを含む配列。

search_value

指定した場合は、これらの値を含むキーのみを返します。

strict

PHP 5 では、このパラメータを使用すると 検索時に厳密な比較 (===) を行います。

返り値

input のすべてのキーを配列で返します。

例

Example#1 array_keys() の例

```
<?php
$array = array(0 => 100, "color" => "red");
print_r(array_keys($array));

$array = array("blue", "red", "green", "blue", "blue");
print_r(array_keys($array, "blue"));

$array = array("color" => array("blue", "red", "green"),
               "size"  => array("small", "medium", "large"));
print_r(array_keys($array));
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => 0
    [1] => color
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
Array
(
    [0] => color
    [1] => size
)
```

参考

- array_values()
- array_key_exists()

---

array_map

(PHP 4 >= 4.0.6, PHP 5)

array_map — 指定した配列の要素にコールバック関数を適用する

説明

array **array_map** (callback `$callback` , array `$arr1` [, array `$...`])

array_map() は、*arr1* の各要素に *callback* 関数を適用した後、 その全ての要素を含む配列を返します。 *callback* 関数が受け付けるパラメータの数は、 **array_map()** に渡される配列の数に一致している必要があります。

パラメータ

callback

配列の各要素に適用するコールバック関数。

arr1

コールバック関数を適用する配列。

array

コールバック関数に渡す引数を指定する配列の可変リスト。

返り値

arr1 の各要素に *callback* 関数を適用した後、 その全ての要素を含む配列を返します。

例

Example#1 array_map() の例

```
<?php
function cube($n)
{
    return($n * $n * $n);
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cube", $a);
print_r($b);
?>
```

この例では、*$b* の内容は以下のようになります。

```
Array
(
    [0] => 1
    [1] => 8
    [2] => 27
    [3] => 64
    [4] => 125
)
```

例

Example#2 array_map() - もっと配列を使ってみる例

```
<?php
function show_Spanish($n, $m)
{
    return("The number $n is called $m in Spanish");
}

function map_Spanish($n, $m)
{
     return(array($n => $m));
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("show_Spanish", $a, $b);
print_r($c);

$d = array_map("map_Spanish", $a , $b);
print_r($d);
?>
```

上の例の出力は以下となります。

```
// printout of $c
Array
(
    [0] => The number 1 is called uno in Spanish
    [1] => The number 2 is called dos in Spanish
    [2] => The number 3 is called tres in Spanish
    [3] => The number 4 is called cuatro in Spanish
    [4] => The number 5 is called cinco in Spanish
)

// printout of $d
Array
(
    [0] => Array
        (
            [1] => uno
        )

    [1] => Array
        (
            [2] => dos
        )

    [2] => Array
        (
            [3] => tres
        )

    [3] => Array
        (
            [4] => cuatro
        )
```

```
    [4] => Array
        (
            [5] => cinco
        )

)
```

通常、二つ以上の配列を使用する場合、 それらの長さは等しい必要があります。 これは、 コールバック関数が対応する要素に対して並行して適用されるためです。 配列の長さが等しくない場合、最も短い配列は空の要素で拡張されます。

この関数の面白い使用方法として、 配列の配列を構築するというものがあります。 これは、コールバック関数の名前として **NULL** を使用することにより、簡単に実行できるものです。

Example#3 配列の配列を生成する

```php
<?php
$a = array(1, 2, 3, 4, 5);
$b = array("one", "two", "three", "four", "five");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => one
            [2] => uno
        )

    [1] => Array
        (
            [0] => 2
            [1] => two
            [2] => dos
        )

    [2] => Array
        (
            [0] => 3
            [1] => three
            [2] => tres
        )

    [3] => Array
        (
            [0] => 4
            [1] => four
            [2] => cuatro
        )

    [4] => Array
        (
            [0] => 5
            [1] => five
            [2] => cinco
        )

)
```

参考

- array_filter()
- array_reduce()
- array_walk()
- create_function()

callback 型に関する情報

---

array_merge_recursive

(PHP 4 >= 4.0.1, PHP 5)

array_merge_recursive — 二つ以上の配列を再帰的にマージする

説明

array **array_merge_recursive** (array $array1 [, array $...])

array_merge_recursive() は、 一つ以上の配列の要素をマージし、 前の配列の最後にもう一つの配列の値を追加します。 マージした後の配列を返します。

入力配列が同じ文字列のキーを有している場合、 これらのキーの値は配列に一つのマージされます。これは再帰的に行われます。 このため、値の一つが配列自体を指している場合、 この関数は別の配列の対応するエントリもマージします。 しかし、配列が同じ数値キーを有している場合、 後の値は元の値を上書せず、追加されます。

パラメータ

array1

マージするもとの配列。

array

再帰的にマージしていく配列の可変リスト。

返り値

すべての引数の配列をマージした結果の配列を返します。

例

Example#1 array_merge_recursive() の例

```
<?php
$ar1 = array("color" => array("favorite" => "red"), 5);
$ar2 = array(10, "color" => array("favorite" => "green", "blue"));
$result = array_merge_recursive($ar1, $ar2);
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [color] => Array
         (
               [favorite] => Array
                  (
                       [0] => red
                       [1] => green
                  )

               [0] => blue
         )

     [0] => 5
     [1] => 10
)
```

参考

- array_merge()

---

array_merge

(PHP 4, PHP 5)

array_merge — ひとつまたは複数の配列をマージする

説明

array **array_merge** (array $array1 [, array $array2 [, array $...]])

array_merge()は、前の配列の後ろに配列を追加することにより、 ひとつまたは複数の配列の要素をマージし、得られた配列を返します。

入力配列が同じキー文字列を有していた場合、そのキーに関する後に指定された値が、 前の値を上書きします。しかし、配列が同じ添字番号を有していても 値は追記されるため、このようなことは起きません。

配列が一つだけ指定され、その配列が数字で添字指定されていた場合、 キーの添字が連続となるように振り直されます。

Example#1 array_merge() の例

```
<?php
$array1 = array("color" => "red", 2, 4);
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid", 4);
$result = array_merge($array1, $array2);
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [color] => green
     [0] => 2
     [1] => 4
     [2] => a
     [3] => b
     [shape] => trapezoid
     [4] => 4
)
```

Example#2 簡単な array_merge() の例

```
<?php
$array1 = array();
$array2 = array(1 => "data");
$result = array_merge($array1, $array2);
?>
```

数値添字が振りなおされることに注意!

```
Array
(
     [0] => data
)
```

配列を完全に維持し、単に追加だけしたい場合 (つまり、既存のキーを上書きしたくはない場合) には、+ 演算子を使用してください:

```
<?php
$array1 = array();
$array2 = array(1 => "data");
$result = $array1 + $array2;
?>
```

数値添字は維持され、関連が保持されます。

```
Array
(
     [1] => data
)
```

警告

array_merge() の動作は PHP 5 で変更されました。PHP 4 とは異なり、 **array_merge()** は、array 型のパラメータのみを受け取るようになりました。しかし、 他の型をマージするために型キャストを使用することも可能です。 詳細は以下の例を参照してください。

Example#3 array_merge() PHP 5 の例

```
<?php
$beginning = 'foo';
$end = array(1 => 'bar');
$result = array_merge((array)$beginning, (array)$end);
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => foo
     [1] => bar
)
```

array_merge_recursive()、 array_combine() および 配列演算子 も参照ください。

array_multisort

(PHP 4, PHP 5)

array_multisort — 複数の多次元の配列をソートする

説明

bool **array_multisort** (array $ar1 [, mixed $arg [, mixed $... [, array $...]]])

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

array_multisort() は、複数の配列を一度に、 または、多次元の配列をその次元の一つでソートする際に使用可能です。 この関数は、ソートの際にキーの相関を維持します。

連想配列のキー (string) は不変ですが、 数値添字は再度振り直されます。

入力の配列は、行をソートする対象となるテーブルの行として扱われます。 これは、SQL ORDER BY 構文と似ています。最初の配列はソートされる最初の配列です。 その配列の行が同じだった場合は、次の入力配列でソートされるといったようになります。

この関数の引数の構造は、やや一般的ではありませんが、柔軟なものです。 先頭の引数は、配列である必要があります。その後の各引数には、 配列または次のリストにあるソート用フラグのどちらかを指定することが可能です。

ソート方法指定フラグ:

- **SORT_ASC** - 昇順にソート
- **SORT_DESC** - 降順にソート

ソート型のフラグ:

- **SORT_REGULAR** - 普通に比較
- **SORT_NUMERIC** - 数値的に比較
- **SORT_STRING** - 文字列として比較

各配列に同じ型のソート用フラグを二つ指定することは出来ません。 ある引数配列に適用を指定されたソート用フラグが適用されるのは、 その配列のみです。新しい配列引数を処理する前にデフォルトの **SORT_ASC** および **SORT_REGULAR** にリセットします。

Example#1 複数の配列をソートする

```
<?php
$ar1 = array(10, 100, 100, 0);
$ar2 = array(1, 3, 2, 4);
array_multisort($ar1, $ar2);

var_dump($ar1);
var_dump($ar2);
?>
```

この例では、ソートの後で、最初の配列は、0, 10, 100, 100 となります。 2番目の配列は、4, 1, 2, 3 を有します。最初の配列 (100 および 100) の同じエントリに対応している二番目の配列のエントリは、 同時にソートされます。

```
array(4) {
   [0]=> int(0)
   [1]=> int(10)
   [2]=> int(100)
   [3]=> int(100)
}
array(4) {
   [0]=> int(4)
   [1]=> int(1)
   [2]=> int(2)
   [3]=> int(3)
}
```

Example#2 多次元の配列をソートする

```
<?php
$ar = array(
        array("10", 11, 100, 100, "a"),
        array(   1,  2, "2",   3,   1)
      );
array_multisort($ar[0], SORT_ASC, SORT_STRING,
                $ar[1], SORT_NUMERIC, SORT_DESC);
var_dump($ar);
?>
```

この例では、ソートされた後、最初の配列は "10", 100, 100, 11, "a" (文字列として昇順でソートされています) に変換され、二番目の配列は、 1, 3, "2", 2, 1 (数値として降順にソートされています) となっています。

```
array(2)  {
   [0]=>  array(5)  {
      [0]=>  string(2)  "10"
      [1]=>  int(100)
      [2]=>  int(100)
      [3]=>  int(11)
      [4]=>  string(1)  "a"
  }
   [1]=>  array(5)  {
      [0]=>  int(1)
      [1]=>  int(3)
      [2]=>  string(1)  "2"
      [3]=>  int(2)
      [4]=>  int(1)
  }
}
```

Example#3 データベースの結果をソートする

この例では、配列 *data* の個々の要素がテーブルのひとつの行を表しています。 これは、データベースのレコードの典型的な形式です。

データの例:

```
volume  |  edition
--------+---------
     67 |        2
     86 |        1
     85 |        6
     98 |        2
     86 |        6
     67 |        7
```

データは *data* という名前の配列に格納します。 これは、例えば mysql_fetch_assoc() の結果をループさせたりすれば得られます。

```php
<?php
$data[] = array('volume'  => 67, 'edition'  => 2);
$data[] = array('volume'  => 86, 'edition'  => 1);
$data[] = array('volume'  => 85, 'edition'  => 6);
$data[] = array('volume'  => 98, 'edition'  => 2);
$data[] = array('volume'  => 86, 'edition'  => 6);
$data[] = array('volume'  => 67, 'edition'  => 7);
?>
```

この例では、データを *volume* の降順、 *edition* の昇順に並べ替えます。

私たちが今もっているのは行方向の配列ですが、 **array_multisort()** で必要なのは列方向の配列です。 そこで、以下のコードで列方向の配列を得たあとでソートを行います。

```php
<?php
// 列方向の配列を得る
foreach  ($data  as  $key  =>  $row)  {
    $volume[$key]  = $row['volume'];
    $edition[$key] = $row['edition'];
}

// データを  volume  の降順、edition  の昇順にソートする。
// $data  を最後のパラメータとして渡し、同じキーでソートする。
array_multisort($volume, SORT_DESC, $edition, SORT_ASC, $data);
?>
```

データセットの行はソートされ、以下のようになります:

```
volume  |  edition
--------+---------
     98 |        2
     86 |        1
     86 |        6
     85 |        6
     67 |        2
     67 |        7
```

Example#4 大文字・小文字を区別しないソート

SORT_STRING と **SORT_REGULAR** はどちらも大文字・小文字を区別し、 大文字ではじまる文字列が小文字で始まる文字列より前になります。

大文字・小文字を区別しないためには、 元の配列の内容をすべて小文字に変換した配列を用意し、 それをソートの基準にします。

```php
<?php
$array  =  array('Alpha', 'atomic', 'Beta', 'bank');
$array_lowercase  = array_map('strtolower', $array);

array_multisort($array_lowercase, SORT_ASC, SORT_STRING, $array);

print_r($array);
?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => Alpha
     [1] => atomic
     [2] => bank
     [3] => Beta
)
```

---

array_pad

(PHP 4, PHP 5)

array_pad — 指定長、指定した値で配列を埋める

説明

array **array_pad** (array $input , int $pad_size , mixed $pad_value)

array_pad() は、*pad_size* で指定した長さになるように値 *pad_value* で埋めて *input* のコピーを返します。 *pad_size* が正の場合、配列の右側が埋められます。 負の場合、配列の左側が埋められます。 *pad_size* の絶対値が *input* の長さ以下の場合、埋める処理は行われません。 一度に追加できる要素の最大数は 1048576 です。

パラメータ

input

値を埋めるもととなる配列。

pad_size

新しい配列のサイズ。

pad_value

input が *pad_size* より小さいときに、 埋めるために使用する値。

返り値

pad_size で指定した長さになるように値 *pad_value* で埋めて *input* のコピーを返します。 *pad_size* が正の場合、配列の右側が埋められます。 負の場合、配列の左側が埋められます。 *pad_size* の絶対値が *input* の長さ以下の場合、埋める処理は行われません。

例

Example#1 array_pad() の例

```php
<?php
$input = array(12, 10, 9);

$result = array_pad($input, 5, 0);
// 結果は、array(12, 10, 9, 0, 0) です。

$result = array_pad($input, -7, -1);
// 結果は、array(-1, -1, -1, -1, 12, 10, 9) です。

$result = array_pad($input, 2, "noop");
// 埋める処理は行われません。
?>
```

参考

- array_fill()
- range()

---

array_pop

(PHP 4, PHP 5)

array_pop — 配列の末尾から要素を取り除く

説明

mixed **array_pop** (array &$array)

array_pop() は配列 *array* の最後の値を取り出して返します。 配列 *array* は、要素一つ分短くなります。 *array* が空 (または、配列でない) の場合、**NULL** が返されます。

> **注意**: この関数は、配列 (array) ポインタを使用した後にリセット (reset()) します。

パラメータ

array

値を取り出す配列。

返り値

配列 *array* の最後の値を取り出して返します。 *array* が空 (または、配列でない) の場合、 **NULL** が返されます。

例

Example#1 array_pop() の例

```
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_pop($stack);
print_r($stack);
?>
```

この後、*$stack* の要素は三つのみとなります。

```
Array
(
     [0] => orange
     [1] => banana
     [2] => apple
)
```

そして、*raspberry* が *$fruit* に代入されます。

参考

array_push()、 array_shift() および array_unshift() も参照ください。

---

array_product

(PHP 5 >= 5.1.0)

array_product — 配列の値の積を計算する

説明

number **array_product** (array $array)

array_product() は、配列の各要素の積を計算して integer または float で返します。

Example#1 array_product() の例

```
<?php

$a = array(2, 4, 6, 8);
echo "product(a) = " . array_product($a) . "¥n";

?>
```

上の例の出力は以下となります。

```
product(a) = 384
```

---

array_push

(PHP 4, PHP 5)

array_push — 一つ以上の要素を配列の最後に追加する

説明

int **array_push** (array &$array , mixed $var [, mixed $...])

array_push()は、*array* をスタックとして処理し、渡された変数を *array* の最後に加えます。配列 *array* の長さは渡された変数の数だけ増加します。各 *var* 毎に以下を繰り返すことと同じ効果があります。

```
<?php
$array[] = $var;
?>
```

各 *var* で繰り返されます。

処理後の配列の中の要素の数を返します。

Example#1 array_push() の例

```
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

> **注意**: もし配列にひとつの要素を加えるために **array_push()** を使用するなら、 関数を呼ぶオーバーヘッドがないので、*$array[] =* を使用するほうがいいです。

> **注意**: 最初の引数が配列でない場合、**array_push()** は警告を発生させます。これは新規配列を生成する場合における *$var[]* の動作と異なります。

array_pop()、 array_shift() および array_unshift() も参照ください。

---

array_rand

(PHP 4, PHP 5)

array_rand — 配列から一つ以上の要素をランダムに取得する

説明

mixed **array_rand** (array $input [, int $num_req])

array_rand() は、 配列から一つ以上のランダムなエントリを取得しようとする場合に有用です。 この関数は、配列 *input* およびオプションとして *num_req* を引数とします。 この引数は、取得するエントリの数を指定します。 指定されない場合は、デフォルトの 1 になります。

エントリを一つだけ取得する場合、 **array_rand()** はランダムなエントリのキーを返します。 その他の場合は、ランダムなエントリのキーの配列を返します。 これにより、ランダムなキーを取得し、 配列から値を取得することが可能になります。

> **注意**: PHP 4.2.0 以降、 srand() または mt_srand() によりランダム数生成器にシードを与える必要はありません。これは、この処理が自動的に行われるためです。

Example#1 array_rand() の例

```
<?php
srand((float) microtime() * 10000000);
$input = array("ネオ", "モーフィアス", "トリニティ", "サイファー", "タンク");
$rand_keys = array_rand($input, 2);
echo $input[$rand_keys[0]] . "¥n";
echo $input[$rand_keys[1]] . "¥n";
?>
```

shuffle() も参照ください。

---

array_reduce

(PHP 4 >= 4.0.5, PHP 5)

array_reduce — コールバック関数を用いて配列を普通の値に変更することにより、 配列を再帰的に減らす

説明

mixed **array_reduce** (array $input , callback $function [, int $initial])

array_reduce() は、配列 *input* の各要素に *function* 関数を繰り返し適用し、 配列を一つの値に減らします。オプション *intial* が利用可能な場合、処理の最初で使用されたり、 配列が空の場合の最終結果として使用されます。 配列が空で *initial* が渡されなかった場合は、 **array_reduce()** は **NULL** を返します。

Example#1 array_reduce() の例

```
<?php
function rsum($v, $w)
{
    $v += $w;
    return $v;
}

function rmul($v, $w)
{
    $v *= $w;
    return $v;
}

$a = array(1, 2, 3, 4, 5);
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
?>
```

これにより、*$b* の値は *15* となり、*$c* の値は *1200* (= 10*1*2*3*4*5)、そして *$d* の値は *1* となります。

array_filter()、 array_map()、 array_unique() および array_count_values() も参照ください。

---

array_reverse

(PHP 4, PHP 5)

array_reverse — 要素を逆順にした配列を返す

説明

array **array_reverse** (array $array [, bool $preserve_keys])

array_reverse() は、*array* を引数とし、要素の順番を逆にした新しい配列を返します。 この時、 *preserve_keys* が **TRUE** の場合はキーが保持されます。

Example#1 array_reverse() の例

```
<?php
$input  = array("php", 4.0, array("green", "red"));
$result = array_reverse($input);
$result_keyed = array_reverse($input, true);
?>
```

この例において、*$result* と *$result_keyed* は同じ要素をもちますが、 キーが違うことに注意してください。 *$result* と *$result_keyed* の出力は次のようになります:

```
Array
(
    [0] => Array
        (
            [0] => green
            [1] => red
        )

    [1] => 4
    [2] => php
)
Array
```

```
(
     [2] => Array
         (
              [0] => green
              [1] => red
         )

     [1] => 4
     [0] => php
)
```

注意: 2 番目のパラメータは、PHP 4.0.3 で追加されました。

array_flip() も参照ください。

---

array_search

(PHP 4 >= 4.0.5, PHP 5)

array_search — 指定した値を配列で検索し、見つかった場合に対応するキーを返す

説明

mixed **array_search** (mixed $needle , array $haystack [, bool $strict])

haystack において *needle* を検索し、 配列中に見付かった場合にそのキーを返します。そうでない場合に**FALSE**を返します。

注意: もし *needle* が文字列の場合、 大文字小文字を区別して比較が行われます。

注意: PHP 4.2.0 以前では、**array_search()** は、 失敗した場合に **FALSE** ではなく **NULL** を返します。

オプションの3番目のパラメータ *strict* に **TRUE** が指定された場合、**array_search()** は *haystack* の中で *needle* の型に一致するかどうかも確認します。

もし *haystack* に 1 つ以上の *needle* に見つかった場合、 最初にマッチしたキーが返されます。 全てのマッチした値に対するキーを返すためには、代わりに array_keys() にパラメータ *search_value* を付けて使用してください。

Example#1 array_search() の例

```php
<?php
$array = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');

$key = array_search('green', $array); // $key = 2;
$key = array_search('red', $array);   // $key = 1;
?>
```

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される *0* や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

array_keys()、 array_values()、 array_key_exists() および in_array() も参照ください。

---

array_shift

(PHP 4, PHP 5)

array_shift — 配列の先頭から要素を一つ取り出す

説明

mixed **array_shift** (array &$array)

array_shift() は、*array* の最初の値を取り出して返します。配列 *array* は、要素一つ分だけ短くなり、全ての要素は前にずれます。 *array* が空の場合 (または配列でない場合)、 **NULL** が返されます。

注意: この関数は、配列 (array) ポインタを使用した後にリセット (reset()) します。

Example#1 array_shift() の例

```php
<?php
$stack = array("orange", "banana", "apple", "raspberry");
```

```
$fruit = array_shift($stack);
print_r($stack);
?>
```

$stack には3つの要素が残されます:

```
Array
(
     [0] => banana
     [1] => apple
     [2] => raspberry
)
```

そして *orange* が *$fruit* に代入されます。

array_unshift()、 array_push() および array_pop() も参照ください。

---

array_slice

(PHP 4, PHP 5)

array_slice — 配列の一部を展開する

説明

array **array_slice** (array `$array` , int `$offset` [, int `$length` [, bool `$preserve_keys`]])

array_slice()は、*array* から引数 *offset* および *length* で指定された連続する要素を返します。

offset が負の値ではない場合、要素位置の計算は、 配列 *array* の offset から始められます。 *offset* が負の場合、要素位置の計算は *array* の最後から行われます。

length が指定され、正の場合、 連続する複数の要素が返されます。*length* が指定され、負の場合、配列の末尾から連続する複数の要素が返されます。 省略された場合、*offset* から配列の最後までの全ての要素が返されます。

array_slice() はデフォルトで配列の数値キーを並べなおし、 リセットすることに注意してください。PHP 5.0.2 からは、 *preserve_keys* を **TRUE** にする事でこの動作を変更することができます。

Example#1 array_slice() の例

```
<?php
$input = array("a", "b", "c", "d", "e");

$output = array_slice($input, 2);      // "c", "d", "e" を返す
$output = array_slice($input, -2, 1);  // "d" を返す
$output = array_slice($input, 0, 3);   // "a", "b", "c" を返す

// 配列キーの違いに注意
print_r(array_slice($input, 2, -1));
print_r(array_slice($input, 2, -1, true));
?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => c
     [1] => d
)
Array
(
     [2] => c
     [3] => d
)
```

array_splice() および unset() も参照ください。

---

array_splice

(PHP 4, PHP 5)

array_splice — 配列の一部を削除し、他の要素で置換する

説明

array **array_splice** (array &$input , int $offset [, int $length [, array $replacement]])

array_splice() は、配列 *input* から *offset* および *length* で指定された要素を削除し、配列 *replacement* でそれを置換します。この関数は、抽出された要素を含む配列を返します。 *input* の配列の添字は保存されないことに注意しましょう。

offset が正の場合、削除される部分は 配列 *input* の最初から指定オフセットの ぶんだけ進んだ位置からとなります。 *offset* が負の場合、削除される部分は、 *input* の末尾から数えた位置からとなります。

length が省略された場合、 *offset* から配列の最後までが全て削除されます。 *length* が指定され、正の場合、複数の要素が削除されます。 *length* が指定され、負の場合、 削除される部分は配列の末尾から複数の要素となります。 ヒント: *replacement* も指定した場合に *offset* から配列の最後まで全てを削除するには、 *length* を求めるために *count($input)* を使用してください。

配列 *replacement* が指定された場合、 削除された要素は、この配列の要素で置換されます。 *offset* および *length* で何も削除しないと指定した場合、配列 *replacement* の要素は *offset* で指定された位置に挿入されます。 置換される配列のキーは保存されないことに注意してください。 もし *replacement* に一つしか要素がない場合、 要素そのものが配列でない限り、*array()* で括る必要はありません。

以下の文は、同様に *$input* の値を変更します。

array_splice() と同等なもの

array_push($input, $x, $y)	array_splice($input, count($input), 0, array($x, $y))
array_pop($input)	array_splice($input, -1)
array_shift($input)	array_splice($input, 0, 1)
array_unshift($input, $x, $y)	array_splice($input, 0, 0, array($x, $y))
$input[$x] = $y // キーがオフセットと等価な配列に対して	array_splice($input, $x, 1, $y)

削除された要素で構成される配列が返されます。

Example#1 array_splice() の例

```
<?php
$input = array("red", "green", "blue", "yellow");
array_splice($input, 2);
// ここでは $input は array("red", "green") となる

$input = array("red", "green", "blue", "yellow");
array_splice($input, 1, -1);
// ここでは $input は array("red", "yellow") となる

$input = array("red", "green", "blue", "yellow");
array_splice($input, 1, count($input), "orange");
// ここでは $input は array("red", "orange") となる

$input = array("red", "green", "blue", "yellow");
array_splice($input, -1, 1, array("black", "maroon"));
// ここでは $input は array("red", "green",
//            "blue", "black", "maroon") となる

$input = array("red", "green", "blue", "yellow");
array_splice($input, 3, 0, "purple");
// ここでは $input は array("red", "green",
//             "blue", "purple", "yellow"); となる
?>
```

array_slice()、 unset() および array_merge() も参照ください。

---

array_sum

(PHP 4 >= 4.0.4, PHP 5)

array_sum — 配列の中の値の合計を計算する

説明

number **array_sum** (array $array)

array_sum() は、配列の中の値の合計を整数または float として返します。

Example#1 array_sum() の例

```
<?php
$a = array(2, 4, 6, 8);
echo "sum(a) = " . array_sum($a) . "¥n";

$b = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
echo "sum(b) = " . array_sum($b) . "¥n";
```

```
?>
```

上の例の出力は以下となります。

```
sum(a) = 20
sum(b) = 6.9
```

> **注意**: 4.2.1 より前のバージョンの PHP は、渡された配列自体を修正し、 文字列を数値 (これは値にもよるが多くの場合ゼロとなります) に変換していました。

---

array_udiff_assoc

(PHP 5)

array_udiff_assoc — データの比較にコールバック関数を用い、 追加された添字の確認を含めて配列の差を計算する

説明

array **array_udiff_assoc** (array $array1 , array $array2 [, array $ ...], callback $data_compare_func)

array_udiff_assoc() は、 *array1* から他の引数の配列の中に現れない全ての値を含む array を返します。 array_diff() や array_udiff() と異なり、キーが比較に使用されることに注意してください。 配列のデータの比較は、ユーザが指定したコールバックを用いて行われます。 この点で、array_diff_assoc() は反対の動作、つまり 内部関数を利用した比較を行います。

Example#1 array_udiff_assoc() の例

```
<?php
class cr {
     private $priv_member;
     function cr($val)
     {
         $this->priv_member = $val;
     }

     function comp_func_cr($a, $b)
     {
         if ($a->priv_member === $b->priv_member) return 0;
         return ($a->priv_member > $b->priv_member)? 1:-1;
     }
}

$a = array("0.1" => new cr(9), "0.5" => new cr(12), 0 => new cr(23), 1=> new cr(4), 2 => new cr(-15),);
$b = array("0.2" => new cr(9), "0.5" => new cr(22), 0 => new cr(3), 1=> new cr(4), 2 => new cr(-15),);

$result = array_udiff_assoc($a, $b, array("cr", "comp_func_cr"));
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [0.1] => cr Object
         (
             [priv_member:private] => 9
         )

     [0.5] => cr Object
         (
             [priv_member:private] => 12
         )

     [0] => cr Object
         (
             [priv_member:private] => 23
         )
)
```

上の例で、*"1" => new cr(4)* の組み合わせが両方の配列にあること、 そしてそれが関数の出力に含まれていないことが確認できます。

比較は、ユーザが指定したコールバック関数を用いて行います。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ 負の数 / ゼロ / 正の数 を返す必要があります。

> **注意**: この関数は n 次元配列の一つの次元しかチェックしないことに注意してください。 もちろん、*array_udiff_assoc($array1[0], $array2[0], "some_comparison_func");* のようにすることでより深い次元でのチェックもできます。

array_diff()、 array_diff_assoc()、 array_diff_uassoc()、 array_udiff()、 array_udiff_uassoc()、 array_intersect()、 array_intersect_assoc()、 array_uintersect()、 array_uintersect_assoc() および array_uintersect_uassoc() も参照ください。

array_udiff_uassoc

(PHP 5)

array_udiff_uassoc — データと添字の比較にコールバック関数を用い、 追加された添字の確認を含めて配列の差を計算する

説明

array **array_udiff_uassoc** (array $array1 , array $array2 [, array $ ...], callback $data_compare_func , callback $key_compare_func)

array_udiff_uassoc() は、 *array1* から他の引数の配列の中に現れない全ての 値を含むarray を返します。 array_diff() や array_udiff() と異なり、キーが比較に使用されることに注意してください。 配列のデータの比較は、ユーザが指定したコールバック *data_compare_func* を用いて行われます。 この点で、array_diff_assoc() は反対の動作、つまり 内部関数を利用した比較を行います。キー（添字）の比較は、コールバック関数 *key_compare_func* で行われます。 array_udiff_assoc() では後者の比較が内部関数で 行われるという点で、この関数とは異なります。

Example#1 array_udiff_uassoc() の例

```
<?php
class cr {
     private $priv_member;
     function cr($val)
     {
         $this->priv_member = $val;
     }

     function comp_func_cr($a, $b)
     {
         if ($a->priv_member === $b->priv_member) return 0;
         return ($a->priv_member > $b->priv_member)? 1:-1;
     }

     function comp_func_key($a, $b)
     {
         if ($a === $b) return 0;
         return ($a > $b)? 1:-1;
     }
}
$a = array("0.1" => new cr(9), "0.5" => new cr(12), 0 => new cr(23), 1=> new cr(4), 2 => new cr(-15),);
$b = array("0.2" => new cr(9), "0.5" => new cr(22), 0 => new cr(3), 1=> new cr(4), 2 => new cr(-15),);

$result = array_udiff_uassoc($a, $b, array("cr", "comp_func_cr"), array("cr", "comp_func_key"));
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [0.1] => cr Object
         (
              [priv_member:private] => 9
         )

     [0.5] => cr Object
         (
              [priv_member:private] => 12
         )

     [0] => cr Object
         (
              [priv_member:private] => 23
         )
)
```

上の例で、*"1" => new cr(4)* の組み合わせが両方の配列にあること、 そしてそれが関数の出力に含まれていないことが確認できます。 コールバック関数を 2 つ指定しなければならないことに注意してください。

比較は、ユーザが指定したコールバック関数を用いて行います。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ 負の数 / ゼロ / 正の数 を返す必要があります。

> **注意**: この関数は n 次元配列の一つの次元しかチェックしないことに注意してください。 もちろん、*array_udiff_uassoc($array1[0], $array2[0], "data_compare_func", "key_compare_func");* のようにすることでより深い次元でのチェックもできます。

array_diff()、 array_diff_assoc()、 array_diff_uassoc()、 array_udiff()、 array_udiff_assoc()、 array_intersect()、 array_intersect_assoc()、 array_uintersect()、 array_uintersect_assoc() および array_uintersect_uassoc() も参照ください。

array_udiff

(PHP 5)

array_udiff — データの比較にコールバック関数を用い、配列の差を計算する

説明

array **array_udiff** (array $array1 , array $array2 [, array $ ...], callback $data_compare_func)

array_udiff() は、他の引数のいずれにも存在しない *array1* の値の全てを有する配列を返します。 キーと値の関係は維持されることに注意してください。 データの比較には *data_compare_func* が用いられます。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ 負の数 / ゼロ / 正の数 を返す必要があります。 この関数は array_diff() に似ていますが、こちらは データの比較に内部関数を利用します。

Example#1 array_udiff() の例

```
<?php
class cr {
     private $priv_member;
     function cr($val)
     {
         $this->priv_member  = $val;
     }

     function comp_func_cr($a, $b)
     {
         if ($a->priv_member  === $b->priv_member) return 0;
         return ($a->priv_member  > $b->priv_member)? 1:-1;
     }
}
$a =  array("0.1"  => new cr(9), "0.5"  => new cr(12), 0 => new cr(23), 1=> new cr(4), 2 => new cr(-15),);
$b =  array("0.2"  => new cr(9), "0.5"  => new cr(22), 0 => new cr(3), 1=> new cr(4), 2 => new cr(-15),);

$result  = array_udiff($a, $b,  array("cr", "comp_func_cr"));
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [0.5] => cr Object
         (
               [priv_member:private] => 12
         )

     [0] => cr Object
         (
               [priv_member:private] => 23
         )

)
```

> **注意**: この関数は n 次元配列の一つの次元しかチェックしないことに注意してください。 もちろん、*array_udiff($array1[0], $array2[0], "data_compare_func");* のようにすることでより深い次元でのチェックもできます。

array_diff()、 array_diff_assoc()、 array_diff_uassoc()、 array_udiff_assoc()、 array_udiff_uassoc()、 array_intersect()、 array_intersect_assoc()、 array_uintersect()、 array_uintersect_assoc() および array_uintersect_uassoc() も参照ください。

---

array_uintersect_assoc

(PHP 5)

array_uintersect_assoc — データの比較にコールバック関数を用い、 追加された添字の確認も含めて配列の共通項を計算する

説明

array **array_uintersect_assoc** (array $array1 , array $array2 [, array $ ...], callback $data_compare_func)

array_uintersect_assoc() は、全ての引数に現れる *array1* の全ての値を含む配列を返します。 array_uintersect()と異なり、キーが比較に使用される ことに注意してください。データはコールバック関数を用いて比較されます。

Example#1 array_uintersect_assoc() の例

```
<?php
$array1  = array("a"  => "green", "b"  => "brown", "c"  => "blue", "red");
$array2  = array("a"  => "GREEN", "B"  => "brown", "yellow", "red");

print_r(array_uintersect_assoc($array1, $array2, "strcasecmp"));
?>
```

上の例の出力は以下となります。

```
Array
(
    [a] => green
)
```

比較は、ユーザが指定したコールバック関数を利用して行われます。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ負の数 / ゼロ / 正の数 を返す必要があります。

array_uintersect()、 array_intersect_assoc()、 array_intersect_uassoc() および array_uintersect_uassoc() も参照ください。

---

array_uintersect_uassoc

(PHP 5)

array_uintersect_uassoc — データと添字の比較にコールバック関数を用い、 追加された添字の確認も含めて配列の共通項を計算する

説明

array **array_uintersect_uassoc** (array $array1 , array $array2 [, array $ ...], callback $data_compare_func , callback $key_compare_func)

array_uintersect_uassoc() は、全ての引数に現れる *array1* の全ての値を含む配列を返します。 array_uintersect()と異なり、 キーが比較に使用されることに注意してください。 データと添字は、それぞれ個別のコールバック関数を用いて比較されます。

Example#1 array_uintersect_uassoc() の例

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

print_r(array_uintersect_uassoc($array1, $array2, "strcasecmp", "strcasecmp"));
?>
```

上の例の出力は以下となります。

```
Array
(
    [a] => green
    [b] => brown
)
```

比較は、ユーザが指定したコールバック関数を利用して行われます。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ負の数 / ゼロ / 正の数 を返す必要があります。

array_uintersect()、 array_intersect_assoc()、 array_intersect_uassoc() および array_uintersect_assoc() も参照ください。

---

array_uintersect

(PHP 5)

array_uintersect — データの比較にコールバック関数を用い、配列の共通項を計算する

説明

array **array_uintersect** (array $array1 , array $array2 [, array $ ...], callback $data_compare_func)

array_uintersect() は、他の全ての引数に存在する *array1* の値を全て有する配列を返します。 データはコールバック関数を用いて比較されます。

Example#1 array_uintersect() の例

```
<?php
$array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

print_r(array_uintersect($array1, $array2, "strcasecmp"));
?>
```

上の例の出力は以下となります。

```
Array
(
     [a] => green
     [b] => brown
     [0] => red
)
```

比較は、ユーザが指定したコールバック関数を利用して行われます。 この関数は、1 つめの引数が 2 つめより小さい / 等しい / 大きい 場合にそれぞれ負の数 / ゼロ / 正の数 を返す必要があります。

array_intersect()、 array_uintersect_assoc()、 array_intersect_uassoc() および array_uintersect_uassoc() も参照ください。

---

array_unique

(PHP 4 >= 4.0.1, PHP 5)

array_unique — 配列から重複した値を削除する

説明

array **array_unique** (array $array)

array_unique() は、 *array* を入力とし、値に重複のない新規配列を返します。

キーは保持されることに注意してください。 **array_unique()** はまず文字列として値をソートし、 各値の最初のキーを保持し、2回目以降の全てのキーを無視します。 これは、ソート前の *array* で最初の関連する値のキーが保持されるという意味ではありません。

> **注意**: *(string) $elem1 === (string) $elem2* の場合のみ二つの要素は等しいとみなされます。 言い換えると、文字列表現が同じ場合となります。 最初の要素が使用されます。

Example#1 array_unique() の例

```php
<?php
$input = array("a" => "green", "red", "b" => "green", "blue", "red");
$result = array_unique($input);
print_r($result);
?>
```

上の例の出力は以下となります。

```
Array
(
     [a] => green
     [0] => red
     [1] => blue
)
```

Example#2 array_unique() と型

```php
<?php
$input = array(4, "4", "3", 4, 3, "3");
$result = array_unique($input);
var_dump($result);
?>
```

上の例の出力は以下となります。

```
array(2) {
   [0] => int(4)
   [2] => string(1) "3"
}
```

---

array_unshift

(PHP 4, PHP 5)

array_unshift — 一つ以上の要素を配列の最初に加える

説明

int **array_unshift** (array &$array , mixed $var [, mixed $...])

array_unshift() は、*array* の先頭に指定された要素を加えます。リストの要素は全体として加えられるため、 加えられた要素の順番は変わらないことに注意してください。 配列の数値添字はすべて新たにゼロから振りなおされます。 リテラルのキーについては変更されません。

処理後の *array* の要素の数を返します。

Example#1 array_unshift() の例

```
<?php
$queue = array("orange", "banana");
array_unshift($queue, "apple", "raspberry");
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => apple
    [1] => raspberry
    [2] => orange
    [3] => banana
)
```

array_shift()、 array_push() および array_pop() も参照ください。

---

array_values

(PHP 4, PHP 5)

array_values — 配列の全ての値を返す

説明

array **array_values** (array $input)

array_values() は、配列 *input* から全ての値を取り出し、数値添字をつけた配列を返します。

Example#1 array_values() の例

```
<?php
$array = array("size" => "XL", "color" => "gold");
print_r(array_values($array));
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => XL
    [1] => gold
)
```

array_keys() も参照ください。

---

array_walk_recursive

(PHP 5)

array_walk_recursive — 配列の全ての要素に、ユーザー関数を再帰的に適用する

説明

bool **array_walk_recursive** (array &$input , callback $funcname [, mixed $userdata])

input 配列の各要素にユーザ定義関数 *funcname* を適用します。 この関数は配列の要素内を再帰的にたどっていきます。通常、 *funcname* は引数を二つとります。 *input* パラメータの値が最初の引数、 キー/添字は二番目の引数となります。 オプションの *userdata* パラメータが指定された場合、コールバック関数 *funcname* への三番目の引数として渡されます。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意: *funcname* により配列の値そのものを変更する必要がある場合、 *funcname* の最初の引数は 参照 として渡す必要があります。この場合、配列の要素に加えた変更は、 配列自体に対して行われます。

Example#1 array_walk_recursive() の例

```
<?php
$sweet = array('a' => 'apple', 'b' => 'banana');
$fruits = array('sweet' => $sweet, 'sour' => 'lemon');

function test_print($item, $key)
{
    echo "$key holds $item¥n";
}

array_walk_recursive($fruits, 'test_print');
?>
```

上の例の出力は以下となります。

```
a holds apple
b holds banana
sour holds lemon
```

キー 'sweet' が表示されないことにお気づきでしょう。array を要素として持つキーは関数に渡されません。

array_walk() および callback 型に関する情報 も参照ください。

---

array_walk

(PHP 4, PHP 5)

array_walk — 配列の全ての要素にユーザ関数を適用する

説明

bool **array_walk** (array &$array , callback $funcname [, mixed $userdata])

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

array 配列の各要素にユーザ定義関数 *funcname* を適用します。通常、 *funcname* は引数を二つとります。 *array* パラメータの値が最初の引数、 キー/添字は二番目の引数となります。 オプションの *userdata* パラメータが指定された場合、 コールバック関数 *funcname* への三番目の引数として渡されます。

funcname 関数が、指定された引数より多いパラメータを必要とする 場合、**array_walk()**が*funcname* をコールする度にエラーレベル E_WARNING が発生します。これらの警告は、**array_walk()** コールの前に PHP エラー演算子 @ を付けるか、error_reporting() により抑制することができます。

注意: *funcname* により配列の値そのものを変更する必要がある場合、 *funcname* の最初の引数は 参照 として渡す必要があります。この場合、配列の要素に加えた変更は、 配列自体に対して行われます。

注意: キー及び userdata を *funcname* に渡す処理は、 バージョン 4.0.0 で追加されました。

array_walk() は *array* の内部配列ポインタに影響されません。**array_walk()** はポインタの位置に関わらず配列の全てに渡って適用されます。

コールバック関数により配列自身を変更することはできません。 例えば、要素の追加、削除、要素の unset 等はできません。 **array_walk()** が適用される配列を 変更しようとすると、関数の動作を定義できず、 予期しない結果を得ることになります。

Example#1 array_walk() の例

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");

function test_alter(&$item1, $key, $prefix)
{
    $item1 = "$prefix: $item1";
}

function test_print($item2, $key)
{
    echo "$key. $item2<br />¥n";
}

echo "Before ...:¥n";
array_walk($fruits, 'test_print');

array_walk($fruits, 'test_alter', 'fruit');
echo "... and after:¥n";

array_walk($fruits, 'test_print');
?>
```

上の例の出力は以下となります。

```
Before ...:
d. lemon
a. orange
b. banana
c. apple
... and after:
d. fruit: lemon
a. fruit: orange
b. fruit: banana
c. fruit: apple
```

array_walk_recursive()、 create_function()、 list()、 foreach、 each()、 call_user_func_array() および array_map() callback 型に関する情報 も参照ください。

---

array

(PHP 4, PHP 5)

array — 配列を生成する

説明

array **array** ([mixed $...])

パラメータの配列を返します。パラメータには、 =>演算子によりインデックスを指定することもできます。 配列に関するより詳しい情報は配列型 のセクションをご覧ください。

> **注意**: **array()**は、実際にはリテラル配列を表現するための 言語構成要素であり、通常の関数ではありません。

カンマで区切った構文 "index => values" は、インデックスと値を定義します。 インデックスは文字列または数値とすることが可能です。 インデックスが省略された場合、0から始まる整数インデックスが自動的に生成されます。 インデックスが整数の場合、次に生成されるインデックスは、 整数インデックスの最大値 + 1 となります。同じインデックスを二度定義した場合、 後の定義により最初の定義が上書きされることに注意してください。

一般的ではないですが、 最後に定義された配列エントリの後に続くカンマがある場合、 これは有効な構文です。

以下の例では、二次元配列の生成方法、連想配列のキーの指定方法、 および通常の配列において添字番号をスキップし、 それに続く要素にアクセスする方法についてご紹介しています。

Example#1 array() の例

```php
<?php
$fruits = array (
    "fruits"  => array("a" => "orange", "b" => "banana", "c" => "apple"),
    "numbers" => array(1, 2, 3, 4, 5, 6),
    "holes"   => array("first", 5 => "second", "third")
);
?>
```

Example#2 array() における自動インデックス

```php
<?php
$array = array(1, 1, 1, 1,  1, 8 => 1,  4 => 1, 19, 3 => 13);
print_r($array);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
    [9] => 19
)
```

インデックス '3' は二度定義されており、後の値 13 が保持されることに注意してください。 インデックス 4 はインデックス 8 の後に定義され、 次に生成されるインデックス (値は 19) は、最大のインデックスが 8 であるため、9 となります。

次の例は、1 から始まる配列を作成します。

Example#3 array() で 1 から始まる配列を作成

```php
<?php
$firstquarter  = array(1 => 'January', 'February', 'March');
print_r($firstquarter);
?>
```

上の例の出力は以下となります。

```
Array
(
     [1] => January
     [2] => February
     [3] => March
)
```

Perl では、ダブルクオートで囲まれた配列の値にアクセスすることができます。 しかしながら、PHP では配列を中括弧で囲む必要があります。

Example#4 ダブルクオートで囲まれた配列にアクセスする

```php
<?php

$foo = array('bar' => 'baz');
echo "Hello {$foo['bar']}!"; // Hello baz!

?>
```

array_pad()、 list()、 count()、 foreach および range() も参照ください。

---

arsort

(PHP 4, PHP 5)

arsort — 連想キーと要素との関係を維持しつつ配列を逆順にソートする

説明

bool **arsort** (array &$array [, int $sort_flags])

この関数は、連想配列において各配列のキーと要素との関係を維持しつつソートを行います。 この関数は、主に実際の要素の並び方が重要である連想配列をソートするために使われます。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 arsort() の例

```php
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");
arsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val¥n";
}
?>
```

上の例の出力は以下となります。

```
a = orange
d = lemon
b = banana
c = apple
```

fruits はアルファベットの逆順にソートされ、 各要素とキーとの関係は維持されます。

オプションのパラメータ *sort_flags* によりソートの動作を修正可能です。詳細については、 sort() を参照ください。

asort()、rsort()、 ksort() および sort() も参照ください。

---

asort

(PHP 4, PHP 5)

asort — 連想キーと要素との関係を維持しつつ配列をソートする

説明

bool **asort** (array &$array [, int $sort_flags])

この関数は、 連想配列において各配列のキーと要素との関係を維持しつつ配列をソートします。 この関数は、 主に実際の要素の並び方が重要である連想配列をソートするために使われます。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 asort() の例

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");
asort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val¥n";
}
?>
```

上の例の出力は以下となります。

```
c = apple
b = banana
d = lemon
a = orange
```

fruits はアルファベットの昇順にソートされ、 各要素とキーとの関係は維持されます。

オプションのパラメータ *sort_flags* によりソートの動作を修正可能です。 詳細については、sort() を参照ください。

arsort()、rsort()、 ksort() および sort() も参照ください。

---

compact

(PHP 4, PHP 5)

compact — 変数名とその値から配列を作成する

説明

array **compact** (mixed $varname [, mixed $...])

compact() がとるパラメータの数は可変です。 各パラメータは、変数名を値とする文字列か、 変数名の配列のどちらかとすることができます。 配列は、変数名を値とする別の配列を持つことができます。 **compact()**はこれを再帰的に処理します。

各引数について、**compact()** は現在のシンボルテーブルにおいてその名前を有する変数を探し、 変数名がキー、変数の値がそのキーに関する値となるように追加します。 端的に言うと、extract() の逆の動作をします。 追加された全ての変数を値とする出力配列を返します。

設定されていない全ての文字列は、単にスキップされます。

> **注意**: **分かった!** 可変変数 は関数内で PHP の スーパーグローバル配列 と併用してはいけませんので、 スーパーグローバル配列を **compact()** に渡してはいけません。

Example#1 compact() の例

```
<?php
$city  = "San Francisco";
$state = "CA";
$event = "SIGGRAPH";

$location_vars = array("city", "state");

$result = compact("event", "nothing_here", $location_vars);
?>
```

上の例の出力は以下となります。

```
Array
(
    [event] => SIGGRAPH
    [city] => San Francisco
    [state] => CA
)
```

extract() も参照ください。

---

count

(PHP 4, PHP 5)

count — 変数に含まれる要素、 あるいはオブジェクトに含まれるプロパティの数を数える

説明

int **count** (mixed $var [, int $mode])

var に含まれる要素の数を返します。 他のものには、1つの要素しかありませんので、通常 *var* は配列です。

オブジェクトに対して、もし SPL がインストールされている場合、インターフェース *Countable* を実装することで **count()** にフックすることができます。このインターフェースには 1 つのメソッド **count()** があり、**count()** 関数に対する値を返します。

もし *var* が配列もしくは *Countable* インターフェースを実装したオブジェクトではない場合、 *1* が返されます。 ひとつ例外があり、*var* が **NULL** の場合、 *0* が返されます。

> **注意**: オプションの引数 *mode* は PHP 4.2.0 以降で使用可能です。

オプションの*mode* 引数が **COUNT_RECURSIVE** (または 1) にセットされた場合、**count()** は再帰的に配列をカウントします。 これは多次元配列の全ての要素をカウントするといった場合に特に有効です。 *mode* のデフォルトは *0* です。 **count()** は無限の再帰を検出しません。

警告

count() は、セットされていない変数に関して 0 を返しますが、変数が空の配列として初期化されている場合にも 0 を返します。 ある変数がセットされているかどうかを調べるには、 isset() を使用してください。

配列の実装やPHPでの使用法に関する詳細な説明については、マニュアルの 配列のセクションを参照ください。

Example#1 count() の例

```
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
// $result == 3

$b[0]  = 7;
$b[5]  = 9;
$b[10] = 11;
$result = count($b);
// $result == 3

$result = count(null);
// $result == 0

$result = count(false);
// $result == 1
?>
```

Example#2 再帰的な count() の例 (PHP >= 4.2.0)

```
<?php
$food = array('fruits' => array('orange', 'banana', 'apple'),
              'veggie' => array('carrot', 'collard', 'pea'));

// 再帰的なカウント
echo count($food, COUNT_RECURSIVE);  // output 8

// 通常のカウント
echo count($food);                   // output 2

?>
```

is_array()、 isset() および strlen() も参照ください。

---

current

(PHP 4, PHP 5)

current — 配列内の現在の要素を返す

説明

mixed **current** (array &$array)

各配列は、"カレント"の要素へのポインタを有しています。 このポインタは、その配列の最初の要素を指すように初期化されます。

current() 関数は、 単に内部ポインタが現在指している配列要素の値を返します。 この関数は、ポインタを全く移動しません。 内部ポインタが最終要素の次を指していた場合、 **current()** は **FALSE** を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される *0* や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

> **注意**: 配列中に boolean **FALSE** の要素が含まれていると、 それを配列の終わりと区別することができません。**FALSE** 要素を含む配列を順に処理するには、each() 関数を参照ください。

Example#1 current() と類似関数の使用例

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport);    // $mode = 'bike';
$mode = current($transport); // $mode = 'bike';
$mode = prev($transport);    // $mode = 'foot';
$mode = end($transport);     // $mode = 'plane';
$mode = current($transport); // $mode = 'plane';
?>
```

end()、key()、 next()、prev()、 reset() および each() も参照ください。

---

each

(PHP 4, PHP 5)

each — 配列から、次のキーと値のペアを返す

説明

array **each** (array &$array)

配列 *array* から次のキーと値のペアを返し、 配列カーソルを進めます。このペアは 4 つの要素を持つ配列で、 それぞれの要素は *0*, *1*, *key*, *value* というキーを有しています。 要素 *0*と*key* の各々は配列要素のキー名称を保持しており、*1* と *value* の各々はそのデータを保持しています。

配列の内部ポインタが配列の最終要素以降を指す場合、 **each()** は **FALSE** を返します。

Example#1 each() の例

```
<?php
$foo = array("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each($foo);
print_r($bar);
?>
```

$bar は、ここでは以下のキー/値のペアを保持しています。

```
Array
(
    [1] => bob
    [value] => bob
    [0] => 0
    [key] => 0
)
```

```
<?php
$foo = array("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each($foo);
print_r($bar);
?>
```

$bar は次のようなキー/値のペアを持つようになります。

```
Array
(
    [1] => Bob
    [value] => Bob
    [0] => Robert
    [key] => Robert
)
```

通常 **each()** は、配列の走査をするために list() と共に使用します。 例えばこのようになります。

Example#2 each() によって配列を走査する

```
<?php
$fruit = array('a' => 'apple', 'b' => 'banana', 'c' => 'cranberry');

reset($fruit);
while (list($key, $val) = each($fruit)) {
    echo "$key => $val¥n";
}
?>
```

上の例の出力は以下となります。

```
a => apple
b => banana
c => cranberry
```

each() を実行した後、配列カーソルは、 配列の次の要素に移動します。配列の最終要素にカーソルがある場合は、 最終要素にとどまります。再度 each を使用して配列を走査するには、 reset() を使用する必要があります。

警告

配列を他の変数に代入すると、もとの配列の内部ポインタがリセットされます。 そのため、上の例のループ内で *$fruit* を他の変数に代入すると、無限ループを引き起こしてしまいます。

key()、list()、 current()、reset()、 next()、prev() および foreach も参照ください。

---

end

(PHP 4, PHP 5)

end — 配列の内部ポインタを最終要素にセットする

説明

mixed **end** (array &$array)

end() は *array* の内部ポインタを最後の要素まで進め、その値を返します。

Example#1 end() の簡単な例

```
<?php

$fruits = array('apple', 'banana', 'cranberry');
echo end($fruits); // cranberry

?>
```

current()、 each()、 prev()、 next() および reset() も参照ください。

---

extract

(PHP 4, PHP 5)

extract — 配列からシンボルテーブルに変数をインポートする

説明

int **extract** (array $var_array [, int $extract_type [, string $prefix]])

この関数は、配列から現在のシンボルテーブルに変数をインポートするために使用されます。 この関数は連想配列 *var_array* を引数とし、そのキーを変数名、値を変数の値として処理します。 各キー/値の組に関して、*extract_type* および *prefix* パラメータに基づき、 現在のシンボルテーブルに変数を一つ作成します。

> **注意**: バージョン 4.0.5 以降、この関数は展開された変数の数を返します。
>
> **注意**: **EXTR_IF_EXISTS** と **EXTR_PREFIX_IF_EXISTS** はバージョン 4.2.0 で導入されました。
>
> **注意**: **EXTR_REFS** はバージョン 4.3.0 で導入されました。

extract() は、各キーについて変数名として有効であるかどうか、 そして、シンボルテーブルの既存の変数と衝突しないかどうかを確認します。 無効ま

たは数値キーおよび衝突に関する対処法は、 *extract_type* で定義されます。 これは以下の値のどれかとなります。

EXTR_OVERWRITE
衝突があった場合、存在する変数が上書きされます。
EXTR_SKIP
衝突があった場合、存在する変数は上書きされません。
EXTR_PREFIX_SAME
衝突があった場合、*prefix* を前につけた新しい変数となります。
EXTR_PREFIX_ALL
全ての変数の前に *prefix* を付けます。PHP 4.0.5 以降、接頭辞を数値とすることも可能です。
EXTR_PREFIX_INVALID
無効または数値の変数名のみに接頭辞 *prefix* を付ける。このフラグは、PHP 4.0.5 で追加されました。
EXTR_IF_EXISTS
カレントのシンボルテーブルに既に存在する場合にのみ上書きします。 例えば $_REQUEST 以外にあなたが定義した変数のみを展開し 有効な変数としたいような場合に有用です。このフラグは PHP 4.2.0 で追加されました。
EXTR_PREFIX_IF_EXISTS
同じ変数だが接頭辞をつけていないバージョンの変数が カレントのシンボルテーブルに存在する場合にのみ 変数を生成します。このフラグは PHP 4.2.0 で追加されました。
EXTR_REFS
変数を参照として展開します。 これはインポート済みの変数が、 *var_array* パラメータの値に常に参照付けられることを意味します。 このフラグを単独で使用するか、 あるいは*extract_type* と和算することにより、 他のフラグとそれを組み合わせることができます。 このフラグは PHP 4.3.0 で追加されました。

extract_type が指定されない場合、 **EXTR_OVERWRITE** とみなされます。

prefix は、 *extract_type* が **EXTR_PREFIX_SAME**、**EXTR_PREFIX_ALL**、 **EXTR_PREFIX_INVALID** あるいは **EXTR_PREFIX_IF_EXISTS** の場合にのみ必要であることに注意してください。 接頭辞を付けた変数名が有効な変数名でない場合、 この変数はシンボルテーブルにインポートされません。接頭辞は、 アンダースコア文字で配列のキーから自動的に分割されます。

extract() は、 各キーが有効な変数名からなるかどうかを確認し、 有効な変数名である場合のみインポート処理を行います。

警告

extract() をユーザー入力 ($_GET, ...) のような信頼できないデータについて使用しないでください。 もし行う場合、例えば register_globals を信頼しているような古いコードを一時的に実行したい場合、 **EXTR_SKIP** のような *extract_type* の値が上書きされていないことを確認してください。そして *php.ini* の variables_order で定義されたものと同じ順で展開すべきであることに留意してください。

extract の使用例としては、シンボルテーブルに wddx_deserialize() から返された連想配列を インポートすることが考えられます。

Example#1 extract() の例

```
<?php

/*  $var_array  はwddx_deserializeから返された配列と仮定します
 */

$size  =  "large";
$var_array  =  array("color"  =>  "blue",
                     "size"   =>  "medium",
                     "shape"  =>  "sphere");
extract($var_array, EXTR_PREFIX_SAME, "wddx");

echo  "$color,  $size,  $shape,  $wddx_size¥n";

?>
```

上の例の出力は以下となります。

```
blue,  large,  sphere,  medium
```

EXTR_PREFIX_SAME を指定したため、*$size* は上書きされず、*$wddx_size* が作成されます。 **EXTR_SKIP** が指定された場合、$wddx_sizeは作成されません。 **EXTR_OVERWRITE** の場合は、*$size* の値は "medium" となります。 **EXTR_PREFIX_ALL** の場合は新規の変数 *$wddx_color*, *$wddx_size*, *$wddx_shape* が作成されます。

連想配列を使用する必要があります。**EXTR_PREFIX_ALL** または **EXTR_PREFIX_INVALID** を使用しない限り、 数値添字の配列には結果は出力されません。

compact() も参照ください。

---

in_array

(PHP 4, PHP 5)

in_array — 配列に値があるかチェックする

説明

bool **in_array** (mixed $needle , array $haystack [, bool $strict])

needle で *haystack* を検索し、配列にそれがあった場合に **TRUE**、それ以外の場合は、**FALSE** を返します。

三番目のパラメータ *strict* が **TRUE** に設定された場合、 **in_array()** は、*haystack* の中の *needle* の 型も確認します。

> **注意**: *needle* が文字列の場合、 比較の際に大文字小文字は区別されます。
>
> **注意**: PHP 4.2.0 以前では *needle* に配列を使用することはできませんでした。

Example#1 in_array() の例

```
<?php
$os = array("Mac", "NT", "Irix", "Linux");
if (in_array("Irix", $os)) {
    echo "Got Irix";
}
if (in_array("mac", $os)) {
    echo "Got mac";
}
?>
```

二番目の条件式は失敗します。**in_array()** は大文字小文字を区別するからです。したがって次のような出力になります。

```
Got Irix
```

Example#2 strict を指定した in_array() の例

```
<?php
$a = array('1.10', 12.4, 1.13);

if (in_array('12.4', $a, true)) {
    echo "'12.4' found with strict check¥n";
}

if (in_array(1.13, $a, true)) {
    echo "1.13 found with strict check¥n";
}
?>
```

上の例の出力は以下となります。

```
1.13 found with strict check
```

Example#3 needleが配列の場合の in_array()

```
<?php
$a = array(array('p', 'h'), array('p', 'r'), 'o');

if (in_array(array('p', 'h'), $a)) {
    echo "'ph' was found¥n";
}

if (in_array(array('f', 'i'), $a)) {
    echo "'fi' was found¥n";
}

if (in_array('o', $a)) {
    echo "'o' was found¥n";
}
?>
```

上の例の出力は以下となります。

```
   'ph' was found
   'o' was found
```

array_search()、 array_key_exists() および isset() も参照ください。

---

key

(PHP 4, PHP 5)

key — 連想配列からキーを取り出す

説明

mixed **key** (array &$array)

key() は、 現在の配列位置における連想配列要素のキーを返します。

Example#1 key() の例

```
<?php
$array = array(
    'fruit1' => 'apple',
    'fruit2' => 'orange',
    'fruit3' => 'grape',
    'fruit4' => 'apple',
    'fruit5' => 'apple');

// このループは値が "apple" である
// 全ての連想配列のキーを表示する
while ($fruit_name = current($array)) {
    if ($fruit_name == 'apple') {
        echo key($array).'<br />';
    }
    next($array);
}
?>
```

current() および next() も参照ください。

---

krsort

(PHP 4, PHP 5)

krsort — 配列をキーで逆順にソートする

説明

bool **krsort** (array &$array [, int $sort_flags])

配列をキーにより逆順にソートします。キーとデータとの関係は維持されます。 この関数は主に連想配列の場合に有用です。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 krsort() の例

```
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
krsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val¥n";
}
?>
```

上の例の出力は以下となります。

```
d = lemon
c = apple
b = banana
a = orange
```

オプションのパラメータ *sort_flags* によりソートの動作を修正可能です。詳細については、 sort() を参照ください。

asort()、arsort()、 ksort()、sort()、 natsort() および rsort() も参照ください。

---

ksort

(PHP 4, PHP 5)

ksort — 配列をキーでソートする

説明

bool **ksort** (array &$array [, int $sort_flags])

キーとデータの関係を維持しつつ、配列をキーでソートします。 この関数は、主として連想配列において有用です。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 ksort() の例

```
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val¥n";
}
?>
```

上の例の出力は以下となります。

```
a = orange
b = banana
c = apple
d = lemon
```

オプションのパラメータ *sort_flags* によりソートの動作を修正可能です。詳細については、 sort() を参照ください。

asort()、arsort()、 krsort()、uksort()、 sort()、natsort() および rsort() も参照ください。

> **注意**: 二番目のパラメータは、PHP 4 で追加されました。

---

list

(PHP 4, PHP 5)

list — 配列と同様の形式で、複数の変数への代入を行う

説明

void **list** (mixed $varname , mixed $...)

array() と同様に、 この関数は実際には関数ではなく言語の構成要素です。 **list()** は、 単一の操作で一連の変数に値を代入するために使われます。

> **注意**: **list()** は数値添字の配列のみを生成します。 添字は 0 から始まります。

Example#1 list() の例

```
<?php

$info = array('コーヒー', '茶色', 'カフェイン');

// すべての変数の取得
list($drink, $color, $power) = $info;
echo "$drink の色は $color で、$power が含まれています。\n";

// 一部の変数の取得
list($drink, , $power) = $info;
echo "$drink には $power が含まれています。\n";

// 三番目のみの取得
list( , , $power) = $info;
echo "$power 欲しい!\n";

// list() は文字列では動作しません
list($bar) = "abcde";
var_dump($bar); // NULL
?>
```

Example#2 list() の使用法の例

```
<table>
  <tr>
    <th>社員氏名</th>
    <th>給与</th>
  </tr>

<?php

$result = mysql_query("SELECT id, name, salary FROM employees", $conn);
while (list($id, $name, $salary) = mysql_fetch_row($result)) {
    echo "  <tr>¥n" .
         "    <td><a href=¥"info.php?id=$id¥">$name</a></td>¥n" .
         "    <td>$salary</td>¥n" .
         "  </tr>¥n";
}
```

```
?>
```

```
</table>
```

警告

list()は、最も右のパラメータから値を代入します。 プレーンな変数を使用している場合には、このことを気にする必要はありません。 しかし、添字配列を使用している場合には、配列の添字の順番が **list()** に書いたものと同じく左から右となることを通常は期待しますが、 そうはなりません。この配列の添字は逆の順番となります。

Example#3 配列の添字を使用した list() の例

```
<?php

$info = array('coffee', 'brown', 'caffeine');

list($a[0], $a[1], $a[2]) = $info;

var_dump($a);

?>
```

次のような出力になります(**list()** の文法に書かれた 順番と、要素の順番の違いに注意):

```
array(3) {
   [2]=>
    string(8) "caffeine"
   [1]=>
    string(5) "brown"
   [0]=>
    string(6) "coffee"
}
```

each()、array() および extract() も参照ください。

---

natcasesort

(PHP 4, PHP 5)

natcasesort — 大文字小文字を区別しない"自然順"アルゴリズムを用いて配列をソートする

説明

bool **natcasesort** (array &$array)

この関数は、人間が行うような手法でアルファベットまたは数字の文字列の順番を キー/値の関係を保持したままソートします。 これは、"自然順(natural ordering)"と呼ばれているものです。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

natcasesort() は、大文字小文字を区別しないバージョンの natsort() です。

Example#1 natcasesort() の例

```
<?php
$array1 = $array2 = array('IMG0.png', 'img12.png', 'img10.png', 'img2.png', 'img1.png', 'IMG3.png');

sort($array1);
echo "Standard sorting¥n";
print_r($array1);

natcasesort($array2);
echo "¥nNatural order sorting (case-insensitive)¥n";
print_r($array2);
?>
```

上の例の出力は以下となります。

```
Standard sorting
Array
(
     [0] => IMG0.png
     [1] => IMG3.png
     [2] => img1.png
     [3] => img10.png
     [4] => img12.png
     [5] => img2.png
)

Natural order sorting (case-insensitive)
Array
```

```
(
     [0] => IMG0.png
     [4] => img1.png
     [3] => img2.png
     [5] => IMG3.png
     [2] => img10.png
     [1] => img12.png
)
```

より詳細な情報については、Martin Poolの» Natural Order String Comparison ページを参照ください。

sort()、 natsort()、 strnatcmp() および strnatcasecmp() も参照ください。

---

natsort

(PHP 4, PHP 5)

natsort — "自然順"アルゴリズムで配列をソートする

説明

bool **natsort** (array &$array)

この関数は、人間が行うような手法でアルファベットまたは数字の文字列の順番を キー/値の関係を保持したままソートします。 これは、"自然順 (natural ordering)"と呼ばれているものです。 このアルゴリズムと (sort() を用いた) 通常のコンピュータ文字列ソートアルゴリズムの違いを示す例を以下に示します。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 natsort() の例

```php
<?php
$array1 = $array2 = array("img12.png", "img10.png", "img2.png", "img1.png");

sort($array1);
echo "Standard sorting¥n";
print_r($array1);

natsort($array2);
echo "¥nNatural order sorting¥n";
print_r($array2);
?>
```

上の例の出力は以下となります。

```
Standard sorting
Array
(
     [0] => img1.png
     [1] => img10.png
     [2] => img12.png
     [3] => img2.png
)

Natural order sorting
Array
(
     [3] => img1.png
     [2] => img2.png
     [1] => img10.png
     [0] => img12.png
)
```

より詳細な情報については、Martin Poolの» Natural Order String Comparison ページを参照ください。

natcasesort()、 strnatcmp() および strnatcasecmp() も参照ください。

---

next

(PHP 4, PHP 5, PECL axis2:0.1.0-0.1.1 xmlreader:1.0-1.0.1)

next — 内部配列ポインタを進める

説明

mixed **next** (array &$array)

内部配列ポインタが次の場所を指すようにし、(ポインタ移動後の) その配列値を返します。それ以上要素がない場合は **FALSE** を返します。

next() は、ひとつの違いを除いて current() と同じです。 **next()** は要素を返す前に内部配列ポインタをひとつ先に進めます。 つまり、次の配列要素を返すとともに内部配列ポインタをひとつ進めるということです。 もし内部配列ポインタをひとつ進めた結果、 要素リストの最後の先まで行ってしまった場合、 **next()**は **FALSE** を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される *0* や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

> **注意**: 配列中に boolean **FALSE** の要素が含まれていると、 それを配列の終わりと区別することができません。**FALSE** 要素を含む配列を順に処理するには、each() 関数を参照ください。

Example#1 next() および類似関数の使用例

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport);    // $mode = 'bike';
$mode = next($transport);    // $mode = 'car';
$mode = prev($transport);    // $mode = 'bike';
$mode = end($transport);     // $mode = 'plane';
?>
```

current()、end()、 prev()、reset()、 および each() も参照ください。

---

pos

(PHP 4, PHP 5)

pos — current() のエイリアス

説明

この関数は次の関数のエイリアスです。 current()

---

prev

(PHP 4, PHP 5)

prev — 内部の配列ポインタをひとつ前に戻す

説明

mixed **prev** (array &$array)

内部の配列ポインタが指している前の場所の配列値を返します。 もう要素がない場合は **FALSE** を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される *0* や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

> **注意**: 配列中に boolean **FALSE** の要素が含まれていると、 それを配列の先頭と区別することができません。**FALSE** 要素を含む配列を順に処理するには、each() 関数を参照ください。

prev() は、 内部の配列ポインタを進めるのではなく戻すということを除けば next() と同じです。

Example#1 prev() および類似関数の使用例

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport);    // $mode = 'bike';
$mode = next($transport);    // $mode = 'car';
$mode = prev($transport);    // $mode = 'bike';
$mode = end($transport);     // $mode = 'plane';
?>
```

current()、end()、 next()、reset()、 および each() も参照ください。

range

(PHP 4, PHP 5)

range — ある範囲の整数を有する配列を作成する

説明

array **range** (mixed $low , mixed $high [, number $step])

range() は、*low* から *high* までの整数の配列を返します。 low > high の場合、順番は high から low となります。

> **注意**: **新しい引数** オプションの引数 *step* が PHP 5.0.0 で追加されました。

step が指定されている場合、それは 要素毎の増加数となります。*step* は正の数でなければなりません。デフォルトは 1 です。

Example#1 range() の例

```
<?php
// array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
foreach (range(0, 12) as $number) {
    echo $number;
}

// step 引数は PHP 5.0.0 以降で使用できます
// array(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
foreach (range(0, 100, 10) as $number) {
    echo $number;
}

// 文字列シーケンスは PHP 4.1.0 以降で使用できます
// array('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i');
foreach (range('a', 'i') as $letter) {
    echo $letter;
}
// array('c', 'b', 'a');
foreach (range('c', 'a') as $letter) {
    echo $letter;
}
?>
```

> **注意**: 4.1.0 より前のバージョンでは、**range()** 関数は、 昇順の整数配列のみを生成しました。 文字シーケンス及び降順の配列のサポートが 4.1.0 で追加されました。 文字のシーケンスは一文字のみに限定されます。それより長い文字が指定された場合は、 最初の文字のみが使用されます。

警告

PHP 4.1.0 から 4.3.2 までは、**range()** は数値文字を文字 (string) として認識し、数値 (integer) としては認識しません。 その代わり、文字列シーケンスが使用されます。例えば、 *"4242"* は *"4"* として扱われます。

shuffle()、 array_fill() および foreach も参照ください。

---

reset

(PHP 4, PHP 5)

reset — 配列の内部ポインタを先頭の要素にセットする

説明

mixed **reset** (array &$array)

reset() は、*array* の内部ポインタの先頭の要素に戻し、配列の最初の要素の値か、 配列が空の場合 **FALSE** を返します。

Example#1 reset() の例

```
<?php

$array = array('step one', 'step two', 'step three', 'step four');

// デフォルトでは、ポインタは先頭要素を指しています
echo current($array) . "<br />¥n"; // "step one"

// 次の 2 ステップをとばします
next($array);
next($array);
echo current($array) . "<br />¥n"; // "step three"
```

```
// ポインタをリセットし、再度ステップ 1 開始します
reset($array);
echo current($array) . "<br />¥n"; // "step one"

?>
```

current()、 each()、end()、 next() および prev() も参照ください。

---

rsort

(PHP 4, PHP 5)

rsort — 配列を逆順にソートする

説明

bool **rsort** (array &$array [, int $sort_flags])

この関数は、配列を逆順に(高位から低位に)ソートします。

> **注意**: この関数は、 *array* パラメータの要素に対して新しいキーを割り当てます。その際、単純にキーを並べ替える代わりに、すでに割り当てられている既存のキーを削除してしまいます。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 rsort() の例

```
<?php
$fruits = array("lemon", "orange", "banana", "apple");
rsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val¥n";
}
?>
```

上の例の出力は以下となります。

```
0 = orange
1 = lemon
2 = banana
3 = apple
```

fruits はアルファベットの逆順にソートされました。

オプションのパラメータ *sort_flags* によりソートの動作を修正可能です。詳細については、 sort() を参照ください。

arsort()、 asort()、ksort()、 krsort()、 sort() および usort() も参照ください。

---

shuffle

(PHP 4, PHP 5)

shuffle — 配列をシャッフルする

説明

bool **shuffle** (array &$array)

この関数は、配列をシャッフル (要素の順番をランダムに) します。

> **注意**: この関数は、 *array* パラメータの要素に対して新しいキーを割り当てます。その際、単純にキーを並べ替える代わりに、すでに割り当てられている既存のキーを削除してしまいます。

Example#1 shuffle() の例

```
<?php
$numbers = range(1, 20);
srand((float)microtime() * 1000000);
shuffle($numbers);
foreach ($numbers as $number) {
    echo "$number ";
}
?>
```

> **注意**: PHP 4.2.0 以降、 srand() または mt_srand() によりランダム数生成器にシードを与える必要はありません。これは、この処理が自動的に行われるためです。

arsort()、asort()、 ksort()、rsort()、 sort() および usort() も参照ください。

---

sizeof

(PHP 4, PHP 5)

sizeof — count() のエイリアス

説明

この関数は次の関数のエイリアスです。 count().

---

sort

(PHP 4, PHP 5)

sort — 配列をソートする

説明

bool **sort** (array `&$array` [, int `$sort_flags`])

この関数は配列をソートします。この関数が正常に終了すると、 各要素は低位から高位へ並べ替えられます。

> **注意**: この関数は、 *array* パラメータの要素に対して新しいキーを割り当てます。その際、単純にキーを並べ替える代わりに、すでに割り当てられている既存のキーを削除してしまいます。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 sort() の例

```
<?php

$fruits = array("lemon", "orange", "banana", "apple");
sort($fruits);
foreach ($fruits as $key => $val) {
    echo "fruits[" . $key . "] = " . $val . "¥n";
}

?>
```

上の例の出力は以下となります。

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

fruits はアルファベットの昇順にソートされました。

オプションの 2 番目のパラメータ *sort_flags* は、以下の値によりソートの動作を修正するために使用することが可能です。

ソート型のフラグ:

- **SORT_REGULAR** - 通常通りに項目を比較 (型は変更しません)
- **SORT_NUMERIC** - 数値的に項目を比較
- **SORT_STRING** - 文字列として項目を比較
- **SORT_LOCALE_STRING** - は、カレントのロケールに に基づき比較を行います。PHP 4.4.0 と PHP 5.0.2で追加されました。 PHP 6 より前のバージョンではシステムロケールを使用します。これは setlocale() を使用して変更可能です。 PHP 6 以降では、**i18n_loc_set_default()** 関数を 使用する必要があります。

> **注意**: 2 番目の引数は PHP 4 で追加されました。

警告

複数の型が混在する配列をソートする場合には、注意してください。 **sort()** が予測不可能な結果を出力することがあります。

arsort()、 asort()、ksort()、 krsort()、 natsort()、natcasesort()、 rsort()、usort()、 array_multisort() および uksort() も参照ください。

---

uasort

(PHP 4, PHP 5)

uasort — ユーザー定義の比較関数で配列をソートし、連想インデックスを保持する

説明

bool **uasort** (array &$array , callback $cmp_function)

この関数は、配列インデックスが関連する配列要素との関係を保持するような配列をソートします。 主に実際の配列の順序に意味がある連想配列をソートするためにこの関数は使用されます。 比較関数はユーザーが定義します。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

> **注意**: ユーザ定義の比較関数の例については、 usort() および uksort() を参照ください。

usort()、uksort()、 sort()、asort()、 arsort()、ksort() および rsort() も参照ください。

---

uksort

(PHP 4, PHP 5)

uksort — ユーザー定義の比較関数を用いて、キーで配列をソートする

説明

bool **uksort** (array &$array , callback $cmp_function)

uksort() は、 ユーザー定義の比較関数を用いて配列のキーをソートします。 ソートしたい配列を複雑な基準でソートする必要がある場合には、 この関数を使う必要があります。

関数 *cmp_function* は、 *array* のキーペアによって満たされる 2 つのパラメータを受け取ります。 この比較関数が返す値は、最初の引数が二番目より小さい場合は負の数、 等しい場合はゼロ、そして大きい場合は正の数でなければなりません。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 uksort() の例

```
<?php
function cmp($a, $b)
{
    $a = ereg_replace('^(a|an|the) ', '', $a);
    $b = ereg_replace('^(a|an|the) ', '', $b);
    return strcasecmp($a, $b);
}

$a = array("John" => 1, "the Earth" => 2, "an apple" => 3, "a banana" => 4);

uksort($a, "cmp");

foreach ($a as $key => $value) {
    echo "$key: $value¥n";
}
?>
```

上の例の出力は以下となります。

```
an apple: 3
a banana: 4
the Earth: 2
John: 1
```

usort()、uasort()、 sort()、asort()、 arsort()、ksort()、 natsort() および rsort() も参照ください。

---

usort

(PHP 4, PHP 5)

usort — ユーザー定義の比較関数を使用して、配列を値でソートする

説明

bool **usort** (array &$array , callback $cmp_function)

この関数は、ユーザー定義の比較関数により配列をその値でソートします。 ソートしたい配列を複雑な基準でソートする必要がある場合、 この関数を使用するべきです。

比較関数は、最初の引数が 2 番目の引数より小さいか、等しいか、大きい場合に、 それぞれゼロ未満、ゼロに等しい、ゼロより大きい整数を返す 必要があります。

> **注意**: 二つのメンバーの比較結果が等しいとなった場合、 ソートされた配列の順番は定義されません。PHP 4.0.6 までは、 ユーザー定義関数はそれらの要素の順番を維持します。しかし PHP 4.1.0 以降で導入された新しいソートアルゴリズムでは、 それと同等のことを行う効果的な方法はありません。

> **注意**: この関数は、 *array* パラメータの要素に対して新しいキーを割り当てます。その際、単純にキーを並べ替える代わりに、すでに割り当てられている既存のキーを削除してしまいます。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 usort() の例

```
<?php
function cmp($a, $b)
{
    if ($a == $b) {
        return 0;
    }
    return ($a < $b) ? -1 : 1;
}

$a = array(3, 2, 5, 6, 1);

usort($a, "cmp");

foreach ($a as $key => $value) {
    echo "$key: $value¥n";
}
?>
```

上の例の出力は以下となります。

```
0: 1
1: 2
2: 3
3: 5
4: 6
```

> **注意**: もちろん、このような簡単な例では rsort() 関数の方がより適当です。

Example#2 多次元配列を使用する usort() の例

```
<?php
function cmp($a, $b)
{
    return strcmp($a["fruit"], $b["fruit"]);
}

$fruits[0]["fruit"] = "lemons";
$fruits[1]["fruit"] = "apples";
$fruits[2]["fruit"] = "grapes";

usort($fruits, "cmp");

while (list($key, $value) = each($fruits)) {
    echo "¥$fruits[$key]: " . $value["fruit"] . "¥n";
}
?>
```

多次元配列をソートする際には、*$a* と *$b* は配列の最初のインデックスへの参照を保持しています。

上の例の出力は以下となります。

```
$fruits[0]: apples
$fruits[1]: grapes
$fruits[2]: lemons
```

Example#3 usort() でオブジェクトのメンバ関数を使用する例

```
<?php
class TestObj {
    var $name;

    function TestObj($name)
    {
        $this->name = $name;
    }

    /* This is the static comparing function: */
    function cmp_obj($a, $b)
    {
        $al = strtolower($a->name);
        $bl = strtolower($b->name);
        if ($al == $bl) {
            return 0;
        }
        return ($al > $bl) ? +1 : -1;
    }
}

$a[] = new TestObj("c");
$a[] = new TestObj("b");
$a[] = new TestObj("d");

usort($a, array("TestObj", "cmp_obj"));

foreach ($a as $item) {
    echo $item->name . "¥n";
}
?>
```

上の例の出力は以下となります。

```
b
c
d
```

uasort()、 uksort()、sort()、 asort()、 arsort()、ksort()、 natsort() および rsort() も参照ください。

目次

- array_slice — 配列の一部を展開する
- array_splice — 配列の一部を削除し、他の要素で置換する
- array_sum — 配列の中の値の合計を計算する
- array_udiff_assoc — データの比較にコールバック関数を用い、 追加された添字の確認を含めて配列の差を計算する
- array_udiff_uassoc — データと添字の比較にコールバック関数を用い、 追加された添字の確認を含めて配列の差を計算する
- array_udiff — データの比較にコールバック関数を用い、配列の差を計算する
- array_uintersect_assoc — データの比較にコールバック関数を用い、 追加された添字の確認も含めて配列の共通項を計算する
- array_uintersect_uassoc — データと添字の比較にコールバック関数を用い、 追加された添字の確認も含めて配列の共通項を計算する
- array_uintersect — データの比較にコールバック関数を用い、配列の共通項を計算する
- array_unique — 配列から重複した値を削除する
- array_unshift — 一つ以上の要素を配列の最初に加える
- array_values — 配列の全ての値を返す
- array_walk_recursive — 配列の全ての要素に、ユーザー関数を再帰的に適用する
- array_walk — 配列の全ての要素にユーザ関数を適用する
- array — 配列を生成する
- arsort — 連想キーと要素との関係を維持しつつ配列を逆順にソートする
- asort — 連想キーと要素との関係を維持しつつ配列をソートする
- compact — 変数名とその値から配列を作成する
- count — 変数に含まれる要素、 あるいはオブジェクトに含まれるプロパティの数を数える
- current — 配列内の現在の要素を返す
- each — 配列から、次のキーと値のペアを返す
- end — 配列の内部ポインタを最終要素にセットする
- extract — 配列からシンボルテーブルに変数をインポートする
- in_array — 配列に値があるかチェックする
- key — 連想配列からキーを取り出す
- krsort — 配列をキーで逆順にソートする
- ksort — 配列をキーでソートする
- list — 配列と同様の形式で、複数の変数への代入を行う
- natcasesort — 大文字小文字を区別しない"自然順"アルゴリズムを用いて配列をソートする
- natsort — "自然順"アルゴリズムで配列をソートする
- next — 内部配列ポインタを進める
- pos — current のエイリアス
- prev — 内部の配列ポインタをひとつ前に戻す
- range — ある範囲の整数を有する配列を作成する
- reset — 配列の内部ポインタを先頭の要素にセットする
- rsort — 配列を逆順にソートする
- shuffle — 配列をシャッフルする
- sizeof — count のエイリアス
- sort — 配列をソートする
- uasort — ユーザー定義の比較関数で配列をソートし、連想インデックスを保持する
- uksort — ユーザー定義の比較関数を用いて、キーで配列をソートする
- usort — ユーザー定義の比較関数を使用して、配列を値でソートする

Aspell関数(古い拡張モジュール)

導入

aspell() 関数により単語のスペルチェックおよび 修正案の提供が可能となります。

> **注意**: このエクステンションは、PHP 4.3.0以降、PHPから削除されています。 PHPでスペルチェックの機能を使 用したい場合は、代わりにpspellを 使用してください。このモジュールは、pspellライブラリを使用し、より新しい バージョンのaspellと組み合わせても動作します。

要件

aspellが動作するのは、非常に古い(.27.*まで)バージョンのaspellライ ブラリのみです。このモジュールも、これらのバージョンのaspellライブ ラリも将来的にもサポートされません。 » http://aspell.sourceforge.net/から取得可能なaspell ライブラリが必要です。

インストール手順

PHP 4では、これらの関数は、PHPが --with-aspell=[DIR]を指定されて いる場合のみ利用可能です。

参考

pspellも参照してください。

---

aspell_check_raw

(PHP 4 <= 4.2.3)

aspell_check_raw — 大文字小文字の変更や削除を行うことなく、単語のチェックを行う [非推奨]

説明

bool **aspell_check_raw** (int `$dictionary_link` , string `$word`)

aspell_check_raw() は単語のスペルをチェックします。 大文字小文字変換や空白の削除は行いません。

パラメータ

`dictionary_link`

aspell_new() が返す辞書リンク識別子。

`word`

調べる単語。

返り値

スペルが正しい場合に **`TRUE`**、正しくない場合に **`FALSE`** を返します。

例

Example#1 aspell_check_raw() の例

```
<?php

$aspell_link  = aspell_new("english");

if (aspell_check_raw($aspell_link, "test")) {
    echo "正しいスペルです";
} else {
    echo "スペルが間違っています";
}

?>
```

参考

- aspell_check()

---

aspell_check

(PHP 4 <= 4.2.3)

aspell_check — 単語をチェックする [非推奨]

説明

bool **aspell_check** (int `$dictionary_link` , string `$word`)

aspell_check() は単語のスペルをチェックします。

パラメータ

`dictionary_link`

aspell_new() が返す辞書リンク識別子。

`word`

調べる単語。

返り値

スペルが正しい場合に **TRUE**、正しくない場合に **FALSE** を返します。

例

Example#1 aspell_check() の例

```
<?php

$aspell_link  = aspell_new("english");

if (aspell_check($aspell_link, "testt")) {
    echo "正しいスペルです";
} else {
    echo "スペルが間違っています";
}

?>
```

参考

- aspell_check_raw()

---

aspell_new

(PHP 4 <= 4.2.3)

aspell_new — 新しい辞書を読み込む [非推奨]

説明

int **aspell_new** (string `$master` [, string `$personal`])

aspell_new() は、 他の aspell 関数で使用する新しい辞書をオープンします。

パラメータ

master

言語。

personal

デフォルトは空の文字列です。

返り値

辞書リンク識別子、あるいはエラーの場合に **FALSE** を返します。

例

Example#1 aspell_new() の例

```
<?php
$aspell_link  = aspell_new("english");
?>
```

---

aspell_suggest

(PHP 4 <= 4.2.3)

aspell_suggest — 単語スペルの修正案を示す [非推奨]

説明

array **aspell_suggest** (int `$dictionary_link` , string `$word`)

aspell_suggest() は、指定した *word* に関するスペルを提案します。

パラメータ

dictionary_link

aspell_new() が返す辞書リンク識別子。

word

調べる単語。

返り値

提案内容の配列を返します。

例

Example#1 aspell_suggest() の例

```
<?php

$aspell_link  = aspell_new("english");

if (!aspell_check($aspell_link, "test")) {
    $suggestions  = aspell_suggest($aspell_link, "test");

     foreach ($suggestions  as $suggestion) {
         echo "Possible  spelling:  $suggestion<br  />¥n";
    }
}

?>
```

目次

BBCode 関数

導入

この拡張モジュールは、BBCode のテキストをパースして HTML やその他のマークアップ言語への変換を支援するためのものです。 正規表現を用いた一般的な手法に比べてはるかに高速にパースすることができます。 さらに、開始/終了タグの並べ替えをしたり 閉じ忘れているタグに自動的に終了タグを追加したりして、 正しい形式の HTML を生成するようにします。

0.10.1 以降では、シングルクォートやダブルクォート あるいは HTML エスケープしたダブルクォートによる 引数のクォートをサポートしています。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 »
http://pecl.php.net/package/bbcode

別の方法としては、PHP で書かれた PEAR パッケージ » HTML_BBCodeParser を使用することもできます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

BBCode 拡張モジュールで使用するリソースは、 BBCode_Container です。これは bbcode_create() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使

用可能です。

BBCODE_TYPE_NOARG (integer)
この BBCode タグは、引数を受け取りません。
BBCODE_TYPE_SINGLE (integer)
この BBCode タグには、対応する終了タグはありません。
BBCODE_TYPE_ARG (integer)
この BBCode タグは、引数をひとつ必要とします。
BBCODE_TYPE_OPTARG (integer)
この BBCode タグは、オプションで引数をひとつ受け取ることができます。
BBCODE_TYPE_ROOT (integer)
この BBCode タグは、特別なルートタグ (ネストレベル 0) です。
BBCODE_FLAGS_ARG_PARSING (integer)
この BBCode タグの引数をパースする必要があります (引数自体も BBCode 拡張モジュールでパースします)。 0.10.2 以降では、引数のパーサとして別のパーサを使用することができます。
BBCODE_FLAGS_CDATA_NOT_ALLOWED (integer)
この BBCode タグにはコンテンツを含めることができません (自動的に空となります)。
BBCODE_FLAGS_SMILEYS_ON (integer) - 0.10.2 以降
この BBCode タグは、顔文字を受け付けます。
BBCODE_FLAGS_SMILEYS_OFF (integer) - 0.10.2 以降
この BBCode タグは、顔文字を受け付けません。
BBCODE_FLAGS_ONE_OPEN_PER_LEVEL (integer) - 0.10.2 以降
この BBCode タグは、同じ型のタグが同一ネストレベルに登場したときに自動的に閉じます。
BBCODE_FLAGS_REMOVE_IF_EMPTY (integer) - 0.10.2 以降
この BBCode タグは、中身が空の場合に自動的に削除します。 軽量な HTML を生成することができます。
BBCODE_FLAGS_DENY_REOPEN_CHILD (integer) - 0.10.3 以降
この BBCode タグは、閉じられていない子要素がが自動的に閉じられたときに、 再開させません。
BBCODE_ARG_DOUBLE_QUOTE (integer) - 0.10.2 以降
これはパーサのオプションで、ダブルクォート (") でクォートしたタグを受け付けます。
BBCODE_ARG_SINGLE_QUOTE (integer) - 0.10.2 以降
これはパーサのオプションで、シングルクォート (') でクォートしたタグを受け付けます。
BBCODE_ARG_HTML_QUOTE (integer) - 0.10.2 以降
これはパーサのオプションで、HTML 版のダブルクォート (") でクォートしたタグを受け付けます。
BBCODE_AUTO_CORRECT (integer) - 0.10.2 以降
これはパーサのオプションで、エラー時の対応方法を変更します。 タグをオープンした順に、自動的にタグを閉じます。 また、開始タグしか存在しない場合にも、 終了タグがあるかのように扱います。
BBCODE_CORRECT_REOPEN_TAGS (integer) - 0.10.2 以降
これはパーサのオプションで、エラー時の対応方法を変更します。 終了タグの並び順が間違っている場合に、 自動的にタグを再開します。
BBCODE_DISABLE_TREE_BUILD (integer) - 0.10.2 以降
これはパーサのオプションで、BBCode のパースを無効にします。 これは、単に "顔文字" の変換機能だけを使いたい場合に便利です。 be used.
BBCODE_DEFAULT_SMILEYS_ON (integer) - 0.10.2 以降
これはパーサのオプションで、 そのタグレベルで何もフラグが設定されていない場合に 顔文字を ON にします。
BBCODE_DEFAULT_SMILEYS_OFF (integer) - 0.10.2 以降
これはパーサのオプションで、 そのタグレベルで何もフラグが設定されていない場合に 顔文字を OFF にします。
BBCODE_FORCE_SMILEYS_OFF (integer) - 0.10.2 以降
これはパーサのオプションで、 顔文字のパースを完全に無効にします。
BBCODE_SMILEYS_CASE_INSENSITIVE (integer) - 0.10.3 以降
単純なバイナリサーチではなく、 大文字小文字を区別せずに顔文字を検出します。
BBCODE_SET_FLAGS_SET (integer) - 0.10.2 以降
これは、完全なフラグをパーサに設定できるようにします。
BBCODE_SET_FLAGS_ADD (integer) - 0.10.2 以降
これは、パーサのフラグを ON に切り替えられるようにします。
BBCODE_SET_FLAGS_REMOVE (integer) - 0.10.2 以降
これは、パーサのフラグを OFF に切り替えられるようにします。

---

bbcode_add_element

(PECL bbcode:0.9.0-0.9.1)

bbcode_add_element — bbcode 要素を追加する

説明

bool **bbcode_add_element** (resource `$bbcode_container` , string `$tag_name` , array `$tag_rules`)

既存の BBCode_Container tag_set に、tag_rules を使用してタグを追加します。

パラメータ

bbcode_container

bbcode_create() が返す BBCode_Container。

tag_name

BBCode_Container tag_set に追加する新しいタグ。

tag_rules

パース規則を指定する連想配列。使用できるキーは bbcode_create() を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

bbcode_add_smiley

(No version information available, might be only in CVS)

bbcode_add_smiley — 顔文字をパーサに追加する

説明

bool **bbcode_add_smiley** (resource `$bbcode_container` , string `$smiley` , string `$replace_by`)

顔文字をパーサに追加します。

パラメータ

bbcode_container

bbcode_create() が返す BBCode_Container リソース。

smiley

見つかった場合に置換の対象となる文字列。

replace_by

顔文字が見つかった場合にそれを置換する文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bbcode_add_smiley() の使用例

```
<?php
/*
 * ルールを設定します
 */
$arrayBBCode=array(
    ''=>          array('type'=>BBCODE_TYPE_ROOT,
                        'childs'=>'!i'),
    'b'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<b>',
                        'close_tag'=>'</b>'),
    'u'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<u>',
                        'close_tag'=>'</u>',
                        'flags'=>BBCODE_FLAGS_SMILEYS_OFF),
    'i'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<i>',
                        'close_tag'=>'</i>',
                        'childs'=>'b'),
);
/*
 * パースするテキスト
 */
$text=<<<EOF
```

```
[i] No parse Test [/i] :)
[b] Parsed, with smiley :( [/b]
[u] Parsed, with no smiley :D [/u]
EOF;
/*
 * パーサを初期化します
 */
$BBHandler=bbcode_create($arrayBBCode);
/*
 * 顔文字のルールをパーサに追加します
 */
bbcode_add_smiley($BBHandler, ":)", "<img src=¥"smiley.gif¥" alt=¥":)¥" />");
bbcode_add_smiley($BBHandler, ":(", "<img src=¥"sad.gif¥" alt=¥":(¥" />");
bbcode_add_smiley($BBHandler, ":D", "<img src=¥"happy.gif¥" alt=¥":D¥" />");
bbcode_add_smiley($BBHandler, ":p", "<img src=¥"tong.gif¥" alt=¥":p¥" />");
bbcode_add_smiley($BBHandler, ":|", "<img src=¥"special.gif¥" alt=¥":|¥" />");
bbcode_add_smiley($BBHandler, ":6:", "<img src=¥"six.gif¥" alt=¥":6:¥" />");
/*
 * テキストをパースします
 */
echo bbcode_parse($BBHandler,$text);
?>
```

上の例の出力は以下となります。

```
[i] No parse Test [/i] <img src="smiley.gif" alt=":)" />
<b> Parsed, with smiley <img src="sad.gif" alt=":(" /> </b>
<u> Parsed, with no smiley :D </u>
```

---

bbcode_create

(PECL bbcode:0.9.0-0.9.1)

bbcode_create — BBCode リソースを作成する

説明

resource **bbcode_create** ([array `$bbcode_initial_tags`])

この関数は、新しい BBCode リソースを返します。 これを用いて BBCode 文字列をパースします。

パラメータ

`bbcode_initial_tags`

タグ名をキー、BBCode を正しくパースするために必要なパラメータをその値とする連想配列。 以下の キー/値 のペアが使用可能です。

- *flags* (任意) - BBCODE_FLAGS_* 定数を組み合わせたフラグ。
- *type* (必須) - タグの種類を表す整数値。BBCODE_TYPE_* 定数を使用します。
- *open_tag* (必須) - 開始タグ用の HTML 置換文字列。
- *close_tag* (必須) - 終了タグ用の HTML 置換文字列。
- *default_arg* (任意) - tag_type が OPTARG で、 引数が指定されていない場合に使用するデフォルトの引数。
- *content_handling* (任意) - コンテンツの変更時に使用するコールバックの名前を指定します。 オブジェクト指向の記法は 0.10.1 以降でしかサポートしていません。 コールバック関数のプロトタイプは string name(string $content, string $argument) となります。
- *param_handling* (任意) - 引数の変更時に使用するコールバックの名前を指定します。 オブジェクト指向の記法は 0.10.1 以降でしかサポートしていません。 コールバック関数のプロトタイプは string name(string $content, string $argument) となります。
- *childs* (任意) - このタグの子として使用できるタグのリスト。 カンマ区切りの文字列で指定します。 先頭が！の場合は、指定したタグ以外のすべての子を許可します。
- *parent* (任意) - このタグの子として使用できるタグのリスト。 カンマ区切りの文字列で指定します。

返り値

BBCode_Container を返します。

例

Example#1 bbcode_create() の例

```
<?php
$arrayBBCode=array(
    ''=>          array('type'=>BBCODE_TYPE_ROOT,  'childs'=>'!i'),
    'i'=>         array('type'=>BBCODE_TYPE_NOARG, 'open_tag'=>'<i>',
                    'close_tag'=>'</i>', 'childs'=>'b'),
    'url'=>       array('type'=>BBCODE_TYPE_OPTARG,
                    'open_tag'=>'<a  href="{PARAM}">', 'close_tag'=>'</a>',
                    'default_arg'=>'{CONTENT}',
```

```
                    'childs'=>'b,i'),
    'img'=>         array('type'=>BBCODE_TYPE_NOARG,
                    'open_tag'=>'<img  src="', 'close_tag'=>'"  />',
                    'childs'=>''),
    'b'=>           array('type'=>BBCODE_TYPE_NOARG, 'open_tag'=>'<b>',
                    'close_tag'=>'</b>'),
);
$text=<<<EOF
[b]太字のテキスト[/b]
[i]斜体のテキスト[/i]
[url]http://www.php.net/[/url]
[url=http://pecl.php.net/][b]PECL  のサイト[/b][/url]
[img]http://static.php.net/www.php.net/images/php.gif[/img]
[url=http://www.php.net/]
[img]http://static.php.net/www.php.net/images/php.gif[/img]
[/url]
EOF;
$BBHandler=bbcode_create($arrayBBCode);
echo bbcode_parse($BBHandler,$text);
?>
```

上の例の出力は以下となります。

```
<b>太字のテキスト</b>
<i>斜体のテキスト</i>
<a  href="http://www.php.net/">http://www.php.net/</a>
<a  href="http://pecl.php.net/"><b>PECL  のサイト</b></a>
<img  src="http://static.php.net/www.php.net/images/php.gif"  />
<a  href="http://www.php.net/">
[img]http://static.php.net/www.php.net/images/php.gif[/img]
</a>
```

---

bbcode_destroy

(PECL bbcode:0.9.0-0.9.1)

bbcode_destroy — BBCode_container リソースを閉じる

説明

bool **bbcode_destroy** (resource `$bbcode_container`)

この関数は、bbcode_create() がオープンしたリソースを閉じます。

パラメータ

bbcode

bbcode_create() が返す BBCode_Container。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

bbcode_parse

(PECL bbcode:0.9.0-0.9.1)

bbcode_parse — 文字列を、指定した規則のもとでパースする

説明

string **bbcode_parse** (resource `$bbcode_container` , string `$to_parse`)

この関数は、bbcode_create() で作成した bbcode_container の規則にもとづいて文字列 to_parse をパースします。

パラメータ

bbcode_container

bbcode_create() が返す BBCode_Container。

to_parse

パースしたい文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

bbcode_set_arg_parser

(No version information available, might be only in CVS)

bbcode_set_arg_parser — 別のパーサをアタッチして、引数のパース用に別のルールセットを設定する

説明

bool **bbcode_set_arg_parser** (resource `$bbcode_container` , resource `$bbcode_arg_parser`)

別のパーサを bbcode_container にアタッチします。 このパーサは、引数をパースする必要が生じた際にのみ用いられます。 この関数を使用しない場合、 デフォルトの引数パーサとして用いられるのはそのパーサ自身です。

パラメータ

bbcode_container

bbcode_create() が返す BBCode_Container リソース。

bbcode_arg_parser

bbcode_create() が返す BBCode_Container リソース。 これは引数のパースにのみ使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bbcode_set_arg_parser() の使用例

```
<?php
/*
 * メインパーサ用の bbcode ルールセットを生成します
 */
$arrayBBCode=array(
    'quote'=>     array('type'=>BBCODE_TYPE_ARG,
                        'open_tag'=>'<quote><h4>Source:  {PARAM}</h4>',
                        'close_tag'=>'</quote>',
                        'flags'=>BBCODE_FLAGS_REMOVE_IF_EMPTY|BBCODE_FLAGS_ARG_PARSING),
    'b'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<b>', 'close_tag'=>'</b>',
                        'flags'=>BBCODE_FLAGS_REMOVE_IF_EMPTY),
    'u'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<u>', 'close_tag'=>'</u>',
                        'flags'=>BBCODE_FLAGS_SMILEYS_OFF  | BBCODE_FLAGS_REMOVE_IF_EMPTY  | BBCODE_FLAGS_SMILEYS_OFF),
    'i'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<i>', 'close_tag'=>'</i>',
                        'flags'=>BBCODE_FLAGS_REMOVE_IF_EMPTY),
);
/*
 * 引数パーサ用の bbcode ルールセットを生成します
 */
$arrayBBCode_arg=array(
    'b'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<b  class="sub">', 'close_tag'=>'</b>',
                        'flags'=>BBCODE_FLAGS_REMOVE_IF_EMPTY),
    'u'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<u>', 'close_tag'=>'</u>',
                        'flags'=>BBCODE_FLAGS_SMILEYS_OFF  | BBCODE_FLAGS_REMOVE_IF_EMPTY  | BBCODE_FLAGS_SMILEYS_OFF),
    'i'=>         array('type'=>BBCODE_TYPE_NOARG,
                        'open_tag'=>'<i>', 'close_tag'=>'</i>',
                        'flags'=>BBCODE_FLAGS_REMOVE_IF_EMPTY),
);
/*
 * パースするテキスト
 */
$text=<<<EOF
[quote="[b]Test[/b]"]
Foo :)
[/quote]
[b]Bar example :)[/b] :)
EOF;
/*
 * ふたつのパーサを初期化します
 */
$BBHandler=bbcode_create($arrayBBCode);
```

```
$BBArgHandler=bbcode_create($arrayBBCode_arg);
/*
 * パーサのフラグを設定します
 */
bbcode_set_flags($BBHandler,
                 BBCODE_CORRECT_REOPEN_TAGS|BBCODE_DEFAULT_SMILEYS_ON|BBCODE_ARG_DOUBLE_QUOTE|
                 BBCODE_ARG_SINGLE_QUOTE|BBCODE_ARG_HTML_QUOTE,BBCODE_SET_FLAGS_SET);
bbcode_set_flags($BBArgHandler,
                 BBCODE_CORRECT_REOPEN_TAGS|BBCODE_DEFAULT_SMILEYS_ON|BBCODE_ARG_DOUBLE_QUOTE|
                 BBCODE_ARG_SINGLE_QUOTE|BBCODE_ARG_HTML_QUOTE,BBCODE_SET_FLAGS_SET);
/*
 * $BBArgHandler を BBHandler の引数パーサに設定します
 */
bbcode_set_arg_parser($BBHandler,$BBArgHandler);
/*
 * 顔文字の処理ルールをメインパーサに追加します
 */
bbcode_add_smiley($BBHandler, ":)", "<img src=¥"smiley.gif¥" alt=¥":)¥" />");
/*
 * メインパーサでテキストをパースします
 */
echo bbcode_parse($BBHandler,$text);
?>
```

上の例の出力は以下となります。

```
<quote><h4>Source:  <b  class="sub">Test</b></h4>
Foo  <img  src="smiley.gif"  alt=":)"  />
</quote>
<b>Bar  example  :)</b>  <img  src="smiley.gif"  alt=":)"  />
```

---

bbcode_set_flags

(No version information available, might be only in CVS)

bbcode_set_flags — パーサのオプションを設定あるいは変更する

説明

bool **bbcode_set_flags** (resource `$bbcode_container` , int `$flags` [, int `$mode`])

パーサのオプションを設定あるいは変更します。

パラメータ

`bbcode_container`

bbcode_create() が返す BBCode_Container リソース。

`flags`

bbcode_container オプションに適用するフラグのセット。

`mode`

BBCODE_SET_FLAGS_* 定数のいずれか。 指定したフラグを設定する、あるいは解除する、あるいは置き換えるのいずれかを指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bbcode_set_flags() の使用例

```
<?php
/*
 * ルールセットを準備します
 */
$arrayBBCode=array(
    'b'=>          array('type'=>BBCODE_TYPE_NOARG,
                         'open_tag'=>'<b>', 'close_tag'=>'</b>'),
    'u'=>          array('type'=>BBCODE_TYPE_NOARG,
                         'open_tag'=>'<u>', 'close_tag'=>'</u>'),
    'i'=>          array('type'=>BBCODE_TYPE_NOARG,
                         'open_tag'=>'<i>', 'close_tag'=>'</i>'),
);
/*
 * 入れ子構造が間違っている BBCode
 */
$text="[i] Parser [b] Auto Correction [/i] at work [/b]¥n";
```

```
$BBHandler=bbcode_create($arrayBBCode);
echo bbcode_parse($BBHandler,$text);
// 閉じた要素を自動的に再開させます
bbcode_set_flags($BBHandler,BBCODE_CORRECT_REOPEN_TAGS,
                 BBCODE_SET_FLAGS_SET);
echo bbcode_parse($BBHandler,$text);

/*
 * 入れ子構造が間違っており、閉じタグが足りない BBCode
 */
$text="[i] Parser [b] Auto Correction [/i] at work¥n";
echo bbcode_parse($BBHandler,$text);
// 閉じていないタグを自動的に終了させます
bbcode_set_flags($BBHandler,
                 BBCODE_CORRECT_REOPEN_TAGS|BBCODE_AUTO_CORRECT,
                 BBCODE_SET_FLAGS_SET);
echo bbcode_parse($BBHandler,$text);
?>
```

上の例の出力は以下となります。

```
<i> Parser <b> Auto Correction </b></i> at work
<i> Parser <b> Auto Correction </b></i><b> at work </b>
<i> Parser [b] Auto Correction </i> at work
<i> Parser <b> Auto Correction </b></i><b> at work
</b>
```

目次

BCMath任意精度数学関数

導入

任意精度演算に関して、PHPは文字列として表された任意の大きさおよび 精度の数をサポートするバイナリ計算機を提供します。

要件

PHP 4.0.4以降、libbcmathがPHPに付属しています。このモジュールを使 用するために外部のライブラリを使用する必要はありません。

インストール手順

これらの関数は、PHPが構築オプション --enable-bcmathを付けてコンパイ ルされている場合にのみ使用できます。 PHP 3では、これらの関数は、PHPが構築オプション --disable-bcmath を付けずにコンパイルされている場合にのみ使用できます。

Windows 版の *PHP* にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

php.ini の設定により動作が変化します。

BC 数学関数設定オプション

名前	デフォルト	変更の可否	変更履歴
bcmath.scale	"0"	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

bcmath.scale integer

全ての bcmath 関数に関する 10 進桁数。 bcscale() も参照してください。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

bcadd

(PHP 4, PHP 5)

bcadd — 2つの任意精度の数値を加算する

説明

string **bcadd** (string $left_operand , string $right_operand [, int $scale])

left_operand を *right_operand* に加算します。

パラメータ

left_operand

左オペランドを表す文字列。

right_operand

右オペランドを表す文字列。

scale

このオプションパラメータを使用して、結果の小数点以下の桁数を指定します。すべての関数で使用するデフォルトのスケールを定義するには bcscale() を使用します。

返り値

二つの数の和を文字列で返します。

例

Example#1 bcadd() の例

```
<?php

$a = '1.234';
$b = '5';

echo bcadd($a, $b);      // 6
echo bcadd($a, $b, 4);   // 6.2340

?>
```

参考

- bcsub()

---

bccomp

(PHP 4, PHP 5)

bccomp — 2 つの任意精度数値を比較する

説明

int **bccomp** (string $left_operand , string $right_operand [, int $scale])

left_operand と *right_operand* を比較し、 結果を整数値で返します。

パラメータ

left_operand

左オペランドを表す文字列。

right_operand

右オペランドを表す文字列。

scale

オプションの *scale* パラメータで、 小数点以下の桁数を指定します。ここまでを使用して比較を行います。

返り値

ふたつのオペランドが等しければ 0、 *left_operand* が *right_operand* より大きければ返り値は 1、小さければ -1 を返します。

例

Example#1 bccomp() の例

```
<?php

echo bccomp('1', '2') . "¥n";   // -1
echo bccomp('1.00001', '1', 3); // 0
echo bccomp('1.00001', '1', 5); // 1

?>
```

---

bcdiv

(PHP 4, PHP 5)

bcdiv — 2つの任意精度数値で除算を行う

説明

string **bcdiv** (string $left_operand , string $right_operand [, int $scale])

left_operand を *right_operand* で除算します。

パラメータ

left_operand

左オペランドを表す文字列。

right_operand

右オペランドを表す文字列。

scale

このオプションパラメータを使用して、結果の小数点以下の桁数を指定します。すべての関数で使用するデフォルトのスケールを定義するには bcscale() を使用します。

返り値

除算結果を文字列で返します。 *right_operand* が 0 の場合は **NULL** を返します。

例

Example#1 bcdiv() の例

```
<?php

echo bcdiv('105', '6.55957', 3);  // 16.007

?>
```

参考

- bcmul()

---

bcmod

(PHP 4, PHP 5)

bcmod — 2 つの任意精度数値の剰余を取得する

説明

string **bcmod** (string `$left_operand` , string `$modulus`)

left_operand の、 *modulus* を法とする剰余を取得します。

パラメータ

left_operand

左オペランドを表す文字列。

modulus

法を表す文字列。

返り値

剰余を文字列で返します。 *modulus* が 0 の場合は **NULL** を返します。

例

Example#1 bcmod() の例

```
<?php
echo bcmod('4', '2'); // 0
echo bcmod('2', '4'); // 2
?>
```

参考

- bcdiv()

---

bcmul

(PHP 4, PHP 5)

bcmul — 2つの任意精度数値の乗算を行う

説明

string **bcmul** (string `$left_operand` , string `$right_operand` [, int `$scale`])

left_operand に *right_operand* を掛けます。

パラメータ

left_operand

左オペランドを表す文字列。

right_operand

右オペランドを表す文字列。

scale

このオプションパラメータを使用して、結果の小数点以下の桁数を指定します。すべての関数で使用するデフォルトのスケールを定義するには bcscale() を使用します。

返り値

結果を文字列で返します。

例

Example#1 bcmul() の例

```
<?php
echo bcmul('1.34747474747', '35', 3); //  47.161
echo bcmul('2', '4'); // 8
?>
```

参考

- bcdiv()

---

bcpow

(PHP 4, PHP 5)

bcpow — 任意精度数値をべき乗する

説明

string **bcpow** (string $left_operand , string $right_operand [, int $scale])

left_operand の *right_operand* 乗を求めます。

パラメータ

left_operand

左オペランドを表す文字列。

right_operand

右オペランドを表す文字列。

scale

このオプションパラメータを使用して、結果の小数点以下の桁数を指定します。すべての関数で使用するデフォルトのスケールを定義するには bcscale() を使用します。

返り値

結果を文字列で返します。

例

Example#1 bcpow() の例

```
<?php
echo bcpow('4.2', '3', 2); //  74.08
?>
```

参考

- bcpowmod()
- bcsqrt()

---

bcpowmod

(PHP 5)

bcpowmod — 任意精度数値のべき乗の、指定した数値による剰余

説明

string **bcpowmod** (string $left_operand , string $right_operand , string $modulus [, int $scale])

modulus で割った余りを求めることを考慮して、 *left_operand* の *right_operand* 乗を高速に計算します。

パラメータ

left_operand

左オペランドを表す文字列。

right_operand

右オペランドを表す文字列。

modulus

法を表す文字列。

scale

このオプションパラメータを使用して、結果の小数点以下の桁数を指定します。すべての関数で使用するデフォルトのスケールを定義するには bcscale() を使用します。

返り値

結果を文字列で返します。*modulus* が 0 の場合は **NULL** を返します。

注意

注意: このメソッドでは剰余計算を行っているので、自然数ではない数を 指定すると予期せぬ結果となります。自然数とは 0 以外の正の整数です。

例

以下の 2 つの文は機能的に同じです。しかし **bcpowmod()** バージョンのほうが実行時間が早いうえ、 より大きな値の計算が可能です。

```
<?php
$a = bcpowmod($x, $y, $mod);

$b = bcmod(bcpow($x, $y), $mod);

// $a と $b は同じ値になります

?>
```

参考

- bcpow()
- bcmod()

---

bcscale

(PHP 4, PHP 5)

bcscale — すべての BC 演算関数におけるデフォルトのスケールを設定する

説明

bool **bcscale** (int $scale)

デフォルトのスケールを設定します。これ以降、 BC 演算関数で明示的にスケールを指定しなかった場合にこの値を使用します。

パラメータ

scale

スケール。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcscale() の例

```
<?php

// デフォルトのスケールを 3 とします
bcscale(3);
echo bcdiv('105', '6.55957'); //  16.007

// これは、bcscale() を使用せずにおなじ結果を得ます
echo bcdiv('105', '6.55957', 3); //  16.007

?>
```

---

bcsqrt

(PHP 4, PHP 5)

bcsqrt — 任意精度数値の平方根を取得する

説明

string **bcsqrt** (string $operand [, int $scale])

operand の平方根を返します。

パラメータ

operand

オペランドを表す文字列。

scale

このオプションパラメータを使用して、結果の小数点以下の桁数を指定します。すべての関数で使用するデフォルトのスケールを定義するには bcscale() を使用します。

返り値

平方根を文字列で返します。 *operand* が負の場合は **NULL** を返します。

例

Example#1 bcsqrt() の例

```
<?php

echo bcsqrt('2', 3); //  1.414

?>
```

参考

- bcpow()

---

bcsub

(PHP 4, PHP 5)

bcsub — 任意精度数値の減算を行う

説明

string **bcsub** (string $left_operand , string $right_operand [, int $scale])

left_operand から *right_operand* を引きます。

パラメータ

left_operand

左オペランドを表す文字列。

right_operand

右オペランドを表す文字列。

scale

このオプションパラメータを使用して、結果の小数点以下の桁数を指定します。すべての関数で使用するデフォルトのスケールを定義するには bcscale() を使用します。

返り値

減算の結果を文字列で返します。

例

Example#1 bcsub() の例

```
<?php

$a = '1.234';
$b = '5';

echo bcsub($a, $b);      // -3
echo bcsub($a, $b, 4);   // -3.7660

?>
```

参考

- bcadd()

目次

PHP バイトコードコンパイラ (bcompiler)

導入

警告

この拡張モジュールは、 *実験的* なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

Bcompiler は、以下のような目的で作成されました。

- プロプライエタリな PHP アプリケーションのスクリプト全体を エンコードするため
- プロプライエタリな PHP アプリケーションの一部のクラスや関数を エンコードするため
- クライアントのデスクトップで動作する php-gtk アプリケーションを、 php.exe を用いずに実行させるため
- PHP から C へのコンバータの実現可能性を調べるため

最初の目的は bcompiler_write_header()、 bcompiler_write_file() および bcompiler_write_footer() を使用することで 実現できます。バイトコードのファイルが、圧縮されていないか あるいはプレーンな形式で書き出されます。 出来上がったバイトコードは、単純に include や require を用いることで使用可能です。

2 番目の目的を実現するには bcompiler_write_header()、 bcompiler_write_class()、 bcompiler_write_footer()、 bcompiler_read() および bcompiler_load() 関数を使用します。 バイトコードのファイルが、圧縮されていないか あるいはプレーンな形式で書き出されます。 bcompiler_load() は、bzip で圧縮された バイトコードファイルを読み込みます。これは元のファイルに比べて 1/3 程度の大きさになります。

EXE 形式のファイルを作成するには、修正された sapi ファイルか 共有ライブラリとしてコンパイルされた PHP とともに bcompiler を使用する必要

があります。この場合、bcompiler は 圧縮されたバイトコードを exe ファイルの後ろから読み込みます。

圧縮しないバイトコードのみで使用した場合、bcompiler によって 処理速度を約 30% 向上させることが可能です。しかし、圧縮しない バイトコードは元のソースコードに比べて 5 倍程度の大きさに なることに気をつけてください。バイトコードを圧縮することで 容量を節約することが可能ですが、圧縮ファイルを展開するのには ソースコードをパースするよりはるかに長い時間がかかります。 bcompiler はバイトコードに対する最適化を一切行いません。 これは将来のバージョンで対応する予定です…。

コードの保護という点に関しては、もとのソースコードやコメントを 復元することは不可能であると考えて間違いありません。bcompiler のバイトコードをもとにしてコードを復元し、クラスに手を加える といったことは無意味です。しかし、bcompiler で作成した バイトコードファイルからデータを取り出すことは可能です。 個人的なパスワードなどの情報をバイトコードの中に含めないでください。

インストール手順

簡単なインストール手順

- 圧縮機能を動作させるには、少なくとも PHP 4.3.0 が必要です。
- PHP 4.3.0 以降のバージョンに Unix コマンドプロンプト上で インストールするには、**pear install bcompiler** とタイプします。
- Windows 上でインストールする場合、バイナリパッケージを配布する 仕組みが整うまでは、pear-general メーリングリストのアーカイブからビルド済みパッケージを探してください (あるいは、もし見つからなければ メーリングリストにメールを送ってください)。
- 古いバージョンにインストールするには、ビルドするために 多少手を加える必要があります。
- *bcompiler.tgz* アーカイブを *php4/ext* に展開します (これは、PECL » http://pecl.php.net/get/bcompiler から取得可能です)。
- 新しく作成されたディレクトリが bcompiler-0.x のような名前に なっていれば、それを bcompiler という名前に変更します (単に PHP モジュールのみをビルドしたいのであれば、これは不要です)。
- PHP 4.3.0 より前のバージョンを使用している場合は、 *Makefile.in.old* を *Makefile.in* に、そして *config.m4.old* を *config.m4* にそれぞれコピーする必要があります。
- *ext/bcompiler* で **phpize** を実行します。
- *php4* で **./buildconf** を実行します。
- --enable-bcompiler (およびその他のオプション) を指定して **configure** を実行します。
- **make; make install**
- これで終わりです。

連絡先

コメント・バグ修正・機能拡張の提案や、 開発を手伝ってくださるという方は、» alan_k@php.net までメールをください。お待ちしています。

---

bcompiler_load_exe

(PECL bcompiler:0.4-0.8)

bcompiler_load_exe — bcompiler の exe ファイルを読み込み、クラスを生成する

説明

bool **bcompiler_load_exe** (string `$filename`)

bcompiler の exe ファイルからデータを読み込み、バイトコードからクラスを生成します。

パラメータ

`filename`

exe ファイルのパスを表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_load() の例

```
<?php

bcompiler_load_exe("/tmp/example.exe");
print_r(get_defined_classes());

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_load()

---

bcompiler_load

(PECL bcompiler:0.4-0.8)

bcompiler_load — bz 圧縮されたファイルを読み込み、クラスを生成する

説明

bool **bcompiler_load** (string `$filename`)

bzip 圧縮されたファイルを読み込み、バイトコードからクラスを生成します。

パラメータ

`filename`

bzcompress されたファイルのパスを表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_load() の例

```
<?php
bcompiler_load("/tmp/example");
print_r(get_defined_classes());
?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

> **注意**: バイトコードをパースするには、include 文や require 文を 使用してください。この関数を使用するよりも、そのほうが より移植性が高く便利です。
> この関数は、バイトコードファイルに含まれるスクリプト本体のコードを 実行するわけではないことに注意しましょう。

参考

- bcompiler_load_exe()

---

bcompiler_parse_class

(PECL bcompiler:0.4-0.8)

bcompiler_parse_class — クラスのバイトコードを読み込み、ユーザ関数をコールする

説明

bool **bcompiler_parse_class** (string `$class` , string `$callback`)

クラスのバイトコードを読み込み、ユーザ関数をコールします。

パラメータ

`class`

クラス名を表す文字列。

`callback`

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_parse_class() の例

```
<?php

function readByteCodes($data) {
    print_r($data);
}

bcompiler_parse_class("DB","readByteCodes");

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

注意: bcompiler 0.5 以降ではこの関数は削除されており、 もはや使用することはできません。

---

bcompiler_read

(PECL bcompiler:0.4-0.8)

bcompiler_read — ファイルハンドルを読み込み、クラスを生成する

説明

bool **bcompiler_read** (resource `$filehandle`)

開いているファイルハンドルからデータを読み込み、バイトコードからクラスを生成します。

パラメータ

`filehandle`

fopen() が返すファイルハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_read() の例

```
<?php
$fh = fopen("/tmp/example","r");
bcompiler_read($fh);
fclose($fh);
print_r(get_defined_classes());

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

> **注意**: バイトコードをパースするには、include 文や require 文を 使用してください。この関数を使用するよりも、そのほうが より移植性が高く便利です。
> この関数は、バイトコードファイルに含まれるスクリプト本体のコードを 実行するわけではないことに注意しましょう。

---

bcompiler_write_class

(PECL bcompiler:0.4-0.8)

bcompiler_write_class — 定義したクラスをバイトコードとして書き込む

説明

bool **bcompiler_write_class** (resource `$filehandle` , string `$className` [, string `$extends`])

この関数は、PHP から既存のクラスをバイトコードとして読み込み、 開かれているファイルハンドルに書き込みます。

パラメータ

`filehandle`

> fopen() が返すファイルハンドル。

`className`

> クラス名を表す文字列。

`extends`

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_class() example

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_class($fh,"DB");
// DB_mysql が DB_common を継承しているのなら、
// DB_mysql より前に DB_common を書き込む必要があります。
bcompiler_write_class($fh,"DB_common");
bcompiler_write_class($fh,"DB_mysql");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

> **注意**: この関数は依存性のチェックを行わないので、 読み込んだ際に *undefined class* とならないように、書き込む順序に注意が必要です。

参考

- bcompiler_write_header()
- bcompiler_write_footer()

---

bcompiler_write_constant

(PECL bcompiler:0.5-0.8)

bcompiler_write_constant — 定義した定数をバイトコードとして書き込む

説明

bool **bcompiler_write_constant** (resource `$filehandle` , string `$constantName`)

この関数は、PHP から既存の定数をバイトコードとして読み込み、 開かれているファイルハンドルに書き込みます。

パラメータ

`filehandle`

fopen() が返すファイルハンドル。

`constantName`

定義済みの定数名を表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_constant() の例

```
<?php
define("MODULE_MAX", 30);

$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_constant($fh,"MODULE_MAX");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_write_header()
- bcompiler_write_footer()

---

bcompiler_write_exe_footer

(PECL bcompiler:0.4-0.8)

bcompiler_write_exe_footer — 開始位置および exe 形式ファイルのフッタを書き込む

説明

bool **bcompiler_write_exe_footer** (resource `$filehandle` , int `$startpos`)

EXE (あるいは実行可能) ファイルは、三つの部分からできています。

- PHP インタプリタ・bcompiler 拡張モジュール・保存されたバイトコードを読み込み、 指定した関数をコールためのスタブ (実行可能なコード。たとえばコンパイル済の C プログラムなど)
- バイトコード (この場合は圧縮していないもののみが対象となります)
- bcompiler の EXE フッタ

bcompiler の CVS 上で examples/embed ディレクトリにある、 php_embed ベースのスタブ phpe.c をコンパイルすることで 適切なスタブが取得できます。

パラメータ

filehandle

fopen() が返すファイルハンドル。

startpos

ファイル内でのバイトコードの開始位置。 ftell($fh) を使用して取得することが可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_footer() の例

```
<?php

/* 出力ファイル (example.exe) を作成します */
$fh = fopen("example.exe", "w");

/* 1) スタブ (phpe.exe) を書き込みます */
$size = filesize("phpe.exe");
$fr = fopen("phpe.exe", "r");
fwrite($fh, fread($fr, $size), $size);
$startpos = ftell($fh);

/* 2) バイトコードを書き込みます */
bcompiler_write_header($fh);
bcompiler_write_class($fh, "myclass");
bcompiler_write_function($fh, "main");
bcompiler_write_footer($fh);

/* 3) EXE フッタを書き込みます */
bcompiler_write_exe_footer($fh, $startpos);

/* 出力ファイルを閉じます */
fclose($fh);
?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_write_header()
- bcompiler_write_class()
- bcompiler_write_footer()

---

bcompiler_write_file

(PECL bcompiler:0.6-0.8)

bcompiler_write_file — php ソースファイルをバイトコードとして書き込む

説明

bool **bcompiler_write_file** (resource `$filehandle` , string `$filename`)

この関数は、指定したソースファイルをバイトコードにコンパイルし、 開かれているファイルハンドルに書き込みます。

パラメータ

filehandle

fopen() が返すファイルハンドル。

filename

ソースファイルのパスを表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_file() の例

```
<?php
$fh = fopen("example.phb", "w");
bcompiler_write_header($fh);
bcompiler_write_file($fh, "example.php");
bcompiler_write_footer($fh);
fclose($fh);
/* 以下はまったく同等となります。
include  "example.php";
   および
include  "example.phb";
*/
?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_write_header()
- bcompiler_write_footer()

---

bcompiler_write_footer

(PECL bcompiler:0.4-0.8)

bcompiler_write_footer — コンパイルされたデータの終了を示す文字 \x00 を書き込む

説明

bool **bcompiler_write_footer** (resource `$filehandle`)

コンパイルされたデータの終了を示す文字 \x00 を書き込みます。

パラメータ

`filehandle`

fopen() が返すファイルハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_footer() の例

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_class($fh,"DB");
bcompiler_write_class($fh,"DB_common");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_write_header()

---

bcompiler_write_function

(PECL bcompiler:0.5-0.8)

bcompiler_write_function — 定義した関数をバイトコードとして書き込む

説明

bool **bcompiler_write_function** (resource `$filehandle` , string `$functionName`)

この関数は、PHP から既存の関数をバイトコードとして読み込み、 開かれているファイルハンドルに書き込みます。書き込む順序を気にする 必要はありません (例えば、関数 b が関数 a を使用している場合に 下の例のようにコンパイルしたとしても正常に動作します)。

パラメータ

`filehandle`

fopen() が返すファイルハンドル。

`functionName`

関数名を表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_function() の例

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_function($fh,"my_function_a");
bcompiler_write_function($fh,"my_function_b");
bcompiler_write_footer($fh);
fclose($fh);

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_write_header()
- bcompiler_write_footer()

---

bcompiler_write_functions_from_file

(PECL bcompiler:0.5-0.8)

bcompiler_write_functions_from_file — ファイル内で定義されているすべての関数をバイトコードとして書き込む

説明

bool **bcompiler_write_functions_from_file** (resource `$filehandle` , string `$fileName`)

指定したファイル内で定義されているすべての関数を検索し、 対応するバイトコードを、 開かれているファイルハンドルに書き込みます。

パラメータ

filehandle

fopen() が返すファイルハンドル。

fileName

コンパイルしたいファイル。 コンパイルしたいファイルを include/require することを忘れないでください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_functions_from_file() の例

```
<?php
require('module.php');

$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_functions_from_file($fh,'module.php');
bcompiler_write_footer($fh);
fclose($fh);

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_write_header()
- bcompiler_write_footer()

---

bcompiler_write_header

(PECL bcompiler:0.3-0.8)

bcompiler_write_header — bcompiler のヘッダを書き込む

説明

bool **bcompiler_write_header** (resource `$filehandle` [, string `$write_ver`])

bcompiler ファイルのヘッダを書き込みます。

パラメータ

filehandle

fopen() が返すファイルハンドル。

write_ver

以前に使われていたフォーマットでバイトコードを書き込む際に使用します。 これにより、古いバージョンの bcompiler でバイトコードを使用することが可能となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 bcompiler_write_header() の例

```
<?php
$fh = fopen("/tmp/example","w");
bcompiler_write_header($fh);
bcompiler_write_class($fh,"DB");
```

```
bcompiler_write_footer($fh);
fclose($fh);

?>
```

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- bcompiler_write_footer()

---

bcompiler_write_included_filename

(PECL bcompiler:0.5-0.8)

bcompiler_write_included_filename — インクルードされたファイルをバイトコードとして書き込む

説明

bool **bcompiler_write_included_filename** (resource `$filehandle` , string `$filename`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

目次

Bzip2 圧縮関数

導入

bzip2関数は、bzip2 (.bz2)圧縮されたファイルを透過的に読み書きする ために使用されます。

要件

このモジュールは、Julian Seward により作成された » bzip2ライブラリの関数を使用して います。このモジュールは、bzip2/libbzip2の バージョン >= 1.0.xを必要とします。

インストール手順

PHP の bzip2 サポートはデフォルトでは有効となっていません。 bzip2 サポートを有効とするには、PHP をコンパイルする際に設定オプション --with-bz2[=DIR] を使用する必要があります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールは、リソース型として処理を行うbz2ファイルを指す ファイルポインタを定義します。

定義済み定数

定数は定義されていません。

例

この例は、テンポラリファイルをオープンし、テスト用の文字列を書き 込みます。この後、このファイルの内容を出力します。

Example#1 簡単な bzip2 の例

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "This is a test string.¥n";

// 書き込み用にオープン
$bz = bzopen($filename, "w");

// ファイルに文字列を書き込む
bzwrite($bz, $str);

// ファイルを閉じる
bzclose($bz);

// 読み込み用にファイルをオープン
$bz = bzopen($filename, "r");

// 10文字読み込む
print bzread($bz, 10);

// ファイルの終端まで出力(または次の1024文字)し、閉じる
print bzread($bz);

bzclose($bz);

?>
```

---

bzclose

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzclose — bzip2 ファイルを閉じる

説明

int **bzclose** (resource $bz)

与えられた bzip2 ファイルポインタを閉じます。

パラメータ

bz

ファイルポインタ。これは有効である必要があり、 bzopen() によりオープンされたファイルを指してい る必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- bzopen()

---

bzcompress

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzcompress — 文字列をbzip2形式のデータに圧縮する

説明

mixed **bzcompress** (string `$source` [, int `$blocksize` [, int `$workfactor`]])

bzcompress() は、与えられた文字列を圧縮し、 bzip2 形式のデータとして返します。

パラメータ

source

圧縮する文字列

blocksize

圧縮時のブロック長を指定します。 1 から 9 の数とする必要があります。この場合、9 の圧縮度が最大ですが、リソースの消費量も最大となります。 *blocksize* のデフォルトは 4 です。

workfactor

繰り返しが多い、最悪の入力データの場合の圧縮処理の動作を制御します。 この値は、0 から 250 までとなり、0 は特別な場合、30 はデフォルト値となります。

workfactor によらず、 生成される出力は同じになります。

返り値

圧縮された文字列、もしくはエラー時はエラー数

例

Example#1 データの圧縮

```
<?php
$str = "sample data";
$bzstr = bzcompress($str, 9);
echo $bzstr;
?>
```

参考

- bzdecompress()

---

bzdecompress

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzdecompress — bzip2 形式のデータを解凍する

説明

mixed **bzdecompress** (string `$source` [, int `$small`])

bzdecompress() は、bzip2 形式のデータを含む文字列を解凍します。

パラメータ

source

解凍する文字列

small

TRUE の場合、よりメモリの消 費量が少ない (所要メモリは最大2300K程度まで少なくなります) 別の圧縮アルゴリズムが使用されますが、速度は約半分になってしまいます。

この機能に関する詳細については、» bzip2 ドキュメント を参照ください。

返り値

解凍された文字列列、もしくはエラー時はエラー数

例

Example#1 文字列を解凍する

```
<?php
$start_str = "This is not an honest face?";
$bzstr = bzcompress($start_str);

echo "Compressed String: ";
echo $bzstr;
echo "¥n<br />¥n";

$str = bzdecompress($bzstr);
echo "Decompressed String: ";
echo $str;
echo "¥n<br />¥n";
?>
```

参考

- bzcompress()

---

bzerrno

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzerrno — bzip2 エラー番号を返す

説明

int **bzerrno** (resource $bz)

与えられたファイルポインタから返された bzip2 エラーのエラー番号を返します。

パラメータ

bz

ファイルポインタ。これは有効である必要があり、 bzopen() によりオープンされたファイルを指してい る必要があります。

返り値

整数としてのエラー番号を返します。

参考

- bzerror()
- bzerrstr()

---

bzerror

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzerror — bzip2 エラー番号とエラー文字列を配列で返す

説明

array **bzerror** (resource $bz)

与えられたファイルポインタから返された bzip2 エラーのエラー番号とエラー文字列を返します。

パラメータ

bz

ファイルポインタ。これは有効である必要があり、 bzopen() によりオープンされたファイルを指してい る必要があります。

返り値

errno エントリにエラーコード、*errstr* エントリにエラーメッセージを持つ連想配列を返します。

例

Example#1 bzerror() の例

```
<?php
$error  = bzerror($bz);

echo $error["errno"];
echo $error["errstr"];
?>
```

参考

- bzerrno()
- bzerrstr()

---

bzerrstr

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzerrstr — bzip2 エラー文字列を返す

説明

string **bzerrstr** (resource $bz)

与えられたファイルポインタから返された bzip2 エラーのエラーの文字列を返します。

パラメータ

bz

ファイルポインタ。これは有効である必要があり、 bzopen() によりオープンされたファイルを指してい る必要があります。

返り値

エラーメッセージを含む文字列を返します。

参考

- bzerrno()
- bzerror()

---

bzflush

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzflush — 全てのバッファリングされたデータを強制的に書き込む

説明

int **bzflush** (resource $bz)

バッファリングされた全ての bzip2 データをファイルポインタ *bz* に書き込みます。

パラメータ

bz

ファイルポインタ。これは有効である必要があり、 bzopen() によりオープンされたファイルを指してい る必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- bzread()
- bzwrite()

---

bzopen

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzopen — bzip2 圧縮されたファイルをオープンする

説明

resource **bzopen** (string `$filename` , string `$mode`)

bzopen() は、 bzip2 (.bz2) ファイルを読み書き用にオープンします。

パラメータ

filename

オープンするファイル名

mode

fopen() 関数 (読み込みは`r'、書き込みは`w'、等) と同様です。

返り値

オープンできなかった場合、**bzopen()** は **FALSE** を返します。それ以外の場合は、 新規にオープンされたファイルへのポインタが返されます。

例

Example#1 bzopen() の例

```
<?php

$file = "/tmp/foo.bz2";
$bz = bzopen($file, "r") or die("Couldn't open $file for reading");

bzclose($bz);

?>
```

参考

- bzclose()

---

bzread

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzread — バイナリ対応の bzip2 ファイル読み込み

説明

string **bzread** (resource `$bz` [, int `$length`])

bzread() は、与えられた bzip2 ファイルポインタから読み込みます。

読み込みは、(圧縮前の状態で) *length* バイトが読み込まれたか、EOF に達したかのどちらか最初に来た方で終了します。

パラメータ

bz

ファイルポインタ。これは有効である必要があり、 bzopen() によりオープンされたファイルを指してい る必要があります。

length

指定されない場合、**bzread()** は一度に (圧縮前の状態で) 1024バイト読み込みます。

返り値

非圧縮データ、もしくはエラー時に **FALSE** を返します。

例

Example#1 bzread() の例

```
<?php

$file = "/tmp/foo.bz2";
$bz = bzopen($file, "r") or die("Couldn't open $file");

$decompressed_file = '';
while (!feof($bz)) {
  $decompressed_file .= bzread($bz, 4096);
}
bzclose($bz);

echo "The contents of $file are: <br />¥n";
echo $decompressed_file;

?>
```

参考

- bzwrite()
- feof()
- bzopen()

---

bzwrite

(PHP 4 >= 4.3.3, PHP 5, PECL bz2:1.0)

bzwrite — バイナリ対応の bzip2 ファイルへの書き込み

説明

int **bzwrite** (resource $bz , string $data [, int $length])

bzwrite() は、文字列を与えられた bzip2 ファイルストリームに書き込みます。

パラメータ

bz

ファイルポインタ。これは有効である必要があり、 bzopen() によりオープンされたファイルを指してい る必要があります。

data

書き込むデータ

length

指定した場合、 (圧縮前の) *length* バイト分の書き込みが終ったか、 *data* の終端に達したかで書き込みは終了します。

返り値

書き込んだバイト数、もしくはエラー時に **FALSE** を返します。

例

Example#1 bzwrite() の例

```
<?php
```

```
$str = "uncompressed  data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

参考

- bzread()
- bzopen()

目次

カレンダー関数

導入

カレンダー関数は、異なったカレンダーフォーマット間の変換を 簡単に行う関数の集まりです。標準としているのは、ユリウス積算日です。 ユリウス積算日は、紀元前 4713 年 1 月 1 日から数え始めわれています。 カレンダーシステム間の変換を行うには、ユリウス積算日に変換した後 に選択したカレンダーシステムに変換しなければなりません。 ユリウス積算日はユリウス暦とは全く違います! ユリウス積算日の詳細を知りたい場合は、 » http://www.hermetic.ch/cal_stud/jdn.htm を参照ください。 カレンダーシステムに関する情報を知りたい場合は、 » http://www.fourmilab.ch/documents/calendar/ を参照ください。 本説明にはこのページからの引用が含まれています。 このページの内容はこのチュートリアルにも反映/引用されています。

インストール手順

これらの関数を動作させるには、 --enable-calendar を指定して PHP をコンパイルする必要があります。

Windows 版の *PHP* にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

CAL_GREGORIAN (integer)
CAL_JULIAN (integer)
CAL_JEWISH (integer)
CAL_FRENCH (integer)
CAL_NUM_CALS (integer)
CAL_DOW_DAYNO (integer)
CAL_DOW_SHORT (integer)
CAL_DOW_LONG (integer)

CAL_MONTH_GREGORIAN_SHORT (integer)
CAL_MONTH_GREGORIAN_LONG (integer)
CAL_MONTH_JULIAN_SHORT (integer)
CAL_MONTH_JULIAN_LONG (integer)
CAL_MONTH_JEWISH (integer)
CAL_MONTH_FRENCH (integer)

以下の定数は、PHP 4.3.0 以降で使用可能です。

CAL_EASTER_DEFAULT (integer)
CAL_EASTER_ROMAN (integer)
CAL_EASTER_ALWAYS_GREGORIAN (integer)
CAL_EASTER_ALWAYS_JULIAN (integer)

以下の定数は、PHP 5.0.0 以降で使用可能です。

CAL_JEWISH_ADD_ALAFIM_GERESH (integer)
CAL_JEWISH_ADD_ALAFIM (integer)
CAL_JEWISH_ADD_GERESHAYIM (integer)

---

cal_days_in_month

(PHP 4 >= 4.0.7, PHP 5)

cal_days_in_month — 指定した年とカレンダーについて、月の日数を返す

説明

int **cal_days_in_month** (int `$calendar` , int `$month` , int `$year`)

この関数は、指定した *calendar* について *year* 年 *month* 月の日数を返します。

パラメータ

calendar

計算に使用するカレンダー。

month

選択したカレンダーにおける月。

year

選択したカレンダーにおける年。

返り値

指定したカレンダーの、その月の日数を返します。

例

Example#1 cal_days_in_month() の例

```
<?php
$num = cal_days_in_month(CAL_GREGORIAN, 8, 2003); // 31
echo "2003 年 8 月の日数は $num 日です";
?>
```

---

cal_from_jd

(PHP 4 >= 4.0.7, PHP 5)

cal_from_jd — ユリウス積算日からサポートされるカレンダーに変換する

説明

array **cal_from_jd** (int `$jd` , int `$calendar`)

cal_from_jd() は、 *jd* で指定したユリウス日を 指定した *calendar* に変換します。 サポートされる *calendar* の値は、 **CAL_GREGORIAN**、 **CAL_JULIAN**、 **CAL_JEWISH** および **CAL_FRENCH** です。

パラメータ

jd

ユリウス日を表す整数値。

calendar

変換するカレンダー。

返り値

カレンダーの情報を含む配列を返します。この配列には、 年、月、日、週、曜日名、月名、"月/日/年" 形式の文字列 などが含まれます。

例

Example#1 cal_from_jd() の例

```
<?php
$today  = unixtojd(mktime(0, 0, 0, 8, 16, 2003));
print_r(cal_from_jd($today, CAL_GREGORIAN));
?>
```

上の例の出力は以下となります。

```
Array
(
     [date] => 8/16/2003
     [month] => 8
     [day] => 16
     [year] => 2003
     [dow] => 6
     [abbrevdayname] => Sat
     [dayname] => Saturday
     [abbrevmonth] => Aug
     [monthname] => August
)
```

参考

- cal_to_jd()
- jdtofrench()
- jdtogregorian()
- jdtojewish()
- jdtojulian()
- jdtounix()

---

cal_info

(PHP 4 >= 4.0.7, PHP 5)

cal_info — 特定のカレンダーに関する情報を返す

説明

array **cal_info** ([int $calendar])

cal_info() は、指定した *calendar* についての情報を返します。

カレンダーの情報は配列として返され、その要素は *calname*、*calsymbol*、 *month*、*abbrevmonth* および *maxdaysinmonth* となります。 *calendar* として指定可能なカレンダー名は以下のとおりです。

- 0 あるいは **CAL_GREGORIAN** - グレゴリウス暦
- 1 あるいは **CAL_JULIAN** - ユリウス暦
- 2 あるいは **CAL_JEWISH** - ユダヤ暦
- 3 あるいは **CAL_FRENCH** - フランス革命暦

calendar が指定されなかった場合は、 サポートするすべてのカレンダーの情報を配列で返します。

パラメータ

calendar

情報を返したいカレンダー。指定しなかった場合は、 すべてのカレンダーに関する情報を返します。

返り値

変更履歴

バージョン	説明
5.0 以降	*calendar* パラメータが省略可能となり、 省略した場合は "すべてのカレンダー" についての情報を返すようになりました。

例

Example#1 cal_info() の例

```
<?php
$info = cal_info(0);
print_r($info);
?>
```

上の例の出力は以下となります。

```
Array
(
     [months] => Array
        (
             [1] => January
             [2] => February
             [3] => March
             [4] => April
             [5] => May
             [6] => June
             [7] => July
             [8] => August
             [9] => September
             [10] => October
             [11] => November
             [12] => December
        )

     [abbrevmonths] => Array
        (
             [1] => Jan
             [2] => Feb
             [3] => Mar
             [4] => Apr
             [5] => May
             [6] => Jun
             [7] => Jul
             [8] => Aug
             [9] => Sep
             [10] => Oct
             [11] => Nov
             [12] => Dec
        )

     [maxdaysinmonth] => 31
     [calname] => Gregorian
     [calsymbol] => CAL_GREGORIAN
)
```

---

cal_to_jd

(PHP 4 >= 4.0.7, PHP 5)

cal_to_jd — サポートされるカレンダーからユリウス積算日に変換する

説明

int **cal_to_jd** (int $calendar , int $month , int $day , int $year)

cal_to_jd() は、指定した *calendar* の日付からユリウス積算日を計算します。 サポートされる *calendar* は **CAL_GREGORIAN**、 **CAL_JULIAN**、 **CAL_JEWISH** および **CAL_FRENCH** です。

パラメータ

calendar

変換元のカレンダー。 **CAL_GREGORIAN**、 **CAL_JULIAN**、 **CAL_JEWISH** あるいは **CAL_FRENCH** のいずれか。

month

月を表す数値。有効な範囲は *calendar* に依存します。

day

日を表す数値。有効な範囲は *calendar* に依存します。

year

年を表す数値。有効な範囲は *calendar* に依存します。

返り値

ユリウス積算日を返します。

参考

- cal_from_jd()
- frenchtojd()
- gregoriantojd()
- jewishtojd()
- juliantojd()
- unixtojd()

---

easter_date

(PHP 4, PHP 5)

easter_date — 指定した年における復活祭の真夜中のUnix時を得る

説明

int **easter_date** ([int $year])

指定した年yearにおける復活祭の真夜中のUnix時を返します。

警告

この関数は、年がUnixタイムスタンプの範囲を越える時 (すなわち、1970より前、または2037より後)に警告を発生します。

復活祭の日付は、西暦325年の Nicaea 会議で春分の日の後の 最初の満月の後の日曜日として定められました。 満月とその次の日曜日の日付の計算を簡単にするために 春分の日は常に3月21日になるとして計算されます。 ここで用いるアルゴリズムは、532年頃、Dionysius Exiguus により 導出されたものです。(1753年より前の年に関して)ユリウス暦の元では 月の周期を追うために簡単な19年周期が用いられます。グレゴリウス暦 (1753年以降。この暦は、ClaviusとLiliusにより考案され、 教皇グレゴリウス13世により1582年10月に導入、イギリス及びその植民地に 1752年9月に導入された。) のもとで、二つの補正係数が周期をより正確に作成するために追加されました。

(このコードは、Simon Kershaw, <webmaster at ely.anglican dot org>によるCプログラムに基づくものです。)

パラメータ

year

1970 から 2037 までの年。

返り値

復活祭の日を Unix タイムスタンプで返します。

変更履歴

バージョン	説明
4.3.0 以降	*year* パラメータはオプションとなり 省略された場合には、ローカル時間に基づく現在の年がデフォルトとなります。

例

Example#1 easter_date() の例

```php
<?php

echo date("M-d-Y", easter_date(1999));         //  Apr-04-1999
echo date("M-d-Y", easter_date(2000));         //  Apr-23-2000
echo date("M-d-Y", easter_date(2001));         //  Apr-15-2001

?>
```

参考

- 1970年以前または2037年以降の復活祭の計算に関しては、 easter_days() を参照ください。

参考

- easter_days()

---

easter_days

(PHP 4, PHP 5)

easter_days — 指定した年において、3 月 21 日から復活祭までの日数を得る

説明

int **easter_days** ([int $year [, int $method]])

指定した年 year において、3 月 21 日から復活祭までの日数を返します。 year が指定されない場合、現在の年が仮定されます。

この関数は、Unix 時の範囲外(すなわち 1970 年以前または 2037 年以降)の復活祭を 計算するために easter_date() の代わりに使用することが できます。

復活祭の日付は、西暦 325 年の Nicaea の会議で春分の日以降の 最初の満月の後の日曜日として定義されました。 満月とその次の日曜日の日付の計算を簡単にするために 春分の日は常に 3 月 21 日になるとして計算されます。 ここで用いるアルゴリズムは、532 年頃に Dionysius Exiguus により 導出されたものです。(1753 年より前の年に関して)ユリウス暦のもとでは 月の周期を追うために簡単な 19 年周期が用いられます。グレゴリウス暦 (1753 年以降。この暦は Clavius と Lilius により考案され、 教皇グレゴリウス 13 世により 1582 年 10 月に導入、イギリス及びその植民地に 1752 年 9 月に導入された) のもとで、二つの補正係数が周期をより正確に作成するために追加されました。

(このコードは、Simon Kershaw, <webmaster at ely.anglican dot org> による C プログラムに基づくものです。)

パラメータ

year

1970 から 2037 までの年。

返り値

指定した年 *year* において、3 月 21 日から復活祭までの日数を返します。

変更履歴

バージョン	説明
4.3.0 以降	*year* はオプションのパラメータとなり、 もし指定されなかった場合は、地方時にもとづいた現在の年がデフォルトとなります。
4.3.0 以降	*method* パラメータもまた PHP 4.3.0 以降で登場 したもので、これを **CAL_EASTER_ROMAN** に設定すると 1582 年から 1752 年までの復活祭の日付をグレゴリウス暦にもとづいて 計算します。それ以外に使用可能な定数については カレンダー定数を参照ください。

例

Example#1 easter_days() の例

```php
<?php

echo easter_days(1999);         // 14, i.e. April 4
echo easter_days(1492);         // 32, i.e. April 22
echo easter_days(1913);         //  2, i.e. March 23

?>
```

参考

- easter_date()

FrenchToJD

(PHP 4, PHP 5)

FrenchToJD — フランス革命暦をユリウス積算日に変換する

説明

int **frenchtojd** (int $month , int $day , int $year)

日付けをフランス革命暦からユリウス積算日に変換します。

これらのルーチンは、1から14年まで(グレゴリウス暦の1792年9月22日から 1806年9月22日)日付けのみを変換します。 この期間は、フランス革命暦が使用されていた期間を十分にカバーしています。

パラメータ

month

月を表す、1 (VendĂŠmiaire) から 13 (各年の最後の 5-6 日) までの数字。

day

日を表す、1 から 30 までの数字。

year

年を表す、1 から 14 までの数字。

返り値

フランス革命暦の日付をユリウス積算日になおした結果を整数値で返します。

参考

- jdtofrench()
- cal_to_jd()

GregorianToJD

(PHP 4, PHP 5)

GregorianToJD — グレゴリウス日をユリウス積算日に変換する

説明

int **gregoriantojd** (int $month , int $day , int $year)

有効なグレゴリウス暦の範囲は紀元前 4714 年から紀元 9999 年までです。

このソフトウェアは日付けを全て紀元前 4714 年にさかのぼることが 出来ますが、この様な使い方は特に意味があるわけではありません。 グレゴリウス暦は 1582 年 10 月 15 日(ユリウス暦では 1582 年 10 月 5 日)まで制定されていませんでした。 この暦は、いくつかの国でもっと後まで受け入れられませんでした。 例えば、イギリスは 1752 年、ロシア(USSR)は 1918 年、ギリシャは 1923 年に 移行しました。 ほとんどのヨーロッパの国々では、グレゴリウス暦の前は ユリウス暦を使用していました。

パラメータ

month

月を表す、1 (January) から 12 (December) までの数字。

day

日を表す、1 から 31 までの数字。

year

年を表す、-4714 から 9999 までの数字。

返り値

指定したグレゴリウス日をユリウス積算日になおした結果を返します。

例

Example#1 カレンダー関数

```
<?php
$jd = GregorianToJD(10, 11, 1970);
echo "$jd¥n";
$gregorian  = JDToGregorian($jd);
echo "$gregorian¥n";
?>
```

参考

- jdtogregorian()
- cal_to_jd()

JDDayOfWeek

(PHP 4, PHP 5)

JDDayOfWeek — 曜日を返す

説明

mixed **jddayofweek** (int `$julianday` [, int `$mode`])

曜日を返します。 モードに依存した文字あるいは整数を返す事が出来ます。

パラメータ

julianday

ユリウス積算日を表す整数値。

mode

暦の曜日モード

モード	意味
0 (デフォルト)	整数で曜日番号 (0=Sunday, 1=Monday, 等)を返します。
1	曜日(英グレゴリウス)を含む文字列を返します。
2	曜日の省略形(英グレゴリウス)を含む文字列を返します。

返り値

グレゴリウス暦の曜日を表す数値あるいは文字列を返します。

JDMonthName

(PHP 4, PHP 5)

JDMonthName — 月の名前を返す

説明

string **jdmonthname** (int `$julianday` , int `$mode`)

月の名前を含んだ文字列を返します。 *mode* はユリウス積算日をどの暦に変換するか、 どんなタイプの月名を返すかを関数に伝えます。

暦モード

モード	意味	値
0	グレゴリウス暦 - 省略形	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
1	グレゴリウス暦	January, February, March, April, May, June, July, August, September, October, November, December
2	ユリウス暦 - 省略形	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
3	ユリウス暦	January, February, March, April, May, June, July, August, September, October, November, December
4	ユダヤ暦	Tishri, Heshvan, Kislev, Tevet, Shevat, Adarl, Adarll, Nisan, Iyyar, Sivan, Tammuz, Av, Elul
5	フランス革命暦	Vendemiaire, Brumaire, Frimaire, Nivose, Pluviose, Ventose, Germinal, Floreal, Prairial, Messidor, Thermidor, Fructidor, Extra

パラメータ

jday

変換したいユリウス積算日。

calendar

月名を取得する暦。

返り値

指定したユリウス積算日と *calendar* にもとづく月名を返します。

JDToFrench

(PHP 4, PHP 5)

JDToFrench — ユリウス積算日をフランス革命暦(共和暦)に変換する

説明

string **jdtofrench** (int $juliandaycount)

ユリウス積算日をフランス革命暦に変換します。

パラメータ

julianday

ユリウス積算日を表す整数値。

返り値

フランス革命暦の日付を "月/日/年" 形式の文字列で返します。

参考

- frenchtojd()
- cal_from_jd()

JDToGregorian

(PHP 4, PHP 5)

JDToGregorian — ユリウス積算日をグレゴリウス日に変換する

説明

string **jdtogregorian** (int $julianday)

ユリウス積算日を "月/日/年" の形式でグレゴリウス日を含む文字列に変換します。

パラメータ

julianday

ユリウス積算日を表す整数値。

返り値

グレゴリウス暦の日付を "月/日/年" 形式の文字列で返します。

参考

- gregoriantojd()
- cal_from_jd()

---

jdtojewish

(PHP 4, PHP 5)

jdtojewish — ユリウス積算日をユダヤ暦に変換する

説明

string **jdtojewish** (int `$juliandaycount` [, bool `$hebrew` [, int `$fl`]])

ユリウス積算日をユダヤ暦に変換します。

パラメータ

julianday

ユリウス積算日を表す整数値。

hebrew

パラメータ *hebrew* が **TRUE** に設定されている場合、 ヘブライ語の文字列形式での出力のために *fl* が使用されます。

fl

使用可能なフォーマットは **CAL_JEWISH_ADD_ALAFIM_GERESH**、 **CAL_JEWISH_ADD_ALAFIM** および **CAL_JEWISH_ADD_GERESHAYIM** です。

返り値

ユダヤ暦の日付を "月/日/年" 形式の文字列で返します。

変更履歴

バージョン	説明
5.0.0	*hebrew* および *fl* が追加されました。

例

Example#1 jdtojewish() の例

```
<?php
echo jdtojewish(gregoriantojd(10, 8, 2002), true,
       CAL_JEWISH_ADD_GERESHAYIM  + CAL_JEWISH_ADD_ALAFIM  + CAL_JEWISH_ADD_ALAFIM_GERESH);
?>
```

参考

- jewishtojd()
- cal_from_jd()

JDToJulian

(PHP 4, PHP 5)

JDToJulian — ユリウス積算日をユリウス暦に変換する

説明

string **jdtojulian** (int `$julianday`)

ユリウス積算日を "月/日/年" の形式でユリウス暦を含む文字列に変換します。

パラメータ

`julianday`

ユリウス積算日を表す整数値。

返り値

ユリウス暦の日付を "月/日/年" 形式の文字列で返します。

参考

- juliantojd()
- cal_from_jd()

---

jdtounix

(PHP 4, PHP 5)

jdtounix — ユリウス歴を Unix タイムスタンプに変換する

説明

int **jdtounix** (int `$jday`)

この関数は、*`jday`* で指定したユリウス歴に 対応する Unix タイムスタンプを返します。 *`jday`* が Unix 歴 (グレゴリウス歴で 1970 年と 2037 年の間または 2440588 <= *`jday`* <= 2465342) の範囲外の場合は **FALSE** を返します。返される時刻は地方時間です (GMT ではありません)。

パラメータ

`jday`

2440588 から 2465342 までのユリウス積算日。

返り値

指定したユリウス積算日の開始時の Unix タイムスタンプを返します。

参考

- unixtojd()

JewishToJD

(PHP 4, PHP 5)

JewishToJD — ユダヤ暦の日付けをユリウス積算日に変換する

説明

int **jewishtojd** (int `$month` , int `$day` , int `$year`)

この関数では1年(紀元前3761年)に遡って全ての日々を扱うこと が出来ますが、この様な使い方は特に意味があるわけではありません。 ユダヤ暦は数千年に渡って使われていますが、当初は月の 始まりが決められていませんでいた。新しい月は新月が最初に観測 された日に始まりました。

パラメータ

month

月を表す、1 から 13 までの数値。

day

日を表す、1 から 30 までの数値。

year

年を表す、1 から 9999 までの数値。

返り値

指定したユダヤ暦の日付に対応するユリウス積算日を返します。

参考

- jdtojewish()
- cal_to_jd()

JulianToJD

(PHP 4, PHP 5)

JulianToJD — ユリウス暦をユリウス積算日に変換する

説明

int **juliantojd** (int $month , int $day , int $year)

ユリウス暦は紀元前 4713 年から紀元 9999 年の範囲で使えます。

この関数は日付けを全て紀元前 4713 年にさかのぼって扱うことが 出来ますが,この様な使い方はあまり意味があるわけではありません。 このカレンダーは紀元前 46 年に作られました、しかし細部は紀元 8 年になるまで、おそらくは 4 世紀になるまで安定しませんでした。 また、年の始まりを一つの文化のものから別の文化のものに変更すること、 つまり、January を最初の月とすることは全く受け入れられませんでした。

警告

覚えておいてほしいのは、現在世界中で使用されているカレンダーは グレゴリウス暦であるということです。この日付をユリウス日に 変換するには、gregoriantojd() が使用可能です。

パラメータ

month

月を表す、1 (for January) から 12 (for December) までの数値。

day

日を表す、1 から 31 までの数値。

year

年を表す、-4713 から 9999 までの数値。

返り値

指定したユリウス暦の日付に対応するユリウス積算日を返します。

参考

- jdtojulian()
- cal_to_jd()

---

unixtojd

(PHP 4, PHP 5)

unixtojd — Unix タイムスタンプをユリウス歴に変換する

説明

int **unixtojd** ([int `$timestamp`])

Unix タイムスタンプ(1970/1/1 からの秒数)*`timestamp`* をユリウス歴に変換して返します。*`timestamp`* が 指定されない場合は現在の日付が使用されます。

パラメータ

`timestamp`

変換したい Unix タイムスタンプ。

返り値

ユリウス積算日を表す整数値を返します。

参考

- jdtounix()

目次

CCVS API 関数 [非推奨]

導入

このモジュールの関数は CCVS API へのインターフェースであり、PHP スクリプトから CCVS を直接利用可能にするものです。CCVS は、 » RedHat による、クレジットカード 処理用の「仲介人」ソリューションです。これにより、*nix マシンと モデムによりクレジットカード情報センターに直接アクセスすることが 可能になります。PHP の CCVS モジュールを使用すれば、 PHP スクリプトにより CCVS を用いて直接クレジットカードの処理が可能になります。以下のリファレンスに処理の概要を示します。

> **注意**: CCVS の開発は Red Hat により中断され、開発やサポートが継続される 予定はありません。代替品を探している場合は、 » Main Street Softworks の MCVE を検討してください。 これは、設計が似ており、PHP のサポートが文書でうたわれています!
> この拡張モジュールは PHP 4.3.0 以降では PHP から削除されており、 使用できません。もしクレジットカードの処理機能を使用したい場合は、 かわりに MCVE が使用可能です。

インストール手順

PHP で CCVS サポートを有効にするには、まず CCVS をインストールした ディレクトリを調べてください。続いて、オプション --with-ccvs を付け

て PHP の configure を実行する必要があります。CCVS のインストールパスを指定せずに このオプションを使用した場合、PHP はデフォルトの CCVS インストール位置 (*/usr/local/ccvs*) を探します。CCVS が標準以外の 場所にある場合、--with-ccvs=[DIR] を付けて configure を実行してください。ただし、DIR は CCVS の インストールパスです。CCVS のサポートは、DIR/lib および DIR/include が存在していること・include ディレクトリに *cv_api.h* があること・lib ディレクトリに *libccvs.a* が あることを前提としていることに注意してください。

また、PHP スクリプトが使用する設定で ccvsd プロセスを実行しておく 必要があります。CCVS がインストールされたのと同じユーザで PHP プロセスが 実行されているかを確認する必要があります (例えば、ユーザ 'ccvs' で CCVS をインストールした場合には、PHP プロセスのユーザも同様に 'ccvs' として実行する必要があります)。

参考

RedHat は CCVS のサポートを終了しています。しかし、やや古めですが 現在でも有用な文書を » http://www.redhat.com/docs/manuals/ccvs/ から入手可能です。

---

ccvs_add

(PHP 4 >= 4.0.3)

ccvs_add — トランザクションにデータを追加する

説明

string **ccvs_add** (string `$session` , string `$invoice` , string `$argtype` , string `$argval`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_auth

(PHP 4 >= 4.0.3)

ccvs_auth — あるトランザクションにおいてクレジット認証を行う

説明

string **ccvs_auth** (string `$session` , string `$invoice`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_command

(PHP 4 >= 4.0.3)

ccvs_command — 単一のプロトコルに基づく、一般的な CCVS API で利用できないコマンドを実行する

説明

string **ccvs_command** (string `$session` , string `$type` , string `$argval`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_count

(PHP 4 >= 4.0.3)

ccvs_count — システムに保存された指定した型のトランザクション数を得る

説明

int **ccvs_count** (string $session , string $type)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_delete

(PHP 4 >= 4.0.3)

ccvs_delete — トランザクションを削除する

説明

string **ccvs_delete** (string $session , string $invoice)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_done

(PHP 4 >= 4.0.3)

ccvs_done — CCVS エンジンを終了し、後始末をする

説明

string **ccvs_done** (string $sess)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_init

(PHP 4 >= 4.0.3)

ccvs_init — CCVS を初期化する

説明

string **ccvs_init** (string $name)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_lookup

(PHP 4 >= 4.0.3)

ccvs_lookup — データベース # にある特定の型の項目を探す

説明

string **ccvs_lookup** (string $session , string $invoice , int $inum)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_new

(PHP 4 >= 4.0.3)

ccvs_new — 新規に空のトランザクションを生成する

説明

string **ccvs_new** (string $session , string $invoice)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_report

(PHP 4 >= 4.0.3)

ccvs_report — バックグラウンドの通信プロセスの状態を返す

説明

string **ccvs_report** (string $session , string $type)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_return

(PHP 4 >= 4.0.3)

ccvs_return — クレジットカード所有者に店側から金を伝送する

説明

string **ccvs_return** (string $session , string $invoice)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_reverse

(PHP 4 >= 4.0.3)

ccvs_reverse — 処理済みの認証を完全に取り消す

説明

string **ccvs_reverse** (string $session , string $invoice)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ccvs_sale

(PHP 4 >= 4.0.3)

ccvs_sale — クレジットカード所有者から店側に金を伝送する

説明

string **ccvs_sale** (string $session , string $invoice)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ccvs_status

(PHP 4 >= 4.0.3)

ccvs_status — 請求書のステータスを確認する

説明

string **ccvs_status** (string `$session` , string `$invoice`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ccvs_textvalue

(PHP 4 >= 4.0.3)

ccvs_textvalue — 前の関数コールに関する返り値を取得する

説明

string **ccvs_textvalue** (string `$session`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ccvs_void

(PHP 4 >= 4.0.3)

ccvs_void — 完了したトランザクションを完全に破棄する

説明

string **ccvs_void** (string `$session` , string `$invoice`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

クラス/オブジェクト関数

導入

以下の関数により、クラスやインスタンスオブジェクトに関する情報を 得ることが可能となります。オブジェクトが属するクラスの名前、その メンバープロパティ、メソッドを取得可能です。 この関数を使用することにより、オブジェクトのクラスメンバーだけで なく親クラス(すなわちそのオブジェクトクラスの派生元)の情報を得る ことも可能です。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

例

この例では、まず基底クラスおよびそのクラスの派生クラスを定義しま す。基底クラスは食用か否か、色とかいった、一般的な野菜を記述しま す。サブクラス*Spinach*はその野菜の料理法と調理 済であるかどうかの情報を追加します。

Example#1 classes.inc

```php
<?php

// メンバープロパティとメソッドを有する基底クラス
class Vegetable {

    var $edible;
    var $color;

    function Vegetable($edible, $color="green")
    {
        $this->edible = $edible;
        $this->color = $color;
    }

    function is_edible()
    {
        return $this->edible;
    }

    function what_color()
    {
        return $this->color;
    }

} // クラスVegetableの終り

// 基底クラスを拡張する
class Spinach extends Vegetable {

    var $cooked = false;

    function Spinach()
    {
        $this->Vegetable(true, "green");
    }

    function cook_it()
    {
        $this->cooked = true;
    }

    function is_cooked()
    {
```

```
        return $this->cooked;
    }

} // クラスSpinachの終り

?>
```

続いて、これらのクラスから二つのオブジェクトのインスタンスを作成し、 親クラスを含む情報を出力します。 また、いくつかのユーティリティ関数を定義します。これらは主に変数 を格好良く表示するためのものです。

Example#2 test_script.php

```
<pre>
<?php

include "classes.inc";

// ユーティリティ関数

function print_vars($obj)
{
     foreach (get_object_vars($obj) as $prop => $val) {
         echo "¥t$prop = $val¥n";
     }
}

function print_methods($obj)
{
     $arr = get_class_methods(get_class($obj));
      foreach ($arr as $method) {
         echo "¥tfunction $method()¥n";
     }
}

function class_parentage($obj, $class)
{
     if (is_subclass_of($GLOBALS[$obj], $class)) {
         echo "Object $obj belongs to class " . get_class($$obj);
         echo " a subclass of $class¥n";
     } else {
         echo "Object $obj does not belong to a subclass of $class¥n";
     }
}

// 二つのオブジェクトのインスタンスを作成

$veggie = new Vegetable(true, "blue");
$leafy = new Spinach();

// オブジェクトに関する情報を出力
echo "veggie: CLASS " . get_class($veggie) . "¥n";
echo "leafy: CLASS " . get_class($leafy);
echo ", PARENT " . get_parent_class($leafy) . "¥n";

// veggieのプロパティを表示
echo "\nveggie: プロパティ\n";
print_vars($veggie);

// そしてleafyのメソッドを表示
echo "\nleafy: メソッド\n";
print_methods($leafy);

echo "¥nParentage:¥n";
class_parentage("leafy", "Spinach");
class_parentage("leafy", "Vegetable");
?>
</pre>
```

注意すべき大事な点ですが、上記の例ではオブジェクト *$leafy*は Vegetableのサブクラスであるクラス Spinachのインスタンスであり、 このスクリプトの最後の部分は以下のような出力となります。

```
        [...]
Parentage:
Object leafy does not belong to a subclass of Spinach
Object leafy belongs to class spinach a subclass of Vegetable
```

---

call_user_method_array

(PHP 4 >= 4.0.5, PHP 5)

call_user_method_array — パラメータの配列を指定してユーザメソッドをコールする [古い関数]

説明

mixed **call_user_method_array** (string `$method_name` , object `&$obj` , array `$params`)

警告

PHP 4.1.0 以降、call_user_method() 関数は非推奨となっています。

例

Example#1 call_user_method_array() の代替策

```
<?php
call_user_func_array(array($obj, $method_name), $params);
call_user_func_array(array(&$obj, $method_name), $params); // PHP 4
?>
```

参考

- call_user_func_array()
- call_user_func()

---

call_user_method

(PHP 4, PHP 5)

call_user_method — 指定したオブジェクトのユーザーメソッドをコールする [古い関数]

説明

mixed **call_user_method** (string $method_name , object &$obj [, mixed $parameter [, mixed $...]])

警告

PHP 4.1.0 以降、**call_user_method()** 関数は非推奨となっています。

例

Example#1 call_user_method() の代替策

```
<?php
call_user_func(array($obj, $method_name), $parameter  /* , ... */);
call_user_func(array(&$obj, $method_name), $parameter  /* , ... */); // PHP 4
?>
```

参考

- call_user_func_array()
- call_user_func()

---

class_exists

(PHP 4, PHP 5)

class_exists — クラスが定義済みかどうかを確認する

説明

bool **class_exists** (string $class_name [, bool $autoload])

この関数は指定したクラスが定義されているかどうかを調べます。

パラメータ

class_name

クラス名。大文字小文字は区別しません。

autoload

デフォルトで __autoload をコールするかしないか。 デフォルトは **TRUE** です。

返り値

クラス *class_name* が定義されている場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.2	定義済みのインターフェイスに対しては **TRUE** を返さないようになりました。 インターフェイスについては interface_exists() を使用します。
5.0.0	*autoload* パラメータが追加されました。

例

Example#1 class_exists() の例

```
<?php
// クラスを使用する前に、それが存在するかどうかを調べます
if (class_exists('MyClass')) {
    $myclass = new MyClass();
}

?>
```

Example#2 *autoload* パラメータの例

```
<?php
function __autoload($class)
{
    include($class . '.php');

    // クラス宣言を含むかどうか確認する
    if (!class_exists($class, false)) {
        trigger_error("Unable to load class: $class", E_USER_WARNING);
    }
}

if (class_exists('MyClass')) {
    $myclass = new MyClass();
}

?>
```

参考

- function_exists()
- interface_exists()
- get_declared_classes()

---

get_class_methods

(PHP 4, PHP 5)

get_class_methods — クラスメソッドの名前を取得する

説明

array **get_class_methods** (mixed `$class_name`)

クラスメソッドの名前を取得します。

パラメータ

class_name

オブジェクトのインスタンスのクラス名。

返り値

この関数は、指定したクラス *class_name* についてメソッドの名前を配列として返します。 エラー時には **NULL** を返します。

変更履歴

バージョン	説明
5.0.0	PHP 5 以降、この関数は宣言されているままのメソッド名 (大文字小文字を区別) を返します。PHP 4 では、小文字で返されます。
4.0.6	オブジェクト自身を指定することが可能となりました。

例

Example#1 get_class_methods() の例

```
<?php

class myclass {
    // コンストラクタ
    function myclass()
    {
        return(true);
    }

    // メソッド1
    function myfunc1()
    {
        return(true);
    }

    // メソッド2
    function myfunc2()
    {
        return(true);
    }
}

$class_methods = get_class_methods('myclass');
// あるいは
$class_methods = get_class_methods(new myclass());

foreach ($class_methods as $method_name) {
    echo "$method_name¥n";
}

?>
```

上の例の出力は以下となります。

```
myclass
myfunc1
myfunc2
```

参考

- get_class()
- get_class_vars()
- get_object_vars()

---

get_class_vars

(PHP 4, PHP 5)

get_class_vars — クラスのデフォルトプロパティを取得する

説明

array **get_class_vars** (string `$class_name`)

指定したクラスのデフォルトプロパティを取得します。

パラメータ

`class_name`

クラス名。

返り値

クラスのデフォルト public プロパティを有する連想配列を返します。 返される配列要素は、***変数名*** => ***値*** の形式となります。

変更履歴

バージョン	説明
4.2.0 より前	初期化されていないクラス変数は、 **get_class_vars()** で返されません。

例

Example#1 get_class_vars() の例

```php
<?php

class myclass {

    var $var1; // この変数にはデフォルト値がありません...
    var $var2 = "xyz";
    var $var3 = 100;
     private $var4; // PHP 5

    // コンストラクタ
    function myclass() {
        // いくつかのプロパティを変更する
         $this->var1 = "foo";
         $this->var2 = "bar";
          return true;
    }

}

$my_class = new myclass();

$class_vars = get_class_vars(get_class($my_class));

foreach ($class_vars as $name => $value) {
    echo "$name : $value¥n";
}

?>
```

上の例の出力は以下となります。

```
// PHP 4.2.0 未満
var2 : xyz
var3 : 100

// PHP 4.2.0 以降
var1 :
var2 : xyz
var3 : 100
```

参考

- get_class_methods()
- get_object_vars()

---

get_class

(PHP 4, PHP 5)

get_class — オブジェクトのクラス名を返す

説明

string **get_class** ([object `$object`])

指定した *object* のクラス名を取得します。

パラメータ

object

調べるオブジェクト。

返り値

オブジェクト *object* がインスタンス であるクラスの名前を返します。*object* がオブジェクトでない場合には**FALSE**が返されます。

変更履歴

バージョン	説明
5.0.0 以降	クラス名はその本来の表記で返されます。
5.0.0 以降	*object* パラメータは、 オブジェクトのメソッドからコールされた場合はオプションとなります。

例

Example#1 get_class() の使用例

```php
<?php

class foo {
     function foo()
    {
    // ここにロジックを書く
    }

     function name()
    {
         echo "My name is " , get_class($this) , "¥n";
    }
}

// オブジェクトを生成
$bar = new foo();

// 外部からコール
echo "Its name is " , get_class($bar) , "¥n";

// 内部からコール
$bar->name();

?>
```

上の例の出力は以下となります。

```
Its name is foo
My name is foo
```

参考

- get_parent_class()
- gettype()
- is_subclass_of()

---

get_declared_classes

(PHP 4, PHP 5)

get_declared_classes — 定義済のクラスの名前を配列として返す

説明

array **get_declared_classes** (void)

Gets the declared classes.

返り値

この関数は、現在のスクリプトで宣言されたクラスの名前の配列を返します。

> **注意**: PHP 4.0.1では、この他に次の3つのクラスが配列の先頭に返されます。：*stdClass* (*Zend/zend.c*で定義)、*OverloadedTestClass* (*ext/standard/basic_functions.c*で定義)、 *Directory* (*ext/standard/dir.c*で定義)
> PHP にコンパイル時に組み込んだり読み込んだりしている拡張モジュールの種類に依存して、 他のクラスも存在する可能性があることにも注意してください。 これは、自作のクラスをそれらと同じ名前で作成できないことを意味します。 定義済みのクラスについては付録の定義済みクラスのセクションを 参照してください。

例

Example#1 get_declared_classes() の例

```php
<?php
print_r(get_declared_classes());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => stdClass
     [1] => __PHP_Incomplete_Class
     [2] => Directory
)
```

参考

- class_exists()
- get_declared_interfaces()

---

get_declared_interfaces

(PHP 5)

get_declared_interfaces — 宣言されている全てのインターフェースの配列を返す

説明

array **get_declared_interfaces** (void)

宣言されているインターフェイスを取得します。

返り値

現在のスクリプトで宣言されているインターフェース名の配列を返します。

例

Example#1 get_declared_interfaces() の例

```
<?php
print_r(get_declared_interfaces());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
      [0] => Traversable
      [1] => IteratorAggregate
     [2] => Iterator
      [3] => ArrayAccess
     [4] => reflector
      [5] => RecursiveIterator
      [6] => SeekableIterator
)
```

参考

- get_declared_classes()

---

get_object_vars

(PHP 4, PHP 5)

get_object_vars — オブジェクトのプロパティを取得する

説明

array **get_object_vars** (object `$object`)

指定した *object* のプロパティを取得します。

パラメータ

object

オブジェクトのインスタンス。

返り値

指定したオブジェクト *object* のプロパティを連想配列として返します。プロパティに値が設定されていない場合は、 **NULL** 値が返されます。

変更履歴

バージョン	説明
4.2.0 より前 *object* をインスタンスとするクラスで宣言された変数が	値を持っていない場合、それらの変数は配列では返されません。

例

Example#1 get_object_vars() の使用例

```
<?php
class Point2D {
     var $x, $y;
     var $label;

     function Point2D($x, $y)
     {
         $this->x = $x;
         $this->y = $y;
     }

     function setLabel($label)
     {
         $this->label = $label;
     }

     function getPoint()
     {
          return array("x" => $this->x,
                       "y" => $this->y,
                       "label" => $this->label);
     }
}

// "$label" は宣言されていますが、未定義です。
$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));

$p1->setLabel("point #1");
print_r(get_object_vars($p1));

?>
```

上の例の出力は以下となります。

```
 Array
 (
      [x] => 1.233
      [y] => 3.445
      [label] =>
 )

 Array
 (
      [x] => 1.233
      [y] => 3.445
      [label] => point #1
 )
```

参考

- get_class_methods()
- get_class_vars()

---

get_parent_class

(PHP 4, PHP 5)

get_parent_class — オブジェクトの親クラスの名前を取得する

説明

string **get_parent_class** ([mixed `$object`])

オブジェクトあるいはクラスの親クラスの名前を取得します。

パラメータ

object

調べたいオブジェクトあるいはクラスの名前。

返り値

object がインスタンスあるいは名前であるクラスの親クラス名を返します。

オブジェクトの外部からこのパラメータを省略してコールすると、 この関数は **FALSE** を返します

変更履歴

バージョン	説明
5.1.0 より前	オブジェクトの外部からパラメータなしでコールすると、 この関数は警告を発生したうえで **NULL** を返します。
5.0.0 以降	オブジェクトのメソッドからコールされた場合、パラメータ *object* はオプションとなります。
4.0.5 以降	*object* が文字列の場合、 その名前のクラスの親クラスの名前を返します。

例

Example#1 get_parent_class() の使用例

```
<?php

class dad {
     function dad()
     {
    // ロジックを実装する
     }
}

class child extends dad {
     function child()
     {
          echo "I'm " , get_parent_class($this) , "'s son¥n";
     }
}

class child2 extends dad {
     function child2()
     {
          echo "I'm " , get_parent_class('child2') , "'s son too¥n";
     }
}

$foo = new child();
$bar = new child2();

?>
```

上の例の出力は以下となります。

```
I'm dad's son
I'm dad's son too
```

参考

- get_class()
- is_subclass_of()

---

interface_exists

(PHP 5 >= 5.0.2)

interface_exists — インターフェースが宣言されているかどうかを確認する

説明

bool **interface_exists** (string `$interface_name` [, bool `$autoload`])

指定したインターフェイスが定義されているかどうかを調べます。

パラメータ

interface_name

インターフェイス名。

autoload

デフォルトで __autoload をコールするかどうか。

返り値

interface_name で与えられたインターフェースが宣言されていれば **TRUE** を返します。 そうでなければ **FALSE** を返します。

例

Example#1 interface_exists() の例

```
<?php
// 使用する前にインターフェースが存在するかどうかを確認する
if (interface_exists('MyInterface')) {
    class MyClass implements MyInterface
    {
        // メソッド
    }
}

?>
```

参考

- class_exists()

---

is_a

(PHP 4 >= 4.2.0, PHP 5)

is_a — オブジェクトがこのクラスのものであるか、このクラスをその親クラスのひとつとしているかどうかを調べる

説明

bool **is_a** (object $object , string $class_name)

指定した *object* がこのクラスのものであるか、 あるいはこのクラスをその親クラスのひとつとしているかどうかを調べます。

> **注意**: **is_a()** 関数は PHP 5 では非推奨となりました。かわりに instanceof 演算子を使用してください。

パラメータ

object

調べたいオブジェクト。

class_name

クラス名。

返り値

オブジェクトがこのクラスのものであるか、 あるいはこのクラスをその親クラスのひとつとしている場合に **TRUE**、それ以外の場合に **FALSE** を返します。

例

Example#1 is_a() の例

```
<?php
// クラス定義
class WidgetFactory
{
  var $oink = 'moo';
}

// オブジェクトを作成します
$WF = new WidgetFactory();

if (is_a($WF, 'WidgetFactory')) {
  echo "はい、\$WF は WidgetFactory です\n";
}
?>
```

Example#2 PHP 5 での *instanceof* 演算子の使用

```
<?php
if ($WF instanceof WidgetFactory) {
    echo 'はい、$WF は WidgetFactory です';
```

```
}
?>
```

参考

- get_class()
- get_parent_class()
- is_subclass_of()

---

is_subclass_of

(PHP 4, PHP 5)

is_subclass_of — あるオブジェクトが指定したクラスのサブクラスに属するかどうかを調べる

説明

bool **is_subclass_of** (mixed `$object` , string `$class_name`)

指定した *object* が、その親のひとつに *class_name* を持つかどうかを調べます。

パラメータ

object

クラス名あるいはオブジェクトのインスタンス。

class_name

クラス名。

返り値

この関数は、オブジェクト *object* が *superclass* のサブクラスであるクラスに属する場合に **TRUE**、その他の場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.3	*object* パラメータに文字列 (クラス名) を指定可能です。

例

Example#1 is_subclass_of() の例

```
<?php
// クラスを定義する
class WidgetFactory
{
   var $oink = 'moo';
}

// 子クラスを定義する
class WidgetFactory_Child  extends WidgetFactory
{
   var $oink = 'oink';
}

// 新規オブジェクトを生成する
$WF = new WidgetFactory();
$WFC = new WidgetFactory_Child();

if (is_subclass_of($WFC, 'WidgetFactory')) {
   echo "yes, ¥$WFC is a subclass of WidgetFactory¥n";
} else {
   echo "no, ¥$WFC is not a subclass of WidgetFactory¥n";
}


if (is_subclass_of($WF, 'WidgetFactory')) {
   echo "yes, ¥$WF is a subclass of WidgetFactory¥n";
} else {
   echo "no, ¥$WF is not a subclass of WidgetFactory¥n";
}


// PHP 5.0.3 以降で使用可能
if (is_subclass_of('WidgetFactory_Child', 'WidgetFactory')) {
   echo "yes, WidgetFactory_Child is a subclass of WidgetFactory¥n";
} else {
   echo "no, WidgetFactory_Child is not a subclass of WidgetFactory¥n";
```

```
}
?>
```

上の例の出力は以下となります。

```
yes, $WFC is a subclass of WidgetFactory
no, $WF is not a subclass of WidgetFactory
yes, WidgetFactory_Child is a subclass of WidgetFactory
```

参考

- get_class()
- get_parent_class()
- is_a()

---

method_exists

(PHP 4, PHP 5)

method_exists — クラスメソッドが存在するかどうかを確認する

説明

bool **method_exists** (object $object , string $method_name)

指定した *object* にクラスメソッドが存在するかどうかを調べます。

パラメータ

object

オブジェクトのインスタンス。

method_name

メソッドの名前。

返り値

method_name で指定したメソッドが 指定した *object* において定義されている場合に **TRUE**、そうでない場合に **FALSE** を返します。

例

Example#1 method_exists() の例

```
<?php
$directory = new Directory('.');
var_dump(method_exists($directory,'read'));
?>
```

上の例の出力は以下となります。

```
bool(true)
```

Example#2 静的な method_exists() の例

```
<?php
$directory = new Directory('.');
var_dump(method_exists('Directory','read'));
?>
```

上の例の出力は以下となります。

```
bool(true)
```

参考

- function_exists()
- is_callable()

property_exists

(PHP 5 >= 5.1.0)

property_exists — オブジェクトもしくはクラスにプロパティが存在するかどうかを調べる

説明

bool **property_exists** (mixed `$class` , string `$property`)

この関数は、与えられたプロパティ *property* が 指定されたクラスに存在するかどうか (かつ現在のスコープからアクセス可能かどうか) を確認します。

> **注意**: isset() とは対象的に、 プロパティの値が **NULL** の場合でも **property_exists()** は **TRUE** を返します。

パラメータ

class

> 確認するクラス名、もしくはクラスのオブジェクトを指定します。

property

> プロパティ名を指定します。

返り値

プロパティが存在している場合は **TRUE**、存在していない場合に **FALSE**、 エラー時には **NULL** を返します。

例

Example#1 property_exists() の例

```
<?php

class myClass {
     public $mine;
     private $xpto;

     static function test() {
        var_dump(property_exists('myClass', 'xpto')); //  ここからアクセス可能なので  true
     }
}

var_dump(property_exists('myClass', 'mine'));    //true
var_dump(property_exists(new myClass, 'mine')); //true
var_dump(property_exists('myClass', 'xpto'));   //public  でないので  false
myClass::test();

?>
```

参考

- method_exists()

目次

- property_exists — オブジェクトもしくはクラスにプロパティが存在するかどうかを調べる

Classkit 関数

導入

これらの関数は、PHP クラスの動的な操作を実行時に行います。

> **注意**: この拡張モジュールは runkit に置き換えられ、 クラスの操作だけでなく関数の操作も同様に可能となっています。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/classkit.

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

CLASSKIT_ACC_PRIVATE (int)
メソッドを *private* としてマークする
CLASSKIT_ACC_PROTECTED (int)
メソッドを *protected* としてマークする
CLASSKIT_ACC_PUBLIC (int)
メソッドを *public* としてマークする

---

classkit_import

(PECL classkit:0.3-0.4 runkit:0.7-0.9)

classkit_import — 新しいクラスメソッドの定義をファイルから読み込む

説明

array **classkit_import** (string `$filename`)

> **注意**: この関数により現在実行中 (もしくはチェーンド)のメソッドを操作することはできません。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

`filename`

クラスメソッドが定義されているファイルの名前。

返り値

読み込まれたメソッドの連想配列を返します。

例

Example#1 classkit_import() の例

```
<?php
// ファイル名: newclass.php
class Example {
     function foo() {
         return "bar!¥n";
     }
}
?>
<?php
// newclass.php が必要（上を参照）
class Example {
     function foo() {
         return "foo!¥n";
     }
}

$e = new Example();

// 本来の出力
echo $e->foo();

// 置き換えるメソッドの読み込み
classkit_import('newclass.php');

// 置き換えられたメソッドの出力
echo $e->foo();

?>
```

上の例の出力は以下となります。

```
foo!
bar!
```

参考

- classkit_method_add()
- classkit_method_copy()

---

classkit_method_add

(PECL classkit:0.1-0.4 runkit:0.7-0.9)

classkit_method_add — 指定したクラスに、新しいメソッドを動的に追加する

説明

bool **classkit_method_add** (string `$classname` , string `$methodname` , string `$args` , string `$code` [, int `$flags`])

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

`classname`

メソッドを追加するクラスの名前。

`methodname`

追加するメソッドの名前。

`args`

カンマで区切った、新しいメソッドの引数。

`code`

methodname がコールされた際に 評価されるコード。

flags

作成するメソッドの型。 **CLASSKIT_ACC_PUBLIC**、 **CLASSKIT_ACC_PROTECTED** あるいは **CLASSKIT_ACC_PRIVATE** のいずれか。

注意: このパラメータは PHP 5 以降でのみ使用されます。なぜなら、それ以前の バージョンでは全てのメソッドが public だからです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 classkit_method_add() の例

```
<?php
class Example {
     function foo() {
         echo "foo!¥n";
     }
}

// Example のオブジェクトを作成
$e = new Example();

// 新しい public メソッドの追加
classkit_method_add(
     'Example',
     'add',
     '$num1, $num2',
     'return $num1 + $num2;',
     CLASSKIT_ACC_PUBLIC
);

// 12 + 4 を計算する
echo $e->add(12, 4);
?>
```

上の例の出力は以下となります。

```
16
```

参考

- classkit_method_copy()
- classkit_method_redefine()
- classkit_method_remove()
- classkit_method_rename()
- create_function()

---

classkit_method_copy

(PECL classkit:0.2-0.4 runkit:0.7-0.9)

classkit_method_copy — あるクラスのメソッドを別のクラスにコピーする

説明

bool **classkit_method_copy** (string $dClass , string $dMethod , string $sClass [, string $sMethod])
警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

dClass

メソッドのコピー先のクラス。

dMethod

コピー先のメソッド名。

sClass

メソッドのコピー元のクラス。

sMethod

元のクラスからコピーするメソッドの名前。指定されなかった場合は *dMethod* と同じであるとみなされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 classkit_method_copy() の例

```
<?php
class Foo {
     function example() {
         return "foo!¥n";
     }
}

class Bar {
    // 最初は、何もメソッドがない
}

// Foo クラスの example() メソッドを Bar クラスに baz() という名前でコピーする
classkit_method_copy('Bar', 'baz', 'Foo', 'example');

// コピーされた関数の出力
echo Bar::baz();
?>
```

上の例の出力は以下となります。

```
foo!
```

参考

- classkit_method_add()
- classkit_method_redefine()
- classkit_method_remove()
- classkit_method_rename()

---

classkit_method_redefine

(PECL classkit:0.1-0.4 runkit:0.7-0.9)

classkit_method_redefine — 指定されたメソッドのコードを動的に変更する

説明

bool **classkit_method_redefine** (string `$classname` , string `$methodname` , string `$args` , string `$code` [, int `$flags`])

注意: この関数により現在実行中 (もしくはチェーンド)のメソッドを操作することはできません。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

classname

メソッドを再定義するクラス。

methodname

再定義するメソッドの名前。

args

カンマで区切られた、再定義後のメソッドの引数。

code

methodname がコールされた際に 評価される、新しいコード。

flags

再定義するメソッドの型。 **CLASSKIT_ACC_PUBLIC**、 **CLASSKIT_ACC_PROTECTED** あるいは **CLASSKIT_ACC_PRIVATE** のいずれか。

> **注意**: このパラメータは PHP 5 以降でのみ使用されます。なぜなら、それ以前の バージョンでは全てのメソッドが public だからです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 classkit_method_redefine() の例

```
<?php
class Example {
     function foo() {
         return "foo!¥n";
     }
}

// Example オブジェクトの作成
$e = new Example();

// Example::foo() の出力（再定義前）
echo "Before: " . $e->foo();

// 'foo' メソッドの再定義
classkit_method_redefine(
     'Example',
     'foo',
     '',
     'return "bar!¥n";',
     CLASSKIT_ACC_PUBLIC
);

// Example::foo() の出力（再定義後）
echo "After: " . $e->foo();
?>
```

上の例の出力は以下となります。

```
Before: foo!
After: bar!
```

参考

- classkit_method_add()
- classkit_method_copy()
- classkit_method_remove()
- classkit_method_rename()

---

classkit_method_remove

(PECL classkit:0.1-0.4 runkit:0.7-0.9)

classkit_method_remove — 指定したメソッドを動的に削除する

説明

bool **classkit_method_remove** (string `$classname` , string `$methodname`)

> **注意**: この関数により現在実行中 (もしくはチェーンド)のメソッドを操作することはできません。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変

更される可能性があります。この関数は自己責任で使用してください。

パラメータ

classname

メソッドを削除するクラス。

methodname

削除するメソッドの名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 classkit_method_remove() の例

```
<?php
class Example {
     function foo() {
         return "foo!¥n";
     }

     function bar() {
         return "bar!¥n";
     }
}

// 'foo' メソッドを削除する
classkit_method_remove(
     'Example',
     'foo'
);

echo implode(' ', get_class_methods('Example'));

?>
```

上の例の出力は以下となります。

```
bar
```

参考

- classkit_method_add()
- classkit_method_copy()
- classkit_method_redefine()
- classkit_method_rename()

---

classkit_method_rename

(PECL classkit:0.1-0.4 runkit:0.7-0.9)

classkit_method_rename — 指定したメソッドの名前を動的に変更する

説明

bool **classkit_method_rename** (string `$classname` , string `$methodname` , string `$newname`)

注意: この関数により現在実行中 (もしくはチェーンド)のメソッドを操作することはできません。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

classname

メソッド名を変更するクラス。

methodname

変更するメソッドの名前。

newname

変更後のメソッドの名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 classkit_method_rename() の例

```
<?php
class Example {
     function foo() {
         return "foo!¥n";
     }
}

// 'foo' メソッドの名前を 'bar' に変更する
classkit_method_rename(
     'Example',
     'foo',
     'bar'
);

// 変更後の関数の出力
echo Example::bar();
?>
```

上の例の出力は以下となります。

```
foo!
```

参考

- classkit_method_add()
- classkit_method_copy()
- classkit_method_redefine()
- classkit_method_remove()

目次

ClibPDF 関数 [非推奨]

導入

警告

ClibPDF のサポートは廃止予定です。代替候補としては、 Haru や PDFlib などがあります。または PHP での実装 のいずれかを使用するという方法もあります。

ClibPDF により、PDF ドキュメントを PHP により作成可能となります。 ClibPDF の機能と API は、PDFlib に 似ています。この文書とともに、よりライブラリの詳細を説明している ClibPDF のマニュアルも読んでください。

元の ClibPDF および PHP モジュールの多くの関数名は、 PDFlib と同様に同じです。 cpdf_open() 以外の全ての関数は、最初の パラメータとしてドキュメントのハンドルをとります。

現在、このハンドルは内部で使用されていません。これは、 ClibPDF が複数の PDF ドキュメントを同時に作成する機能をサポートして いないためで

す。実際、これを試みても、結果は期待できません。 マルチスレッド環境での結果を予測することはできません。ClibPDF の 作者によれば、これは将来のリリースで変更されます (これを書いている時点での最新版は 1.10 です)。 もしこの機能が必要なら、pdflib モジュールを使用してください。

ClipPDF (および PDFlib) が優れているのは、 テンポラリファイルを使用せずに pdf ドキュメントを 完全にメモリ上で作成することが可能であることです。 あらかじめ定義された単位長さの座標を渡す機能も有しています (この機能は、PDFlib 関数の pdf_translate() で模擬することも可能です)。

その他の ClibPDF の機能で優れているのは、新規のページのオープン後で あっても任意のページをいつでも修正可能であるという点です。関数 cpdf_set_current_page() により、カレントの ページを離れて、他のページを修正できるようになります。

多くの機能の使用法は非常に容易です。最も困難なのは、おそらく新規に PDF ドキュメントを作成する場合でしょう。次の例が導入の手助けと なるはずです。この例では、ページを 1 つ有するドキュメントを 作成します。このページは、30pt のアウトラインフォントでテキスト "Times-Roman" により書かれます。テキストには下線が引かれます。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.0.

> **注意**: 外部の PDF ライブラリを使用しない別のフリーな PDF 作成ツールに ついては、関連する FAQ を参照ください。

要件

ClibPDF 関数を使用するには、ClibPDF パッケージをインストールする 必要があります。ClibPDF は、» FastIO からダウンロード可能ですが、商用で使用する場合には ライセンスを購入する必要があります。 PHP では、cpdflib >= 2 を使用する必要があります。 ClibPDF ライブラリの開発は終了しており、FastIO のウェブサイトもそのうちにアクセスできなくなることでしょう。

インストール手順

この関数が動作するには、 --with-cpdflib[=DIR] を指定して PHP をコンパイルする必要があります。DIR は cpdflib をインストールする ディレクトリで、*/usr* がデフォルトです。 ClibPDF が使用する jpeg ライブラリと tiff ライブラリの場所を指定することも できます。これを行うには、configure にオプション --with-jpeg-dir[=DIR] および --with-tiff-dir[=DIR] を 指定してください。

実行時設定

設定ディレクティブは定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

CPDF_PM_NONE (integer)
CPDF_PM_OUTLINES (integer)
CPDF_PM_THUMBS (integer)
CPDF_PM_FULLSCREEN (integer)
CPDF_PL_SINGLE (integer)
CPDF_PL_1COLUMN (integer)
CPDF_PL_2LCOLUMN (integer)
CPDF_PL_2RCOLUMN (integer)

例

Example#1 簡単な ClibPDF の例

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 50);
cpdf_end_text($cpdf);
cpdf_moveto($cpdf, 50, 50);
cpdf_lineto($cpdf, 740, 330);
cpdf_stroke($cpdf);
cpdf_finalize_page($cpdf, 1);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

pdflib の配布ファイルには、アナログクロックを有する複数ページを 作成するような複雑な例が含まれています。以下に ClipPDF 拡張を使用して こ

の例を PHP に変換したものを示します。

Example#2 pdflib 2.0 配布ファイルからの pdfclock の例

```
<?php
$radius  = 200;
$margin  = 20;
$pagecount  = 40;

$pdf  = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php");
cpdf_set_title($pdf, "Analog  Clock");

while  ($pagecount-- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius  + $margin), 2 * ($radius  + $margin), 1.0);

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */

  cpdf_translate($pdf, $radius  + $margin, $radius  + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /* minute strokes */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha  = 0; $alpha  < 360; $alpha  += 6) {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
  }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* 5 minute strokes */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha  = 0; $alpha  < 360; $alpha  += 30) {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }

  $ltime  = getdate();

  /* draw hour hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw minute hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius  * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw second hand */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
  cpdf_lineto($pdf, $radius, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* draw little circle at center */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);

  cpdf_restore($pdf);

  cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type:  application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```

参考

PDFlib 拡張モジュールの ドキュメントも参照ください。

---

cpdf_add_annotation

(PHP 4, PHP 5 <= 5.0.5)

cpdf_add_annotation — 注記を追加する

説明

bool **cpdf_add_annotation** (int $pdf_document , float $llx , float $lly , float $urx , float $ury , string $title , string $content [, int $mode])

指定した位置に注記を追加します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

llx

左下の X。

lly

左下の Y。

urx

右上の X。

ury

右上の Y。

title

注記のタイトル。

text

注記の内容。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_add_outline

(PHP 4, PHP 5 <= 5.0.5)

cpdf_add_outline — 現在のページにブックマークを追加する

説明

int **cpdf_add_outline** (int $pdf_document , int $lastoutline , int $sublevel , int $open , int $pagenr , string $text)

テキスト *text* により、現在のページを指すブックマークを設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

lastoutline

sublevel

open

pagenr

text

The bookmark text

例

Example#1 ページアウトラインの追加

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
// ...
// some drawing
// ...
cpdf_finalize($cpdf);
header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

---

cpdf_arc

(PHP 4, PHP 5 <= 5.0.5)

cpdf_arc — 円弧を描く

説明

bool **cpdf_arc** (int $pdf_document , float $x_coor , float $y_coor , float $radius , float $start , float $end [, int $mode])

円弧を描きます。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x

中心の x 座標。

y

中心の y 座標。

radius

円弧の半径。

start

開始位置の角度。

end

終了位置の角度。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_circle()

---

cpdf_begin_text

(PHP 4, PHP 5 <= 5.0.5)

cpdf_begin_text — テキストセクションを開始する

説明

bool **cpdf_begin_text** (int `$pdf_document`)

テキストセクションを開始します。

テキストセクションは、cpdf_end_text() で終了する必要があります。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 テキスト出力

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some  text");
cpdf_end_text($pdf)
?>
```

参考

- cpdf_end_text()

---

cpdf_circle

(PHP 4, PHP 5 <= 5.0.5)

cpdf_circle — 円を描く

説明

bool **cpdf_circle** (int `$pdf_document` , float `$x_coor` , float `$y_coor` , float `$radius` [, int `$mode`])

円を描きます。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`x_coor`

中心の x 座標。

`y_coor`

中心の y 座標。

radius

円の半径。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_arc()

---

cpdf_clip

(PHP 4, PHP 5 <= 5.0.5)

cpdf_clip — 現在のパスを切り取る

説明

bool **cpdf_clip** (int `$pdf_document`)

現在のパスのすべての描画内容を切り取ります。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_close

(PHP 4, PHP 5 <= 5.0.5)

cpdf_close — pdf ドキュメントを閉じる

説明

bool **cpdf_close** (int `$pdf_document`)

PDF ドキュメントを閉じます。

この関数は、 cpdf_finalize(), cpdf_output_buffer(), cpdf_save_to_file() の後で、最後にコールする必要があります。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_open()

---

cpdf_closepath_fill_stroke

(PHP 4, PHP 5 <= 5.0.5)

cpdf_closepath_fill_stroke — パスを閉じ、塗りつぶし、描く

説明

bool **cpdf_closepath_fill_stroke** (int `$pdf_document`)

現在のパスを閉じ、内部を現在の塗りつぶし色で塗りつぶし、 描画します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_closepath()
- cpdf_stroke()
- cpdf_fill()
- cpdf_setgray_fill()
- cpdf_setgray()
- cpdf_setrgbcolor_fill()
- cpdf_setrgbcolor()

---

cpdf_closepath_stroke

(PHP 4, PHP 5 <= 5.0.5)

cpdf_closepath_stroke — パスを閉じ、線をパスに沿って描く

説明

bool **cpdf_closepath_stroke** (int `$pdf_document`)

現在のパスを閉じ、パスに沿って線を描きます。

cpdf_closepath_stroke() は cpdf_closepath() と cpdf_stroke() を組み合わせたものです。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_closepath()
- cpdf_stroke()

---

cpdf_closepath

(PHP 4, PHP 5 <= 5.0.5)

cpdf_closepath — パスを閉じる

説明

bool **cpdf_closepath** (int $pdf_document)

現在のパスを閉じます。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_continue_text

(PHP 4, PHP 5 <= 5.0.5)

cpdf_continue_text — 次の行にテキストを出力する

説明

bool **cpdf_continue_text** (int $pdf_document , string $text)

文字列 *text* を次の行に出力します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

text

出力する文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_show_xy()
- cpdf_text()
- cpdf_set_leading()
- cpdf_set_text_pos()

---

cpdf_curveto

(PHP 4, PHP 5 <= 5.0.5)

cpdf_curveto — 曲線を描く

説明

bool **cpdf_curveto** (int $pdf_document , float $x1 , float $y1 , float $x2 , float $y2 , float $x3 , float $y3 [, int $mode])

cpdf_curveto() 関数は、 (*x1* , *y1*) および (*x2* , *y2*) を制御点として 現在の点から (*x3* , *y3*) にベジエ曲線を 描きます。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x1

最初の制御点の x 座標。

y1

最初の制御点の y 座標。

x2

二番目の制御点の x 座標。

y2

二番目の制御点の y 座標。

x3

対象の x 座標。

y3

対象の y 座標。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_moveto()
- cpdf_rmoveto()
- cpdf_rlineto()
- cpdf_lineto()

---

cpdf_end_text

(PHP 4, PHP 5 <= 5.0.5)

cpdf_end_text — テキストセクションを終了する

説明

bool **cpdf_end_text** (int `$pdf_document`)

cpdf_begin_text() で開始したテキストセクションを終了します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 テキスト出力

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some  text");
```

```
cpdf_end_text($pdf)
?>
```

参考

- cpdf_begin_text()

---

cpdf_fill_stroke

(PHP 4, PHP 5 <= 5.0.5)

cpdf_fill_stroke — 現在のパスを塗りつぶし、描く

説明

bool **cpdf_fill_stroke** (int `$pdf_document`)

現在のパスの内部を現在の塗りつぶし色で塗りつぶし、 それを描画します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_closepath()
- cpdf_stroke()
- cpdf_fill()
- cpdf_setgray_fill()
- cpdf_setgray()
- cpdf_setrgbcolor_fill()
- cpdf_setrgbcolor()

---

cpdf_fill

(PHP 4, PHP 5 <= 5.0.5)

cpdf_fill — 現在のパスを塗りつぶす

説明

bool **cpdf_fill** (int `$pdf_document`)

現在の塗りつぶし色で現在のパスの内部を塗りつぶします。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_closepath()
- cpdf_stroke()
- cpdf_setgray_fill()

- cpdf_setgray()
- cpdf_setrgbcolor_fill()
- cpdf_setrgbcolor()

---

cpdf_finalize_page

(PHP 4, PHP 5 <= 5.0.5)

cpdf_finalize_page — ページを終了する

説明

bool **cpdf_finalize_page** (int $pdf_document , int $page_number)

ページ番号 *page number* のページを終了します。

この関数は、メモリに節約する意味しかありません。 ページを終了することにより、消費メモリは少なくなりますが、 修正することはできなくなります。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

page_number

ページ番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_page_init()

---

cpdf_finalize

(PHP 4, PHP 5 <= 5.0.5)

cpdf_finalize — ドキュメントを終了する

説明

bool **cpdf_finalize** (int $pdf_document)

PDF ドキュメントを終了します。

さらに cpdf_close() をコールする必要があります。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_close()

---

cpdf_global_set_document_limits

(PHP 4, PHP 5 <= 5.0.5)

cpdf_global_set_document_limits — PDF ドキュメントの制限を設定する

説明

bool **cpdf_global_set_document_limits** (int $maxpages , int $maxfonts , int $maximages , int $maxannotations , int $maxobjects)

ドキュメントのいくつかの制限を設定します。この関数を使用する際には、 cpdf_open() の前にコールする必要があります。 この関数は、この後オープンする全てのドキュメントに制限を設定します。

パラメータ

maxPages

ページ数の最大値。

maxFonts

フォント数の最大値。

maxImages

画像数の最大値。

maxAnnots

注記数の最大値。

maxObjects

オブジェクト数の最大値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_open()

---

cpdf_import_jpeg

(PHP 4, PHP 5 <= 5.0.5)

cpdf_import_jpeg — JPEG 画像をオープンする

説明

bool **cpdf_import_jpeg** (int $pdf_document , string $file_name , float $x_coor , float $y_coor , float $angle , float $width , float $height , float $x_scale , float $y_scale , int $gsave [, int $mode])

JPEG 画像をオープンします。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

file_name

画像ファイルへのパス。画像の形式は *JPEG* でなければなりません。

x_coor

画像の x 座標。

y_coor

画像の y 座標。

angle

画像の回転角度。

width

画像の幅。

height

画像の高さ。

x_scale

y_scale

gsave

この関数を正しく動作させるには、非ゼロを指定しなければなりません。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_place_inline_image()

---

cpdf_lineto

(PHP 4, PHP 5 <= 5.0.5)

cpdf_lineto — 線を描く

説明

bool **cpdf_lineto** (int $pdf_document , float $x_coor , float $y_coor [, int $mode])

cpdf_lineto() 関数は現在位置から (*x-coor* , *y-coor*) に線を描きます。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x_coor

対称点の x 座標。

y_coor

対称点の y 座標。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_moveto()
- cpdf_rmoveto()
- cpdf_curveto()

---

cpdf_moveto

(PHP 4, PHP 5 <= 5.0.5)

cpdf_moveto — 現在位置を設定する

説明

bool **cpdf_moveto** (int `$pdf_document` , float `$x_coor` , float `$y_coor` [, int `$mode`])

現在位置を、指定した場所に設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`x_coor`

x 座標。

`y_coor`

y 座標。

`mode`

オプションのパラメータ *`mode`* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

---

cpdf_newpath

(PHP 4, PHP 5 <= 5.0.5)

cpdf_newpath — 新規パスを開始する

説明

bool **cpdf_newpath** (int `$pdf_document`)

ドキュメント上で新しいパスを開始します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

---

cpdf_open

(PHP 4, PHP 5 <= 5.0.5)

cpdf_open — 新規 pdf ドキュメントをオープンする

説明

int **cpdf_open** (int `$compression` [, string `$filename` [, array `$doc_limits`]])

新しい PDF ドキュメントをオープンします。

パラメータ

`compression`

0 以外を指定すると、ドキュメントの圧縮を行います。

`filename`

ドキュメントを書き込むファイルを設定します。- は標準出力を表します。

省略すると、ドキュメントがメモリ上に作成され、 cpdf_save_to_file() でファイルに書き込んだり cpdf_output_buffer() で標準出力に書き込んだりします。

> **注意**: PHP が apache モジュールとしてコンパイルされている場合、 標準出力を使用するとうまく動作しません。この問題を解決するには、 ファイル名を指定せずに、後で cpdf_output_buffer() を使用して PDF ドキュメントを出力します。

`doc_limits`

ドキュメントの制限を表す配列。詳細は cpdf_global_set_document_limits() を参照ください。

返り値

ドキュメントのハンドルを返します。このライブラリのその他の関数で、 最初の引数としてこれを必要とします。

参考

- cpdf_close()
- cpdf_output_buffer()

---

cpdf_output_buffer

(PHP 4, PHP 5 <= 5.0.5)

cpdf_output_buffer — pdf ドキュメントをメモリバッファに出力する

説明

bool **cpdf_output_buffer** (int `$pdf_document`)

ドキュメントを標準出力に出力します。

ドキュメントは、メモリ上に作成されている 必要があります。メモリ上に作成されるのは、 cpdf_open() を filename パラメータを指定せずに コールした場合です。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- cpdf_open()

---

cpdf_page_init

(PHP 4, PHP 5 <= 5.0.5)

cpdf_page_init — 新規ページを開始する

説明

bool **cpdf_page_init** (int `$pdf_document` , int `$page_number` , int `$orientation` , float `$height` , float `$width` [, float `$unit`])

新しいページを開始します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`page_number`

ページ番号。

`orientation`

0 は縦長、1 は横長。

`height`

ページの高さ。

`width`

ページの幅。

`unit`

座標系の単位。この値は、 単位当たりのポストスクリプトのポイント数とする必要があります。 1 インチは 72 ポイントに等しいので、72 という値は、単位をインチに 設定します。デフォルト値も 72 です。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- cpdf_set_current_page()

---

cpdf_place_inline_image

(PHP 4, PHP 5 <= 5.0.5)

cpdf_place_inline_image — 画像をページに置く

説明

bool **cpdf_place_inline_image** (int `$pdf_document` , int `$image` , float `$x_coor` , float `$y_coor` , float `$angle` , float `$width` , float `$height` , int `$gsave` [, int `$mode`])

PHP のイメージ関数で作成された画像を配置します。 同時に画像の縮小・拡大を行うことができます。

> **注意**: この関数は、PHP が GD グラフィックスライブラリ 1.3 とともに コンパイルされている場合にのみ使用可能です。詳細な情報は GD 拡張モジュール のインストール手順を参照ください。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

image

PHP の画像関数で作成した画像。

x_coor

画像の x 座標。

y_coor

画像の y 座標。

angle

画像の回転角。

width

画像の幅。

height

画像の高さ。

gsave

この関数を正常に動作させるには、非ゼロでなければなりません。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_import_jpeg()

---

cpdf_rect

(PHP 4, PHP 5 <= 5.0.5)

cpdf_rect — 矩形を描く

説明

bool **cpdf_rect** (int `$pdf_document` , float `$x_coor` , float `$y_coor` , float `$width` , float `$height` [, int `$mode`])

矩形を描画します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x_coor

左下隅の x 座標。

y_coor

左下隅の y 座標。

width

矩形の幅。

height

矩形の高さ。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 矩形の描画

```
<?php

$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);

// set the fill color to red
cpdf_setrgbcolor($cpdf, 1, 0, 0);

// draw a (180 * 100) rectangle
cpdf_rect($cpdf, 645, 400, 180, 100);

// fill the rectangle
cpdf_fill($cpdf);

cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);

?>
```

---

cpdf_restore

(PHP 4, PHP 5 <= 5.0.5)

cpdf_restore — 以前に保存した環境を回復させる

説明

bool **cpdf_restore** (int `$pdf_document`)

cpdf_save()で保存した環境を回復します。 この関数は、postscript のコマンド grestore と同様に動作します。 他のオブジェクトに影響を与えずにあるオブジェクトを変換または 回転したい場合に非常に便利です。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 保存/回復

```
<?php
cpdf_save($pdf);
// do all kinds of rotations, transformations, ...
cpdf_restore($pdf)
?>
```

参考

- cpdf_save()

---

cpdf_rlineto

(PHP 4, PHP 5 <= 5.0.5)

cpdf_rlineto — 線を描く

説明

bool **cpdf_rlineto** (int $pdf_document , float $x_coor , float $y_coor [, int $mode])

cpdf_rlineto() 関数は、 現在位置から指定した相対位置に線を描きます。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x_coor

対称点の x 座標。

y_coor

対称点の y 座標。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_moveto()
- cpdf_rmoveto()
- cpdf_curveto()

---

cpdf_rmoveto

(PHP 4, PHP 5 <= 5.0.5)

cpdf_rmoveto — 現在位置を設定する

説明

bool **cpdf_rmoveto** (int $pdf_document , float $x_coor , float $y_coor [, int $mode])

現在の位置を、指定した座標からの相対位置として設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x_coor

対称点の x 座標。

y_coor

対称点の y 座標。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_moveto()

---

cpdf_rotate_text

(PHP 4, PHP 5 <= 5.0.5)

cpdf_rotate_text — テキスト回転角を設定する

説明

bool **cpdf_rotate_text** (int $pdfdoc , float $angle)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cpdf_rotate

(PHP 4, PHP 5 <= 5.0.5)

cpdf_rotate — 回転を設定する

説明

bool **cpdf_rotate** (int $pdf_document , float $angle)

回転角を *angle* 度に設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

angle

角度。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_save_to_file

(PHP 4, PHP 5 <= 5.0.5)

cpdf_save_to_file — pdf ドキュメントをファイルに書きこむ

説明

bool **cpdf_save_to_file** (int $pdf_document , string $filename)

メモリ上に作成された pdf ドキュメントをファイルに出力します。

この関数は、cpdf_open() のパラメータとして filename を指定してオープンした場合にはコールする必要はありません。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

filename

PDF ドキュメントを保存するパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_output_buffer()
- cpdf_open()

---

cpdf_save

(PHP 4, PHP 5 <= 5.0.5)

cpdf_save — 現在の環境を保存する

説明

bool **cpdf_save** (int `$pdf_document`)

現在の環境を保存します。 ポストスクリプトのコマンド gsave と同様に動作します。 他のオブジェクトに影響を与えずに、 あるオブジェクトを変換または回転したい場合に非常に便利です。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_restore()

---

cpdf_scale

(PHP 4, PHP 5 <= 5.0.5)

cpdf_scale — 倍率を設定する

説明

bool **cpdf_scale** (int `$pdf_document` , float `$x_scale` , float `$y_scale`)

両方向の倍率を指定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x_scale

X 座標のパーセント。デフォルトは *0.0*。

y_scale

Y 座標のパーセント。デフォルトは *0.0*。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_set_action_url

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_action_url — ハイパーリンクを設定する

説明

bool **cpdf_set_action_url** (int `$pdfdoc` , float `$xll` , float `$yll` , float `$xur` , float `$xur` , string `$url` [, int `$mode`])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cpdf_set_char_spacing

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_char_spacing — 文字間隔を設定する

説明

bool **cpdf_set_char_spacing** (int `$pdf_document` , float `$space`)

文字の間隔を設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`space`

文字列内の個々の文字の間隔。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_word_spacing()
- cpdf_set_leading()

---

cpdf_set_creator

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_creator — pdf ドキュメントの creator フィールドを設定する

説明

bool **cpdf_set_creator** (int `$pdf_document` , string `$creator`)

PDF ドキュメントの creator を設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

creator

ドキュメントの作者。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_subject()
- cpdf_set_title()
- cpdf_set_keywords()

---

cpdf_set_current_page

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_current_page — 現在のページを設定する

説明

bool **cpdf_set_current_page** (int `$pdf_document` , int `$page_number`)

全ての操作を行うページを設定します。cpdf_finalize_page() によりページを終了するまでページ間を移動することができます。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

page_number

ページ番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_finalize_page()

---

cpdf_set_font_directories

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

cpdf_set_font_directories — 外部フォントを使用した際、検索するディレクトリを設定する

説明

bool **cpdf_set_font_directories** (int `$pdfdoc` , string `$pfmdir` , string `$pfbdir`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cpdf_set_font_map_file

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

cpdf_set_font_map_file — 外部フォントを使用している場合、フォント名をファイル名変換マップに設定する

説明

bool **cpdf_set_font_map_file** (int $pdfdoc , string $filename)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cpdf_set_font

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_font — フォントの種類とサイズを選択する

説明

bool **cpdf_set_font** (int $pdf_document , string $font_name , float $size , string $encoding)

現在のフォントフェイス、フォントサイズおよびエンコーディングを設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

font_name

フォント名。現在サポートされているのは、 標準の postscript フォントだけです。

詳細な情報、特にアジア用フォント (訳注：日本語 フォントを含む CJK フォント) のサポートについては、ClibPDF のマニュアルを参照ください。

size

フォントサイズ。

encoding

次のいずれかの値を指定します。*MacRomanEncoding*、 *MacExpertEncoding*、 *WinAnsiEncoding* および *NULL* (フォントの組み込みエンコーディング)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_set_horiz_scaling

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_horiz_scaling — テキストの水平方向の倍率を設定する

説明

bool **cpdf_set_horiz_scaling** (int $pdf_document , float $scale)

水平方向の倍率を *scale* パーセントに設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

scale

設定する倍率。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_set_keywords

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_keywords — pdf ドキュメントの keywords フィールドを設定する

説明

bool **cpdf_set_keywords** (int `$pdf_document` , string `$keywords`)

PDF ドキュメントのキーワードを設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`keywords`

ドキュメントに設定するキーワード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_title()
- cpdf_set_creator()
- cpdf_set_subject()

---

cpdf_set_leading

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_leading — テキスト行の間隔を設定する

説明

bool **cpdf_set_leading** (int `$pdf_document` , float `$distance`)

テキストの行間 *`distance`* を設定します。これは cpdf_continue_text() でテキストを出力する場合に使用されます。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`distance`

テキストの行間隔を表す float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_continue_text()

---

cpdf_set_page_animation

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_page_animation — ページ間の移行時間を設定する

説明

bool **cpdf_set_page_animation** (int `$pdf_document` , int `$transition` , float `$duration` , float `$direction` , int `$orientation` , int `$inout`)

ページ間の移行に要する時間を設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`transition`

`transition` の値は次のようにすることができます。

- 0 は何もしない。
- 1 は、ページを表示する際に2行ずつスクロールする。
- 2 はページを表示する際に複数行ずつスクロールする。
- 3 はページの周りに箱を描く。
- 4 はページを表示する際に1行ずつスクロールする。
- 5 はページを描画する際に古いページを消去する。
- 6 は画面の一つの隅から他の隅に移動するように消すといった効果を用いる。
- 7 は古いページを新しいページで単純に置換する (デフォルト)。

`duration`

`duration` の値はページ間の切り替えに要する 秒数です。

`direction`

`orientation`

`inout`

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_set_subject

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_subject — pdf ドキュメントの subject フィールドを設定する

説明

bool **cpdf_set_subject** (int `$pdf_document` , string `$subject`)

PDF ドキュメントの *`subject`* を設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`subject`

ドキュメントの表題。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_title()
- cpdf_set_creator()
- cpdf_set_keywords()

---

cpdf_set_text_matrix

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_text_matrix — テキスト行列を設定する

説明

bool **cpdf_set_text_matrix** (int `$pdf_document` , array `$matrix`)

現在のテキストフォントに適用する変換を記述する行列を設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`matrix`

6 つの要素からなる、変換内容を指定する配列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_text_rendering()
- cpdf_set_text_rise()

---

cpdf_set_text_pos

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_text_pos — テキスト位置を設定する

説明

bool **cpdf_set_text_pos** (int `$pdf_document` , float `$x_coor` , float `$y_coor` [, int `$mode`])

次に cpdf_show() がコールされた時の テキストの位置を設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`x_coor`

テキスト位置の x 座標。

`y_coor`

テキスト位置の y 座標。

`mode`

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_show()
- cpdf_text()

---

cpdf_set_text_rendering

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_text_rendering — テキストのレンダリング法を定義する

説明

bool **cpdf_set_text_rendering** (int `$pdf_document` , int `$rendermode`)

テキストのレンダリング方法を指定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

rendermode

mode で使用可能な値は、 0=塗りつぶす、1=輪郭を描く、2=塗りつぶし輪郭を描く、3=不可視、 4=塗りつぶしクリッピングパスに追加、5=輪郭を描いてクリッピングパスに 追加、6=塗りつぶし輪郭を描いてクリッピングパスに追加、 7=クリッピングパスに追加です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_text_matrix()
- cpdf_set_text_rise()

---

cpdf_set_text_rise

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_text_rise — テキストの高さを設定する

説明

bool **cpdf_set_text_rise** (int `$pdf_document` , float `$value`)

テキストの高さを *value* 単位に設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

value

テキストの高さを表す float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_text_rendering()
- cpdf_set_text_matrix()

---

cpdf_set_title

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_title — pdf ドキュメントの title フィールドを設定する

説明

bool **cpdf_set_title** (int `$pdf_document` , string `$title`)

PDF ドキュメントの *title* を設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

title

ドキュメントのタイトル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_subject()
- cpdf_set_creator()
- cpdf_set_keywords()

---

cpdf_set_viewer_preferences

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_viewer_preferences — ドキュメントのビューワ上での表示方法を設定する

説明

bool **cpdf_set_viewer_preferences** (int `$pdfdoc` , array `$preferences`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cpdf_set_word_spacing

(PHP 4, PHP 5 <= 5.0.5)

cpdf_set_word_spacing — 単語間の間隔を設定する

説明

bool **cpdf_set_word_spacing** (int `$pdf_document` , float `$space`)

単語間の間隔を設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

space

間隔を表す float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_set_char_spacing()
- cpdf_set_leading()

---

cpdf_setdash

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setdash — 破線のパターンを設定する

説明

bool **cpdf_setdash** (int `$pdf_document` , float `$white` , float `$black`)

破線のパターンを指定したものに設定します。両方のパラメータを 0 にした場合は実線となります。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

white

白い部分の長さを表す float 値。

black

黒い部分の長さを表す float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_setflat

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setflat — flatness を設定する

説明

bool **cpdf_setflat** (int `$pdf_document` , float `$value`)

flatness を設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

value

flatness を表す 0 から 100 までの float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_setgray_fill

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setgray_fill — 塗りつぶし色をグレー値に設定する

説明

bool **cpdf_setgray_fill** (int `$pdf_document` , float `$value`)

パスを塗りつぶす現在のグレー値 *value* を設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

value

グレー値を表す float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_setrgbcolor_fill()

---

cpdf_setgray_stroke

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setgray_stroke — 描画色をグレー値に設定する

説明

bool **cpdf_setgray_stroke** (int `$pdf_document` , float `$gray_value`)

現在の描画色を、指定したグレー値に設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

gray_value

グレー値を表す float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_setrgbcolor_stroke()

---

cpdf_setgray

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setgray — 描画、塗りつぶし色をグレー値に設定する

説明

bool **cpdf_setgray** (int `$pdf_document` , float `$gray_value`)

現在の描画色および塗りつぶし色を、指定したグレー値に設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`gray_value`

グレー値を表す float 値。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- cpdf_setrgbcolor_stroke()
- cpdf_setrgbcolor_fill()

---

cpdf_setlinecap

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setlinecap — linecap パラメータを設定する

説明

bool **cpdf_setlinecap** (int `$pdf_document` , int `$value`)

linecap パラメータを設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`value`

0 = butt end, 1 = round, 2 = projecting square。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

---

cpdf_setlinejoin

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setlinejoin — linejoin パラメータを設定する

説明

bool **cpdf_setlinejoin** (int `$pdf_document` , int `$value`)

linejoin パラメータを設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

value

0 = miter, 1 = round, 2 = bevel。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_setlinewidth

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setlinewidth — 線幅を設定する

説明

bool **cpdf_setlinewidth** (int `$pdf_document` , float `$width`)

線幅を設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

width

線幅を表す float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_setmiterlimit

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setmiterlimit — miter のリミットを設定する

説明

bool **cpdf_setmiterlimit** (int `$pdf_document` , float `$value`)

miter のリミットを設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

value

miter のリミット。1 以上でなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

cpdf_setrgbcolor_fill

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setrgbcolor_fill — 塗りつぶし色を RGB カラー値に設定する

説明

bool **cpdf_setrgbcolor_fill** (int `$pdf_document` , float `$red_value` , float `$green_value` , float `$blue_value`)

指定した RGB カラー値をパスを塗りつぶす色として設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`red_value`

赤の値。0.0 (黒) から 1.0 (赤) までの float 値。

`green_value`

緑の値。0.0 (黒) から 1.0 (緑) までの float 値。

`blue_value`

青の値。0.0 (黒) から 1.0 (青) までの float 値。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- cpdf_setrgbcolor_stroke()
- cpdf_setrgbcolor()

---

cpdf_setrgbcolor_stroke

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setrgbcolor_stroke — 描画色を RGB カラー値に設定する

説明

bool **cpdf_setrgbcolor_stroke** (int `$pdf_document` , float `$red_value` , float `$green_value` , float `$blue_value`)

指定した RGB カラー値を現在の描画色として設定します。

パラメータ

`pdf_document`

cpdf_open() が返す、ドキュメントのハンドル。

`red_value`

赤の値。0.0 (黒) から 1.0 (赤) までの float 値。

`green_value`

緑の値。0.0 (黒) から 1.0 (緑) までの float 値。

`blue_value`

青の値。0.0 (黒) から 1.0 (青) までの float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_setrgbcolor()
- cpdf_setrgbcolor_fill()

---

cpdf_setrgbcolor

(PHP 4, PHP 5 <= 5.0.5)

cpdf_setrgbcolor — 描画色および塗りつぶし色を RGB 値に設定する

説明

bool **cpdf_setrgbcolor** (int $pdf_document , float $red_value , float $green_value , float $blue_value)

指定した RGB カラー値を現在の描画色および塗りつぶし色として設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

red_value

赤の値。0.0 (黒) から 1.0 (赤) までの float 値。

green_value

緑の値。0.0 (黒) から 1.0 (緑) までの float 値。

blue_value

青の値。0.0 (黒) から 1.0 (青) までの float 値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_setrgbcolor_stroke()
- cpdf_setrgbcolor_fill()

---

cpdf_show_xy

(PHP 4, PHP 5 <= 5.0.5)

cpdf_show_xy — 指定位置にテキストを出力する

説明

bool **cpdf_show_xy** (int $pdf_document , string $text , float $x_coor , float $y_coor [, int $mode])

指定した位置に文字列 *text* を出力します。

注意: 関数 **cpdf_show_xy()** は、オプションの パラメータを除き cpdf_text() と同じです。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

text

テキスト。

x_coor

テキスト位置の x 座標。

y_coor

テキスト位置の y 座標。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_text()

---

cpdf_show

(PHP 4, PHP 5 <= 5.0.5)

cpdf_show — 現在位置にテキストを出力する

説明

bool **cpdf_show** (int $pdf_document , string $text)

現在位置に *text* の文字列を出力します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

text

テキストを表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_text()
- cpdf_begin_text()
- cpdf_end_text()

---

cpdf_stringwidth

(PHP 4, PHP 5 <= 5.0.5)

cpdf_stringwidth — 現在のフォントのテキストの幅を返す

説明

float **cpdf_stringwidth** (int $pdf_document , string $text)

文字列 *text* の幅を返します。

この関数は、フォントがあらかじめ設定されている必要があります。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

text

テキストを表す文字列。

返り値

text の幅を返します。

参考

- cpdf_set_font()

---

cpdf_stroke

(PHP 4, PHP 5 <= 5.0.5)

cpdf_stroke — パスに沿って線を描く

説明

bool **cpdf_stroke** (int $pdf_document)

現在のパスに沿って線を描きます。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_closepath()
- cpdf_closepath_stroke()

---

cpdf_text

(PHP 4, PHP 5 <= 5.0.5)

cpdf_text — パラメータを元にテキストを出力する

説明

bool **cpdf_text** (int $pdf_document , string $text [, float $x_coor], float $y_coor [, int $mode [, float $orientation [, int $alignmode]]])

文字列 *text* を、指定した位置に出力します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

text

テキスト。

x_coor

テキスト位置の x 座標。

y_coor

テキスト位置の y 座標。

mode

オプションのパラメータ *mode* は単位長さを定義します。この値が 0 または値が省略された場合、このページで指定されたデフォルト値が使用されます。それ以外の場合、現在の単位を無視してポストスクリプトのポイントで座標が計測されます。

orientation

テキストの回転角。

alignmode

テキストの配置方法を指定します。 使用できる値は ClibPDF のドキュメントを参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- cpdf_show_xy()

---

cpdf_translate

(PHP 4, PHP 5 <= 5.0.5)

cpdf_translate — 座標系の原点を設定する

説明

bool **cpdf_translate** (int `$pdf_document` , float `$x_coor` , float `$y_coor`)

座標系の原点を、指定した場所に設定します。

パラメータ

pdf_document

cpdf_open() が返す、ドキュメントのハンドル。

x_coor

位置の x 座標。

y_coor

位置の y 座標。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

目次

- cpdf_circle — 円を描く
- cpdf_clip — 現在のパスを切り取る
- cpdf_close — pdf ドキュメントを閉じる
- cpdf_closepath_fill_stroke — パスを閉じ、塗りつぶし、描く
- cpdf_closepath_stroke — パスを閉じ、線をパスに沿って描く
- cpdf_closepath — パスを閉じる
- cpdf_continue_text — 次の行にテキストを出力する
- cpdf_curveto — 曲線を描く
- cpdf_end_text — テキストセクションを終了する
- cpdf_fill_stroke — 現在のパスを塗りつぶし、描く
- cpdf_fill — 現在のパスを塗りつぶす
- cpdf_finalize_page — ページを終了する
- cpdf_finalize — ドキュメントを終了する
- cpdf_global_set_document_limits — PDF ドキュメントの制限を設定する
- cpdf_import_jpeg — JPEG 画像をオープンする
- cpdf_lineto — 線を描く
- cpdf_moveto — 現在位置を設定する
- cpdf_newpath — 新規パスを開始する
- cpdf_open — 新規 pdf ドキュメントをオープンする
- cpdf_output_buffer — pdf ドキュメントをメモリバッファに出力する
- cpdf_page_init — 新規ページを開始する
- cpdf_place_inline_image — 画像をページに置く
- cpdf_rect — 矩形を描く
- cpdf_restore — 以前に保存した環境を回復させる
- cpdf_rlineto — 線を描く
- cpdf_rmoveto — 現在位置を設定する
- cpdf_rotate_text — テキスト回転角を設定する
- cpdf_rotate — 回転を設定する
- cpdf_save_to_file — pdf ドキュメントをファイルに書きこむ
- cpdf_save — 現在の環境を保存する
- cpdf_scale — 倍率を設定する
- cpdf_set_action_url — ハイパーリンクを設定する
- cpdf_set_char_spacing — 文字間隔を設定する
- cpdf_set_creator — pdf ドキュメントの creator フィールドを設定する
- cpdf_set_current_page — 現在のページを設定する
- cpdf_set_font_directories — 外部フォントを使用した際、検索するディレクトリを設定する
- cpdf_set_font_map_file — 外部フォントを使用している場合、フォント名をファイル名変換マップに設定する
- cpdf_set_font — フォントの種類とサイズを選択する
- cpdf_set_horiz_scaling — テキストの水平方向の倍率を設定する
- cpdf_set_keywords — pdf ドキュメントの keywords フィールドを設定する
- cpdf_set_leading — テキスト行の間隔を設定する
- cpdf_set_page_animation — ページ間の移行時間を設定する
- cpdf_set_subject — pdf ドキュメントの subject フィールドを設定する
- cpdf_set_text_matrix — テキスト行列を設定する
- cpdf_set_text_pos — テキスト位置を設定する
- cpdf_set_text_rendering — テキストのレンダリング法を定義する
- cpdf_set_text_rise — テキストの高さを設定する
- cpdf_set_title — pdf ドキュメントの title フィールドを設定する
- cpdf_set_viewer_preferences — ドキュメントのビューワ上での表示方法を設定する
- cpdf_set_word_spacing — 単語間の間隔を設定する
- cpdf_setdash — 破線のパターンを設定する
- cpdf_setflat — flatness を設定する
- cpdf_setgray_fill — 塗りつぶし色をグレー値に設定する
- cpdf_setgray_stroke — 描画色をグレー値に設定する
- cpdf_setgray — 描画、塗りつぶし色をグレー値に設定する
- cpdf_setlinecap — linecap パラメータを設定する
- cpdf_setlinejoin — linejoin パラメータを設定する
- cpdf_setlinewidth — 線幅を設定する
- cpdf_setmiterlimit — miter のリミットを設定する
- cpdf_setrgbcolor_fill — 塗りつぶし色を RGB カラー値に設定する
- cpdf_setrgbcolor_stroke — 描画色を RGB カラー値に設定する
- cpdf_setrgbcolor — 描画色および塗りつぶし色を RGB 値に設定する
- cpdf_show_xy — 指定位置にテキストを出力する
- cpdf_show — 現在位置にテキストを出力する

- cpdf_stringwidth — 現在のフォントのテキストの幅を返す
- cpdf_stroke — パスに沿って線を描く
- cpdf_text — パラメータを元にテキストを出力する
- cpdf_translate — 座標系の原点を設定する

COM と .Net (Windows)

導入

COM は Component Object Model の略語であり、 DCE RPC (オープンスタンダード) の最上位のオブジェクト指向レイヤーです。 COM はコール手順を共通化し、あらゆる言語でコードを記述し、 (COM に対応した)他の言語で書かれたコードをコール、相互運用することを可能にします。 あらゆる言語で書くことを可能にするだけではなく、 同じ実行形式の一部となることすら不要です。 コードは、同じマシンで実行される他のプロセスのコードである DLL からロードしたり、 または、リモートマシン上の他のプロセスにあるコードを DCOM (分散 COM) で 利用することができます。 この場合、コードの中では、コンポーネントの存在する場所を意識する必要はありません。

OLE オートメーションと呼ばれる COM のサブセットがあります。 これは、COM オブジェクトに祖な結合を行うことができる COM インターフェイスを 提供します。これにより、コンパイル時にオブジェクトの動作を知ることなく、 実行時にコールを行うことができるようになります。 PHP COM 拡張モジュールは、OLE オートメーションを使用して スクリプトから互換性のあるオブジェクトを作成/コールすることができます。 技術的に述べると、 全ての COM オブジェクトが OLE 互換であるというわけではないため、 実際には、この拡張モジュールは "PHP の OLE オートメーション 拡張モジュール" と呼ばれるべきものです。

ところで、なぜ COM を使用する必要があるのでしょう? COM は、 Windows 環境でアプリケーションとコンポーネントを結び付ける代表的な手法の一つで、 COM を使用して Microsoft Word を起動し、 ドキュメントテンプレートを埋めて、Word 文書として結果を保存し、Web サイトの 訪問者に送信することができます。 また COM を使用して、ネットワークの管理タスクを処理したり IIS を設定したりすることができます。これらは最も一般的な使用法にすぎません。 COM でできることはまだまだたくさんあります。

PHP 5 以降、この拡張モジュール(とこの文書)は最初から書き直され、 古い紛らわしい部分は削除されました。さらに Microsoft により提供された COM との相互運用レイヤーを用いて .Net アセンブリのインスタンス化と生成をサポートしました。

PHP 5 におけるこの拡張モジュールの変更点の概要については、 » この文章 を参照してください。

要件

COM 関数は、Windows 版の PHP でのみ利用可能です。

.Net サポートは、PHP 5 と .Net ランタイムを必要とします。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

Windows 版の *PHP* にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

あなたには、(MS Word のような)使用する様々な COM オブジェクトのインストールを正しく 行っておく責任があります。 PHP にこれら全てをバンドルすることはできません。

foreach

PHP 5 以降、標準的な COM/OLE IEnumVariant の内容について、 PHP の foreach 命令を使用した反復処理を行うことができます。分かりやすく言うと、 これは、VB/ASP のコードで *For Each* を使用できる場所には foreach を使用できるということを意味します。

Example#1 ASP における For Each

```
<%
Set domainObject = GetObject("WinNT://Domain")
For Each obj in domainObject
   Response.Write obj.Name & "<br />"
Next
%>
```

Example#2 PHP 4 におけるwhile() ... Next()

```
<?php
$domainObject = new COM("WinNT://Domain");
while ($obj = $domainObject->Next()) {
    echo $obj->Name . "<br />";
}
?>
```

Example#3 PHP 5 における foreach

```
<?php
$domainObject  = new COM("WinNT://Domain");
foreach  ($domainObject  as $obj) {
    echo $obj->Name  . "<br  />";
}
?>
```

配列と配列形式の COM プロパティ

多くの COM オブジェクトは、プロパティを配列で公開したり 配列形式を使用してアクセスできるようにしています。 PHP 4 では、PHP の配列構文を使用してこれらのプロパティに対する読み書きが できますが、1 次元の配列のみがサポートされます。多次元のプロパティを 読み込みたい場合は、プロパティへのアクセスを関数コールに組み込んで 各パラメータを入れtうの各次元に対応させるという方法が可能ですが、 そのようなプロパティに対する書き込みの手段はありません。

PHP 5 では以下の新機能を用いることで多少ましになりました。

- 多次元配列・複数パラメータを要求する COM プロパティへの PHP の配列構文を使用したアクセス。 書き込みやプロパティの設定にもこの技法が使用可能です。
- foreach 制御構造を使用した SafeArrays ("真の" 配列) の値の取得。SafeArrays が自分自身の サイズについての情報を含んでいることからこれが可能となります。 配列形式のプロパティが IEnumVariant を実装している場合は、その プロパティに対しても foreach が使用可能です。この項目についての 詳細な情報は、COM をごらんください。

例外 (PHP 5)

COM から致命的なエラーが報告された場合、この拡張モジュールは *com_exception* クラスのインスタンスをスローします。 すべての COM 例外は *code* という定義済みの プロパティを保持しており、これは COM 操作が返す HRESULT 値に対応します。 プログラム上での例外の処理方法を決定するために、この値を使用する ことができます。

実行時設定

php.ini の設定により動作が変化します。

COM 設定オプション

名前	デフォルト	変更の範囲	変更履歴
com.allow_dcom	"0"	PHP_INI_SYSTEM	PHP 4.0.5 以降で使用可能です。
com.autoregister_typelib	"0"	PHP_INI_ALL	PHP 4 では PHP_INI_SYSTEM です。PHP 4.1.0 以降で使用可能です。
com.autoregister_verbose	"0"	PHP_INI_ALL	PHP 4 では PHP_INI_SYSTEM です。PHP 4.1.0 以降で使用可能です。
com.autoregister_casesensitive	"1"	PHP_INI_ALL	PHP 4 では PHP_INI_SYSTEM です。PHP 4.1.0 以降で使用可能です。
com.code_page	""	PHP_INI_ALL	PHP 5.0.0 以降で使用可能です。
com.typelib_file	""	PHP_INI_SYSTEM	PHP 4.0.5 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

com.allow_dcom

これを on にすると、PHP が D-COM (分散 COM) クライアントとして動作することを許可し、 PHP スクリプトがリモートサーバ上に COM オブジェクトを生成することを 許可します。

com.autoregister_typelib

これを on にすると、生成したオブジェクトのタイプライブラリから取得した 定数を PHP に登録しようと試みます。ただし、それはオブジェクトが 当該情報を取得するためのインターフェースを提供している場合のみです。 登録する定数の大文字小文字を区別するかどうかについては、COM 設定ディレクティブで 制御します。

com.autoregister_verbose

これを on にすると、オブジェクト生成時のタイプライブラリの読み込み中に 発生したすべての問題が PHP のエラー機構を用いて報告されます。 デフォルトは off で、この場合はタイプライブラリの検索や読み込みの際の エラーは一切報告されません。

com.autoregister_casesensitive

これを on にすると (デフォルト)、自動読み込みされた タイプライブラリ中に見つかった定数が、大文字小文字を区別して 登録されます。詳細は com_load_typelib() を参照ください。

com.code_page

これは、COM オブジェクトとの文字列の受け渡しに使用するデフォルトの 文字セットコードページを制御します。空の文字列が設定された場合、 PHP は **CP_ACP** が指定されたと仮定します。 これは、デフォルトのシステム ANSI コードページです。

スクリプト中のテキストがデフォルトとは異なるエンコーディング/ 文字セットを使用している場合、このディレクティブを設定することで COM クラスのコンストラクタのパラメータとして コードページを指定する必要がなくなります。 (他の PHP 設定ディレクティブとともに) このディレクティブを使用すると、 PHP スクリプトの移植可能性が悪くなることに注意しましょう。 できる限り、COM のコンストラクタにパラメータを指定する方式をとるべきです。

注意: この設定ディレクティブは PHP 5 以降で使用可能です。

com.typelib_file

このパラメータでは、起動時に読み込まれるタイプライブラリの一覧を 含むファイルへのパスを保持します。このファイル内の各行が タイプライブラリ名として扱われ、com_load_typelib() をコールした際にそれが読み込まれます。登録された定数は永続的に 保持されるので、ライブラリの読み込みは一度だけでよくなります。 タイプライブラリの名前が *#cis* あるいは *#case_insensitive* で終わる場合は、そのライブラリから読み込まれた定数は大文字小文字を区別せずに登録されます。

リソース型

この拡張モジュールでは、COM コンポーネントへの参照を定義しています。 これは、非推奨の com_load() 関数が返すものです (この関数は PHP 5 には存在しません。かわりに COM クラスを使用します)。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

CLSCTX_INPROC_SERVER (integer)
CLSCTX_INPROC_HANDLER (integer)
CLSCTX_LOCAL_SERVER (integer)
CLSCTX_REMOTE_SERVER (integer)
CLSCTX_SERVER (integer)
CLSCTX_ALL (integer)
VT_NULL (integer)
VT_EMPTY (integer)
VT_UI1 (integer)
VT_I2 (integer)
VT_I4 (integer)
VT_R4 (integer)
VT_R8 (integer)
VT_BOOL (integer)
VT_ERROR (integer)
VT_CY (integer)
VT_DATE (integer)
VT_BSTR (integer)
VT_DECIMAL (integer)
VT_UNKNOWN (integer)
VT_DISPATCH (integer)
VT_VARIANT (integer)
VT_I1 (integer)
VT_UI2 (integer)
VT_UI4 (integer)
VT_INT (integer)
VT_UINT (integer)
VT_ARRAY (integer)
VT_BYREF (integer)
CP_ACP (integer)
CP_MACCP (integer)
CP_OEMCP (integer)
CP_UTF7 (integer)
CP_UTF8 (integer)
CP_SYMBOL (integer)
CP_THREAD_ACP (integer)
VARCMP_LT (integer)

VARCMP_EQ (integer)
VARCMP_GT (integer)
VARCMP_NULL (integer)
NORM_IGNORECASE (integer)
NORM_IGNORENONSPACE (integer)
NORM_IGNORESYMBOLS (integer)
NORM_IGNOREWIDTH (integer)
NORM_IGNOREKANATYPE (integer)
NORM_IGNOREKASHIDA (integer)
DISP_E_DIVBYZERO (integer)
DISP_E_OVERFLOW (integer)
MK_E_UNAVAILABLE (integer)

参考

COM についてのより詳細な情報は » COM 仕様 を読むか、あるいは Don Box の » Yet Another COM Library (YACL) をごらんください。その他の有用な情報が、 PHP と COM の FAQ からも得られるでしょう。 MS Office アプリケーションをサーバサイドで使用しようと考えておられるなら、 » Considerations for Server-Side Automation of Office の情報も読んでおくべきでしょう。

COM

(No version information available, might be only in CVS)

COM — COM クラス

```
$obj = new COM("Application.ID")
```

説明

COM クラスにより、OLE 互換の COM オブジェクトのインスタンスを作成し、 そのメソッドをコールしたりそのプロパティにアクセスしたりすることが 可能となります。

メソッド

com **COM::COM** (string `$module_name` [, mixed `$server_name` [, int `$codepage` [, string `$typelib`]]])

COM クラスのコンストラクタ。 パラメータには以下のような意味があります。

module_name
: ロードするコンポーネントを指定する プログラム ID、クラス ID またはモニカです。 プログラムID は、通常、アプリケーションまたは DLL の名前の後に、 *Word.Application* のようにピリオドと オブジェクト名を付けたものです。 クラス ID は、指定されたクラスがユニークに定義する UUID です。 モニカは、URL スキームの考え方に似た特別な命名形式です。リソースと、 それがどのように読み込まれるかを指定します。例として、モジュール名に Word ドキュメントのフルパスを指定すると、Word を読み込んで ドキュメントに対応するオブジェクトを取得することができます。 あるいは、LDAP への ADSI インターフェースを指すモニカとして *LDAP:* を使用することが可能です。

server_name
: コンポーネントが読み込んで実行される DCOM サーバの名前です。 **NULL** の場合、オブジェクトはアプリケーションのデフォルトを 指定して実行されます。典型的なデフォルトは、ローカルマシン上で 実行させることでが、管理者によってはアプリケーションを別の マシン上で実行させるように設定しているかもしれません。 非 **NULL** 値をサーバに指定した場合、 COM 設定オプションが **TRUE** に設定されていない限り PHP はオブジェクトの読み込みを 拒否します。

  server_name が配列の場合、以下の要素 (大文字小文字を区別します!)を含まなければなりません。 これらはすべて省略可能であることに注意しましょう (とはいえ、Username および Password は両方指定する必要があります)。 もしサーバ設定を省略した場合は(上で説明した)デフォルトのサーバが 使用され、オブジェクトの生成は COM ディレクティブの影響を受けません。

DCOM サーバ名

server_name のキー	型	説明
Server	string	サーバの名前。
Username	string	接続するユーザ名。
Password	string	*Username* に対するパスワード。
Flags	integer	以下の定数のうちのひとつまたは複数の論理和。 **CLSCTX_INPROC_SERVER**、 **CLSCTX_INPROC_HANDLER**、 **CLSCTX_LOCAL_SERVER**、 **CLSCTX_REMOTE_SERVER**、 **CLSCTX_SERVER** および **CLSCTX_ALL**。指定されていない場合の デフォルト値は、*Server* を省略した場合が **CLSCTX_SERVER** でサーバ名を指定した場合は **CLSCTX_REMOTE_SERVER** です。 これらの定数の意味についての詳細な情報を得るには、Microsoft のドキュメントで CoCreateInstance について調べましょう。 通常はこれらを使用する必要はないはずです。

codepage

文字列と Unicode 文字列との相互変換に使用するコードページを指定します。 PHP の文字列と COM オブジェクトのメソッドとの受け渡しの際には、いつも 変換が行われます。PHP 5 では、コードページの扱いは面倒です。というのは、 オブジェクトだけではなくオブジェクトから返される variant にもそれが 影響するからです。 とりうる値は **CP_ACP** (システムのデフォルト ANSI コードページを使用する - このパラメータが指定されなかった場合の デフォルト)、 **CP_MACCP**、 **CP_OEMCP**、**CP_SYMBOL**、 **CP_THREAD_ACP** (現在実行中のスレッドの コードページ/ロケールを使用する)、**CP_UTF7** および **CP_UTF8** です。コードページに対応する 数値を指定することも可能です。コードページとそれに対応する数値に ついての詳細は、Microsoft のドキュメントを参照ください。

オーバーロードされたメソッド

返されるオブジェクトは、オーバーロードされたものです。つまり、 PHP 側では通常のクラスのメソッドは見えないということです。 その代わりに、プロパティやメソッドへのアクセスは COM を通じて行います。

PHP 5 以降では、参照渡しのパラメータを受け付けるメソッドを PHP が 自動検出するようになりました。それらのメソッドについては、PHP が自動的に 変数を参照渡し形式に変換します。つまり、メソッドのコールをより自然に 行えるということです。コードの中で特別な処理をする必要はありません。

PHP 4 では、パラメータを参照渡しする場合には、そのパラメータをラップする ために VARIANT クラスのインスタンスを生成する 必要があります。

疑似メソッド

PHP 5 より前のバージョンでは、以下のようなメソッド名を COM に渡すことが できず、直接 PHP で扱うことができないというあまりうれしくない出来事が ありました。PHP 5 ではこれらを解決します。スクリプトの修正方法に ついては以下の詳細を参照ください。これらのマジックメソッドの名前は 大文字小文字を区別しません。

void COM::AddRef (void)

COM オブジェクトの参照カウントを作為的に追加します。

警告

このメソッドを使用することはまずないはずです。 これは、以下で説明する Release() メソッドを論理的に補完するものとして 存在するものです。

void COM::Release (void)

COM オブジェクトの参照カウントを作為的に削除します。

警告

このメソッドを使用することはまずないはずです。 これは、COM オブジェクトが必要以上に動作を続けてしまうというバグに 対応する方法として PHP に存在するものです。

反復処理のための疑似メソッド

以下の疑似メソッドは、 com_isenum() が **TRUE** を返す場合のみ利用可能で、この場合、これらのメソッドは、 通常は COM オブジェクトにより提供される同じ名前を有する全ての メソッドを隠蔽します。これらのメソッドは PHP 5 では完全に廃止されています。 代わりに COM を使用してください。

variant **COM::All** (void)

10 の要素を保持する SafeArray を表す variant を返します。 個々の要素は empty/null の variant となります。この関数は イテレータからの全ての要素を含む配列を返すことを想定していますが、 決して完了しません。使用しないでください。

variant **COM::Next** (void)

イテレータの次の要素を表す variant を返します。要素が もうない場合には **FALSE** を返します。

variant **COM::Prev** (void)

イテレータの前の要素を現す variant を返します。要素が もうない場合には **FALSE** を返します。

void **COM::Reset** (void)

イテレータを最初の場所まで巻き戻します。

COM の例

Example#1 COM の例 (1)

```
<?php
// word を起動します
$word = new COM("word.application") or die("Unable to instantiate Word");
echo "Loaded Word, version {$word->Version}¥n";

// 前面に移動させます
$word->Visible = 1;

// 空のドキュメントを開きます
$word->Documents->Add();

// 何か複雑なことを行います
$word->Selection->TypeText("This is a test...");
$word->Documents[1]->SaveAs("Useless test.doc");

// word を閉じます
$word->Quit();

// オブジェクトを開放します
$word = null;
?>
```

Example#2 COM の例 (2)

```
<?php

$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "¥n";

for ($i=0; $i < $num_columns; $i++) {
    $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF) {
    for ($i=0; $i < $num_columns; $i++) {
        echo $fld[$i]->value . "¥t";
    }
    echo "¥n";
    $rowcount++;            // 行カウントの加算
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs = null;
$conn = null;

?>
```

DOTNET

(No version information available, might be only in CVS)

DOTNET — DOTNET クラス

```
$obj = new DOTNET("assembly", "classname")
```

説明

DOTNET クラスにより、.Net アセンブリからクラスを生成して そのメソッドをコールしたりプロパティにアクセスしたりできるようになります。

メソッド

string **DOTNET::DOTNET** (string `$assembly_name` , string `$class_name` [, int `$codepage`])

DOTNET クラスのコンストラクタです。 読み込むアセンブリ名を *assembly_name* で指定し、アセンブリから生成するクラスを *class_name* で指定し

ます。unicode 文字列の変換に使用するため、オプションの パラメータ *codepage* を設定することができます。 コードページの詳細については COM クラスを 参照ください。

返されるオブジェクトはオーバーロードされています。つまり、通常の クラスのような固定されたメソッドは存在しないということです。 そのかわりに、プロパティやメソッドへのアクセスは COM を通じて DOTNET に渡されます。言い換えれば、.Net ランタイムが提供する 機能によって .Net オブジェクトが COM 連携用にマップされているという ことです。

DOTNET オブジェクトが作成されると、PHP はそれを他の COM オブジェクトと同等のものとして扱い、同様の規則が適用されます。

Example#1 DOTNET の例

```
<?php
   $stack = new DOTNET("mscorlib", "System.Collections.Stack");
  $stack->Push(".Net");
  $stack->Push("Hello ");
   echo $stack->Pop() . $stack->Pop();
?>
```

> **注意**: この機能を使用するには、Web サーバに .Net ランタイムを インストールする必要があります。

VARIANT

(No version information available, might be only in CVS)

VARIANT — VARIANT クラス

```
$vVar = new VARIANT($var)
```

説明

VARIANT は、PHP の zval と同等の COM の値です。構造体形式になっており、 異なる型の値を含めることが可能です。COM 拡張モジュールが提供する VARIANT クラスによって、PHP と COM の間のデータの受け渡しを制御できるように なります。

メソッド

object **VARIANT::VARIANT** ([mixed `$value` [, int `$type` [, int `$codepage`]]])

VARIANT クラスのコンストラクタ。パラメータは以下のとおりです。

value
: 初期値。省略したり **NULL** を設定した場合には、VT_EMPTY オブジェクトが作成されます。

type
: VARIANT オブジェクトの content 型を指定します。使用可能な値は、 **VT_XXX** COM のうちのひとつです。 PHP 5 より前のバージョンでは、*type* に **VT_BYREF** を加えることで PHP から variant オブジェクトに参照渡しをさせることが可能でした。PHP 5 では、 この方法はサポートされていません。そのかわりに、PHP 5 では パラメータが参照渡しされた場合にそれを自動的に認識します。 VARIANT オブジェクトとして渡す必要さえありません。 VARIANT 型についてのその他の情報については、MSDN ライブラリを参照ください。

codepage
: 文字列を unicode に変換する際に使用するコードページを指定します。 詳細な情報については、 COM クラスの同名のパラメータを参照ください。

PHP 5 より前のバージョンでは、VARIANT クラスのインスタンスには 多くの(ドキュメント化されていない)仮想プロパティが存在します。 これらのプロパティは PHP 5 ですべて削除され、かわりにより自然な構文で アクセスできるようになっています。これらの違いについて、以下の例で 説明します。

Example#1 PHP 4.x 形式 での Variant の例

```
<?php
$v = new VARIANT(42);
print "The type is " . $v->type . "<br/>";
print "The value is " . $v->value . "<br/>";
?>
```

Example#2 PHP 5 形式での Variant の例

```
<?php
$v = new VARIANT(42);
print "The type is " . variant_get_type($v) . "<br/>";
print "The value is " . $v . "<br/>";
?>
```

このように変更された理由は、内部的には、COM 拡張モジュールが VARIANT、COM および DOTNET クラスを同一のものとして扱い、これらの クラスのプロパティやメンバへのアクセスは(何のインターフェースも 経由せず)直接 COM とやり取りを行うようにするという設計思想によるものです。新しい構文は、より自然でより手軽です。また、削除された仮想プロパティの ほとんどは、PHP のコンテキストにおいては何の意味も持たないものでした。

> **注意**: PHP 5 では、VARIANT を扱うためのよりシンプルな手段をとっています。 値を返したり variant プロパティを取得したりする際に variant が PHP の値に変換されるのは、情報を失うことなく型変換ができる型が 存在する場合に限られます。それ以外の場合は、結果は VARIANT クラスの インスタンスとして返されます。明示的にキャスト演算子を指定することで variant を PHP のネイティブ型として扱うことが可能です。また、 print() を使用すると、暗黙のうちに値が文字列に 変換されます。variant に対する計算のためのさまざまな関数が用意されており、 型変換でデータを失うリスクを犯さなくてもそのままの形式で計算を 行うことが可能です。

variant_get_type() も参照ください。

---

com_addref

(PHP 4 >= 4.0.7)

com_addref — コンポーネントの参照カウンタを増やす [非推奨]

説明

void **com_addref** (void)

コンポーネントの参照カウンタを増やします。

警告

この関数を使用する必要は決してないはずです。

---

com_create_guid

(PHP 5)

com_create_guid — グローバルユニーク ID (GUID) を生成する

説明

string **com_create_guid** (void)

Globally Unique Identifier (GUID) を生成し、文字列で返します。 GUID は DCE UUID と同様の方法で生成されますが、Microsoft の 規約により、GUID が中括弧で囲まれるという点が異なります。

PECL uuid 拡張モジュールの **uuid_create()** も参照ください。

---

com_event_sink

(PHP 4 >= 4.2.0, PHP 5)

com_event_sink — COM オブジェクトのイベントを PHP オブジェクトに接続する

説明

bool **com_event_sink** (variant `$comobject` , object `$sinkobject` [, mixed `$sinkinterface`])

comobject が生成したイベントを PHP オブジェクト *sinkobject* に通知するよう、COM に指示します。 PHP は *comobject* に関連するタイプライブラリで 指定されたデフォルトのディスパッチインターフェースを使用しようと しますが、使用させたいインターフェース名を *sinkinterface* に指定することで、それを 上書きすることが可能です。

sinkobject には、要求されるディスパッチ インターフェースと同じ名前のメソッドを持つクラスのインスタンスを 指定する必要があります。この要求を満たすクラスを作成するために、 com_print_typeinfo() を使用することができます。

この機能を利用する際には注意しましょう。もし以下の例のようなことを 行いたいのであれば、Web サーバ上でそれを行うことには まったく意味がありません。

Example#1 COM イベントシンクの例

```
<?php
class IEEventSinker  {
  var $terminated  = false;

  function ProgressChange($progress, $progressmax) {
```

```
    echo "ダウンロードの進行状況: $progress / $progressmax\n";
  }

  function DocumentComplete(&$dom, $url) {
    echo "ドキュメント $url 完了\n";
  }

  function OnQuit() {
    echo "終了!\n";
    $this->terminated = true;
  }
}
$ie = new COM("InternetExplorer.Application");
// PHP 5 では & が必要ないことに注意!
$sink =& new IEEventSinker();
com_event_sink($ie, $sink, "DWebBrowserEvents2");
$ie->Visible = true;
$ie->Navigate("http://www.php.net");
while(!$sink->terminated) {
  com_message_pump(4000);
}
$ie = null;
?>
```

com_print_typeinfo()、 com_message_pump() も参照ください。

---

com_get_active_object

(PHP 5)

com_get_active_object — すでに実行中の COM オブジェクトのインスタンスへのハンドルを返す

説明

variant **com_get_active_object** (string `$progid` [, int `$code_page`])

com_get_active_object() は、新しい COM オブジェクトのインスタンスを作成することに 似ています。しかし、オブジェクトがすでに実行中の場合には そのオブジェクトが返されるという点が違います。 OLE アプリケーションは、既知のアプリケーションを一度だけ起動させる ことを許可するために、Running Object Table というものを使用します。 この関数は、実行中のインスタンスのハンドルを取得するために COM ライブラリ関数 GetActiveObject() を公開します。

progid は、アクセスしたいオブジェクトの ProgID あるいは CLSID (たとえば *Word.Application*) である必要があります。*code_page* は、 COM クラスの場合と同様の働きをします。

要求されたオブジェクトが実行中の場合は、他の COM オブジェクトと 同様にスクリプトへ返されます。そうでない場合は *com_exception* が発生します。この関数が失敗する 理由はさまざまなものが考えられますが、最も一般的なのは オブジェクトがまだ起動していないことです。そのような場合、 例外のエラーコードは **MK_E_UNAVAILABLE** となります。例外オブジェクトの *getCode* メソッドを使用することで、例外コードの内容を調べることが可能です。

警告

Web 環境で **com_get_active_object()** を使用する ことは、あまり良い考えではありません。ほとんどの COM/OLE アプリケーションは複数のクライアントから同時に利用されることを 考慮していないのです。(あの) Microsoft Office でさえもです! この件に関する一般的な問題についての詳細な情報は、» Considerations for Server-Side Automation of Office を参照ください。

---

com_get

(PHP 4)

com_get — COM コンポーネントのプロパティの値を得る [非推奨]

説明

mixed **com_get** (resource `$com_object` , string `$property`)

com_object が指す COM コンポーネントの *property* の値を返します。エラー時に **FALSE** を返します。

Example#1 com_get() を使用せず、かわりにオブジェクト指向の構文を使用する

```
<?php
// こちらを使用します
$var = $obj->property;
// こちらは推奨しません
$var = com_get($obj, 'property');
?>
```

> **注意**: この関数は PHP 5 には存在しません。そのかわりに、標準的でより自然な オブジェクト指向の構文を使用してプロパティへのアクセスやメソッドの コールを行うべきです。

---

com_invoke

(PHP 4)

com_invoke — COM コンポーネントのメソッドをコールする [非推奨]

説明

mixed **com_invoke** (resource `$com_object` , string `$function_name` [, mixed `$function_parameters`])

com_invoke() は、 *com_object* が指す COM コンポーネントの メソッド *function_name* をコールします。 **com_invoke()** はエラーの場合に **FALSE** を返し、 成功時に *function_name* の返り値を返します。 追加パラメータ *function_parameters* の値が メソッド *function_name* に渡されます。

Example#1 com_invoke() を使用せず、かわりにオブジェクト指向の構文を使用する

```
<?php
// こちらを使用します
$val = $obj->method($one, $two);
// こちらは推奨しません
$val = com_invoke($obj, 'method', $one, $two);
?>
```

> **注意**: この関数は PHP 5 には存在しません。そのかわりに、標準的でより自然な オブジェクト指向の構文を使用してプロパティへのアクセスやメソッドの コールを行うべきです。

---

com_isenum

(PHP 4 >= 4.0.7)

com_isenum — COM オブジェクトが IEnumVariant インターフェースを実装しているかどうかを 示す [非推奨]

説明

bool **com_isenum** (variant `$com_module`)

COM オブジェクトから *Next()* メソッドによって 値を列挙することが可能かどうかを調べます。可能である場合に **TRUE**、可能でない場合に **FALSE** を返します。 このメソッドの詳細については、COM クラスを 参照ください。

> **注意**: この関数は PHP 5 には存在しません。COM オブジェクトから値を順に 取り出すには、より自然な方法である foreach 文を使用します。 詳細は COM を参照ください。

---

com_load_typelib

(PHP 4 >= 4.0.7, PHP 5)

com_load_typelib — タイプライブラリを読み込む

説明

bool **com_load_typelib** (string `$typelib_name` [, bool `$case_insensitive`])

タイプライブラリを読み込んで定数をエンジン内に登録し、それらが define() を使用して定義されているかのように 扱えるようにします。 *case_insensitive* は、 define() 関数の同名のパラメータと同じ働きをします。

typelib_name は、以下のいずれかの形式となります。

- *.tlb* ファイル、あるいはタイプライブラリを含む 実行モジュールのファイル名。
- タイプライブラリの GUID の後にバージョン番号を続ける。たとえば *{00000200-0000-0010-8000-00AA006D2EA4},2,0* のような形式。
- タイプライブラリ名。たとえば *Microsoft OLE DB ActiveX Data Objects 1.0 Library* のような形式。

PHP は、この順序でタイプライブラリ名の解決を試みます。リストの下の ほうにいくほど検索処理のコストが高くなります。タイプライブラリ名に よ

る検索は、一致するものが見つかるまでレジストリを物理的に列挙していく という方法をとっています。

あまり融通の利く方法ではありませんが、 COM 設定を利用して定数の 事前読み込みと登録をすませるほうがずっと効率的であることに 注意しましょう。

COM を有効にすると、 PHP がインスタンス化した COM オブジェクトに関連付けられている定数を 自動的に登録しようと試みます。これは COM オブジェクト自身の提供する インターフェースに依存し、常に可能であるとは限りません。

---

com_load

(PHP 4)

com_load — COM コンポーネントへの新規リファレンスを作成する [非推奨]

説明

resource **com_load** (string `$module_name` [, string `$ server_name` [, int `$ codepage`]])

new 演算子を使用して COM クラスのインスタンスを生成するのと同等の処理を行います。この関数を 使用するかわりに、クラスのインスタンスを生成する方式を使用すべきです。

Example#1 com_load() を使用せず、かわりにオブジェクト指向の構文を使用する

```
<?php
// こちらを使用します
$obj = new COM($module);
// こちらは推奨しません
$obj = com_load($module);
?>
```

> **注意**: この関数は PHP 5 には存在しません。かわりに COM クラスを使用してください。

---

com_message_pump

(PHP 4 >= 4.2.0, PHP 5)

com_message_pump — COM メッセージを処理し、timeoutms ミリ秒の間待つ

説明

bool **com_message_pump** ([int `$timeoutms`])

この関数は、*timeoutms* ミリ秒が経過するか キューにメッセージが到着するまで待機します。タイムアウトの前に メッセージが到着した場合は、そのメッセージが処理されて 関数は **TRUE** を返します。メッセージを処理することなく タイムアウトが発生した場合は、この関数は **FALSE** を返します。*timeoutms* の値を指定しなかった場合は 0 であるとみなされます。0 という値は、一切待機しないことを 意味します。処理待ち状態のメッセージがあればそれを処理し、 なければこの関数はすぐに **FALSE** を返します。

この関数の目的は、別々の場所にある COM のコールを振り分け、同期の 問題をうまく処理することです。これにより、発生するイベントを 効率的にスクリプトで待ち受けられるようになり、バックグラウンドで 動作している他のコードやイベントも処理できるようになります。 COM オブジェクトのイベントを使用している間は、 com_event_sink() 関数の例で示したように この関数をループ内で使用すべきです。

---

com_print_typeinfo

(PHP 4 >= 4.2.0, PHP 5)

com_print_typeinfo — ディスパッチインターフェースのために、PHP のクラス定義を出力する

説明

bool **com_print_typeinfo** (object `$comobject` [, string `$dispinterface` [, bool `$wantsink`]])

この関数の目的は、イベントシンクに使用するスケルトンクラスの作成を 支援することです。 You may also use it to generate a dump of any COM object, provided that it supports enough of the introspection interfaces, and that you know the name of the interface you want to display.

comobject は、COM オブジェクトのインスタンスか あるいはタイプライブラリの名前 (com_load_typelib() の規則にしたがって名前解決されます) のいずれかです。 *dispinterface* は、結果を表示したい インターフェースで、このインターフェースは IDispatch を 継承したものです。*wantsink*

が **TRUE** の場合、対応するシンクインターフェースが代わりに表示されます。

com_event_sink()、 com_load_typelib() も参照ください。

---

com_propget

(PHP 4)

com_propget — com_get() のエイリアス

説明

この関数は次の関数のエイリアスです。 com_get().

> **注意**: この関数は PHP 5 には存在しません。そのかわりに、標準的でより自然な オブジェクト指向の構文を使用してプロパティへのアクセスやメソッドの コールを行うべきです。

---

com_propput

(PHP 4)

com_propput — com_set() のエイリアス

説明

この関数は次の関数のエイリアスです。 com_set().

> **注意**: この関数は PHP 5 には存在しません。そのかわりに、標準的でより自然な オブジェクト指向の構文を使用してプロパティへのアクセスやメソッドの コールを行うべきです。

---

com_propset

(PHP 4)

com_propset — com_set() のエイリアス

説明

この関数は次の関数のエイリアスです。 com_set().

> **注意**: この関数は PHP 5 には存在しません。そのかわりに、標準的でより自然な オブジェクト指向の構文を使用してプロパティへのアクセスやメソッドの コールを行うべきです。

---

com_release

(PHP 4 >= 4.0.7)

com_release — コンポーネントリファレンスカウンタを減らす [廃止]

説明

void **com_release** (void)

コンポーネントリファレンスカウンタを減らします。

警告

この関数を使用する必要はありません。

> **注意**: この関数は、PHP 5では廃止されています。

---

com_set

(PHP 4)

com_set — COM コンポーネントのプロパティに値を代入する

説明

void **com_set** (resource `$com_object` , string `$property` , mixed `$value`)

com_object が指す COM コンポーネントの *property* の値を設定します。成功時に 新規に設定された値、エラーの場合に **FALSE** を返します。

Example#1 com_set() を使用せず、かわりにオブジェクト指向の構文を使用する

```
<?php
// こちらを使用します
$obj->property  = $value;
// こちらは推奨しません
com_set($obj, 'property', $value);
?>
```

> **注意**: この関数は PHP 5 には存在しません。そのかわりに、標準的でより自然な オブジェクト指向の構文を使用してプロパティへのアクセスやメソッドの コールを行うべきです。

---

variant_abs

(PHP 5)

variant_abs — variant の絶対値を返す

説明

mixed **variant_abs** (mixed `$val`)

val の絶対値を返します。

abs() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_add

(PHP 5)

variant_add — 2 つの variant 値を「加算」し、結果を返す

説明

mixed **variant_add** (mixed `$left` , mixed `$right`)

(MSDN ライブラリによる) 以下の規則に従い、 *left* を *right* に加算します。この規則は Visual Basic の規則と一致します。

Variant の加算規則

もし〜なら	このようにします
両者ともに文字列型	連結
一方が文字列型で、もう一方が文字	追加
一方が数値で、もう一方が文字列	追加
両者ともに数値	加算
どちらか一方が NULL	NULL が返される
両者ともに空の値	Integer 型が返される

variant_sub() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_and

(PHP 5)

variant_and — 2 つの variant の論理積を計算し、結果を返す

説明

mixed **variant_and** (mixed `$left` , mixed `$right`)

以下の表に基づいた論理 AND 処理を行います。通常の AND 操作とは 少し違いがあることに注意しましょう。

Variant の AND 規則

left	*right*	結果
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
TRUE	NULL	NULL
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE
FALSE	NULL	FALSE
NULL	TRUE	NULL
NULL	FALSE	FALSE
NULL	NULL	NULL

variant_or() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_cast

(PHP 5)

variant_cast — variant を、別の型の新しい variant に変換する

説明

variant **variant_cast** (variant `$variant` , int `$type`)

この関数は、*variant* のコピーを作成して、 それを指定した *type* に強制的に変換します。 *type* は、定数 **VT_XXX** の中のひとつでなければなりません。

この関数は、COM ライブラリの VariantChangeType() をラップしたものです。 詳細な情報は MSDN ライブラリを参照ください。

variant_set_type() も参照ください。

---

variant_cat

(PHP 5)

variant_cat — 2 つの variant 値を連結し、その結果を返す

説明

mixed **variant_cat** (mixed `$left` , mixed `$right`)

left と *right* を連結し、その結果を返します。

文字列の連結演算子については 文字列演算子 も参照ください。この関数は、理論上は *$left* . *$right* と等価です。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_cmp

(PHP 5)

variant_cmp — 2 つの variant を比較する

説明

int **variant_cmp** (mixed `$left` , mixed `$right` [, int `$lcid` [, int `$flags`]])

left と *right* を比較し、以下のいずれかの値を返します。

Variant の比較結果

値	意味
VARCMP_LT	*left* は *right* より小さい
VARCMP_EQ	*left* は *right* と等しい
VARCMP_GT	*left* は *right* より大きい
VARCMP_NULL	*left* と *right* のいずれか、あるいは両方が **NULL**

この関数はスカラー値のみを比較します。配列や variant レコードは 比較しません。

lcid は、文字列の比較に使用する 有効なロケール識別子です (これは文字列の比較に影響します)。 *flags* は、以下のひとつあるいは複数の値を OR で組み合わせたもので、文字列の比較に影響します。

Variant の比較フラグ

値	意味
NORM_IGNORECASE	大文字小文字を区別せずに比較する
NORM_IGNORENONSPACE	非スペーシング文字を無視する
NORM_IGNORESYMBOLS	記号を無視する
NORM_IGNOREWIDTH	全角半角を区別しない
NORM_IGNOREKANATYPE	ひらがなカタカナを区別しない
NORM_IGNOREKASHIDA	アラビア語の kashida 文字を無視する

注意: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_date_from_timestamp

(PHP 5)

variant_date_from_timestamp — Unix タイムスタンプを、日付形式の variant で返す

説明

variant **variant_date_from_timestamp** (int `$timestamp`)

unix タイムスタンプ値 *timestamp* を **VT_DATE** 型の variant に変換します。 これにより、PHP の unix 的な部分と COM とのやりとりが容易になります。

この関数の逆の処理については variant_date_to_timestamp() を、そして mktime()、time() も参照ください。

---

variant_date_to_timestamp

(PHP 5)

variant_date_to_timestamp — 日付/時刻の variant 値を Unix タイムスタンプに変換する

説明

int **variant_date_to_timestamp** (variant `$variant`)

variant を、**VT_DATE** (あるいはそれに似た型) から Unix タイムスタンプに変換します。 これにより、PHP の Unix 的な部分と COM とのやりとりが容易になります。

この関数の逆の処理については variant_date_from_timestamp() を、そして date()、strftime() も参照ください。

---

variant_div

(PHP 5)

variant_div — 2 つの variant の除算結果を返す

説明

mixed **variant_div** (mixed `$left` , mixed `$right`)

left を *right* で割り、以下の規則にしたがってその結果を返します。

Variant の除算規則

もし〜なら	このようにします
両者ともに文字列型・日付型・文字・論理型	Double 型を返す
一方が文字列型で、もう一方が文字	除算結果を double 型で返す
一方が数値で、もう一方が文字列	除算結果を double 型で返す
両者ともに数値	除算結果を double 型で返す
どちらか一方が NULL	NULL が返される
right が空の値で *left* がそれ以外の型	com_exception コード **DISP_E_DIVBYZERO** がスローされる
left が空の値で *right* がそれ以外の型	double 型の 0 が返される
両者ともに空の値	com_exception コード **DISP_E_OVERFLOW** がスローされる

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_eqv

(PHP 5)

variant_eqv — 2 つの variant のビット値が等しいかどうかを調べる

説明

mixed **variant_eqv** (mixed `$left` , mixed `$right`)

left の各ビットが *right* の対応するビットに等しい場合に **TRUE** を、それ以外の場合に **FALSE** を返します。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_fix

(PHP 5)

variant_fix — variant の整数部を返す

説明

mixed **variant_fix** (mixed `$variant`)

variant が負の数の場合、variant の値以上の 最初の整数値を返します。それ以外の場合は *variant* の値の整数部分を返します。

variant_int()、 variant_round()、floor()、 ceil()、 round() も参照ください。

警告

このドキュメントは MSDN ドキュメントに基づいています。 この関数は variant_int() とまったく同じか、 そうでなければ MSDN ドキュメントに間違いがあると思われます。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_get_type

(PHP 5)

variant_get_type — variant オブジェクトの型を返す

説明

int **variant_get_type** (variant $variant)

この関数は、*variant* の型を表す整数値を 返します。*variant* は COM、 DOTNET あるいは VARIANT クラスのインスタンスです。 返される値は、**VT_XXX** 定数のいずれかと比較する ことが可能です。

COM および DOTNET オブジェクトの返す値は、通常 **VT_DISPATCH** です。これらのクラスに対して この関数が動作する理由は、単に COM および DOTNET が VARIANT を 継承しているからというだけのことです。

PHP のバージョン 5 より前では、この情報は VARIANT クラスの インスタンスからしか取得できません。そのためには 擬似プロパティ *type* の値を読み取ります。 これについてのより詳細な情報は VARIANT クラスを参照ください。

---

variant_idiv

(PHP 5)

variant_idiv — variants を整数に変換し、除算の結果を返す

説明

mixed **variant_idiv** (mixed $left , mixed $right)

left および *right* を 整数値に変換し、以下の規則に基づいて整数の除算を行います。

Variant の整数除算規則

もし〜なら	このようにします
両者ともに文字列型・日付型・文字・論理型	除算結果を integer 型で返す
一方が文字列型で、もう一方が文字	除算
一方が数値で、もう一方が文字列	除算
両者ともに数値	除算
どちらか一方が NULL	NULL が返される
両者ともに空の値	com_exception コード **DISP_E_DIVBYZERO** がスローされる

variant_div() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_imp

(PHP 5)

variant_imp — 2 つの variant のビット implication を行う

説明

mixed **variant_imp** (mixed $left , mixed $right)

以下の表に基づき、ビットの implication 操作を行います。

Variant Implication テーブル

left	*right*	結果
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	NULL	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	TRUE
FALSE	NULL	TRUE
NULL	TRUE	TRUE
NULL	FALSE	NULL
NULL	NULL	NULL

注意: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_int

(PHP 5)

variant_int — variant の整数部を返す

説明

mixed **variant_int** (mixed `$variant`)

variant が負の数の場合、variant の値以上の 最初の整数値を返します。それ以外の場合は *variant* の値の整数部分を返します。

variant_fix()、 variant_round()、floor()、 ceil()、 round() も参照ください。

注意: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_mod

(PHP 5)

variant_mod — 2 つの variant の除算を行い、剰余を返す

説明

mixed **variant_mod** (mixed `$left` , mixed `$right`)

left を *right* で除算し、剰余を返します。

variant_div()、 variant_idiv() も参照ください。

注意: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

variant_mul

(PHP 5)

variant_mul — 2 つの variant の乗算を行い、その結果を返す

説明

mixed **variant_mul** (mixed `$left` , mixed `$right`)

以下の規則に基づいて *left* と *right* の乗算を行い、結果を返します。

Variant の乗算規則

もし〜なら	このようにします
両者ともに文字列型・日付型・文字・論理型	乗算
一方が文字列型で、もう一方が文字	乗算
一方が数値で、もう一方が文字列	乗算
両者ともに数値	乗算
どちらか一方が NULL	NULL が返される
両者ともに空の値	空の文字列が返される

論理型の値は、**FALSE** が -1 に、そして **TRUE** が 0 に変換されます。

variant_div()、 variant_idiv() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

variant_neg

(PHP 5)

variant_neg — variant の論理否定演算を行う

説明

mixed **variant_neg** (mixed `$variant`)

variant の論理否定演算を行い、その結果を 返します。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

variant_not

(PHP 5)

variant_not — variant のビット否定演算を行う

説明

mixed **variant_not** (mixed $variant)

variant のビット否定演算を行い、その 結果を返します。*variant* が **NULL** の場合、 結果も **NULL** となります。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_or

(PHP 5)

variant_or — 2 つの variant の論理和を計算する

説明

mixed **variant_or** (mixed $left , mixed $right)

以下の表に基づいた論理 OR 処理を行います。通常の OR 操作とは 少し違いがあることに注意しましょう。

Variant の OR 規則

left	*right*	結果
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	NULL	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE
FALSE	NULL	NULL
NULL	TRUE	TRUE
NULL	FALSE	NULL
NULL	NULL	NULL

variant_and()、 variant_xor() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_pow

(PHP 5)

variant_pow — 2 つの variant の累乗計算を行い、その結果を返す

説明

mixed **variant_pow** (mixed $left , mixed $right)

left の *right* 乗を返します。

pow() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照

ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_round

(PHP 5)

variant_round — 指定した桁で variant を丸める

説明

mixed **variant_round** (mixed `$variant` , int `$decimals`)

variant を、 *decimals* で指定した桁で丸めた結果を返します。

round() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_set_type

(PHP 5)

variant_set_type — variant を「その場で」別の型に変換する

説明

void **variant_set_type** (variant `$variant` , int `$type`)

この関数は variant_cast() に似ていますが、 「その場で」変換されるという点が異なります。新しい variant は 作成されません。この関数に渡すパラメータの意味は variant_cast() の場合と同じです。

variant_cast() も参照ください。

---

variant_set

(PHP 5)

variant_set — variant オブジェクトに新しい値を代入する

説明

void **variant_set** (variant `$variant` , mixed `$value`)

value を variant に変換し、それを *variant* オブジェクトに代入します。 新しい variant オブジェクトは作成されず、 *variant* の元の値は開放されます。

---

variant_sub

(PHP 5)

variant_sub — 左の variant から右の variant を引き、その結果を返す

説明

mixed **variant_sub** (mixed `$left` , mixed `$right`)

以下の規則に基づいて、*left* から *right* を引きます。

Variant の減算規則

もし〜なら	このようにします
両者ともに文字列型	減算
一方が文字列型で、もう一方が文字	減算
一方が数値で、もう一方が文字列	減算
両者ともに数値	減算
どちらか一方が NULL	NULL が返される
両者ともに空の値	空の文字列が返される

variant_add() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

---

variant_xor

(PHP 5)

variant_xor — 2 つの variant の排他的論理和を計算する

説明

mixed **variant_xor** (mixed `$left` , mixed `$right`)

以下の表に基づいて排他的論理和を計算します。

Variant の XOR 規則

left	*right*	結果
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE
NULL	NULL	NULL

variant_and()、 variant_or() も参照ください。

> **注意**: variant に関するすべての計算関数では、関数のパラメータとして PHP の ネイティブ型 (整数、文字列、浮動小数点数、論理型あるいは **NULL**) か COM・VARIANT・DOTNET クラスのインスタンスを指定可能です。 PHP のネイティブ型は、VARIANT クラスの コンストラクタと同様の規則を使用して variant に変換されます。 COM および DOTNET オブジェクトは、デフォルトのプロパティの値を variant 値として使用します。
> variant の計算関数は、COM ライブラリの同名の関数のラッパーです。 これらの関数についての詳細な情報は MSDN ライブラリを参照ください。 一部の関数は、PHP での関数名が少し異なります。例えば PHP での variant_add() に対応する関数は、MSDN ドキュメントでは VarAdd() となります。

目次

- com_invoke — COM コンポーネントのメソッドをコールする [非推奨]
- com_isenum — COM オブジェクトが IEnumVariant インターフェースを実装しているかどうかを 示す [非推奨]
- com_load_typelib — タイプライブラリを読み込む
- com_load — COM コンポーネントへの新規リファレンスを作成する [非推奨]
- com_message_pump — COM メッセージを処理し、timeoutms ミリ秒の間待つ
- com_print_typeinfo — ディスパッチインターフェースのために、PHP のクラス定義を出力する
- com_propget — com_get のエイリアス
- com_propput — com_set のエイリアス
- com_propset — com_set のエイリアス
- com_release — コンポーネントリファレンスカウンタを減らす [廃止]
- com_set — COM コンポーネントのプロパティに値を代入する
- variant_abs — variant の絶対値を返す
- variant_add — 2 つの variant 値を「加算」し、結果を返す
- variant_and — 2 つの variant の論理積を計算し、結果を返す
- variant_cast — variant を、別の型の新しい variant に変換する
- variant_cat — 2 つの variant 値を連結し、その結果を返す
- variant_cmp — 2 つの variant を比較する
- variant_date_from_timestamp — Unix タイムスタンプを、日付形式の variant で返す
- variant_date_to_timestamp — 日付/時刻の variant 値を Unix タイムスタンプに変換する
- variant_div — 2 つの variant の除算結果を返す
- variant_eqv — 2 つの variant のビット値が等しいかどうかを調べる
- variant_fix — variant の整数部を返す
- variant_get_type — variant オブジェクトの型を返す
- variant_idiv — variants を整数に変換し、除算の結果を返す
- variant_imp — 2 つの variant のビット implication を行う
- variant_int — variant の整数部を返す
- variant_mod — 2 つの variant の除算を行い、剰余を返す
- variant_mul — 2 つの variant の乗算を行い、その結果を返す
- variant_neg — variant の論理否定演算を行う
- variant_not — variant のビット否定演算を行う
- variant_or — 2 つの variant の論理和を計算する
- variant_pow — 2 つの variant の累乗計算を行い、その結果を返す
- variant_round — 指定した桁で variant を丸める
- variant_set_type — variant を「その場で」別の型に変換する
- variant_set — variant オブジェクトに新しい値を代入する
- variant_sub — 左の variant から右の variant を引き、その結果を返す
- variant_xor — 2 つの variant の排他的論理和を計算する

クラック関数 (Crack)

導入

このモジュールの関数により、パスワードの '強度' を試すための CrackLib ライブラリが使用可能となります。パスワードの '強度' は、長さ、大文字/小文字の使用で確認され、指定した CrackLib の辞書を用いて確認されます。 CrackLib は、パスワードを '強化する' ために有用な統計情報も出力します。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.0.0.

要件

CrackLib に関するより詳細な情報は、 » http://sourceforge.net/projects/cracklib にあります。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/crack.

PHP 4 の場合、この PECL 拡張モジュールのソースは、PHP のソースの *ext/* ディレクトリ、または上の PECL リンクで入手可能です。 これらの関数を使用するためには、設定オプション --with-crack[=DIR] を用いて Crackサポートを有効にしてPHPを コンパイルする必要があります。

Windowsユーザは、これらの関数を使用するために、 *php.ini* の中に *php_crack.dll* を 追加します。 PHP 4 の場合、この DLL は PHP の Windows ダウンロードバイナリの *extensions/* ディレクトリにあります。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

php.ini の設定により動作が変化します。

Crack設定オプション

名前	デフォルト	変更可能	変更履歴
crack.default_dictionary	NULL	PHP_INI_PERDIR	crack <= 0.2 では PHP_INI_SYSTEM。PHP 4.0.5 から利用可能です。PHP 5.0.0 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

CrackLib 拡張モジュールでは、crack_opendict() が返すディレクトリリソース識別子を定義しています。

定義済み定数

定数は定義されていません。

例

以下の例は、CrackLib 辞書をオープンする方法を示しており、 指定したパスワードを試験し、解析メッセージを取得し、辞書を閉じます。

Example#1 CrackLib の例

```
<?php
// CrackLib 辞書をオープンする
$dictionary = crack_opendict('/usr/local/lib/pw_dict')
      or die('Unable to open CrackLib dictionary');

// パスワードチェックを行う
$check = crack_check($dictionary, 'gx9A2s0x');

// メッセージを取得する
$diag = crack_getlastmessage();
echo $diag; // 'strong password'

// 辞書を閉じる
crack_closedict($dictionary);
?>
```

> **注意**: crack_check() は **TRUE** を返し、 crack_getlastmessage() は 'strong password' を返します。

---

crack_check

(PHP 4 >= 4.0.5, PECL crack:0.1-0.4)

crack_check — 指定したパスワードに関して強度チェックを行う

辞書

bool **crack_check** (resource `$dictionary` , string `$password`)
bool **crack_check** (string `$password`)

指定した辞書を用いて指定したパスワードの強度チェックを行います。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

`dictionary`

crackライブラリの辞書。指定しない場合は、直近にオープンされた辞書が 使用されます。

password

チェックされるパスワード

返り値

password が強い場合に、 **TRUE**、それ以外の場合に **FALSE** を返します。

---

crack_closedict

(PHP 4 >= 4.0.5, PECL crack:0.1-0.4)

crack_closedict — オープンされているCrackLib辞書を閉じる

辞書

bool **crack_closedict** ([resource `$dictionary`])

crack_closedict()は、指定した *dictionary* IDを閉じます。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

dictionary

クローズする辞書。 指定されない場合、カレントの辞書がクローズされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- crack_opendict()

---

crack_getlastmessage

(PHP 4 >= 4.0.5, PECL crack:0.1-0.4)

crack_getlastmessage — 直近の強度チェックからのメッセージを返す

説明

string **crack_getlastmessage** (void)

crack_getlastmessage() は、 直近の強度チェックからのメッセージを返します。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

返り値

直近の強度チェックによるメッセージ、あるいはチェックがまだ行われていない場合は **FALSE** を返します。

メッセージは以下のいずれかです。

- *it's WAY too short*
- *it is too short*
- *it does not contain enough DIFFERENT characters*
- *it is all whitespace*
- *it is too simplistic/systematic*

- *it looks like a National Insurance number.*
- *it is based on a dictionary word*
- *it is based on a (reversed) dictionary word*
- *strong password*

参考

- crack_check()

---

crack_opendict

(PHP 4 >= 4.0.5, PECL crack:0.1-0.4)

crack_opendict — 新規CrackLib辞書をオープンする

説明

resource **crack_opendict** (string `$dictionary`)

crack_opendict() は、指定した CrackLib *`dictionary`* をオープンします。この辞書は、 crack_check()で使用されます。

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

> **注意**: 1度にオープンできる辞書は1つだけです。

パラメータ

`dictionary`

> Cracklib辞書のパス。

返り値

成功時に辞書のリソースID、失敗時に **FALSE** 。

参考

- crack_check()
- crack_closedict()

目次

文字型 (ctype) 関数

導入

以下の関数は、現在のロケール(setlocale()も参照) に基づき文字または文字列がある文字クラスに含まれるかどうかを調べます。

整数の引数を指定してコールした場合、これらの関数は、*ctype.h* に記述された C の同名の関数と全く同様に動作します。 これは、256 より小さな整数が指定された場合、 指定した範囲 (数値は 0x30-0x39) に収まっているかどうかを 調べるために、そのアスキー値を使用することを意味します。 数値が -128 および -1 (境界を含む) の間の場合、256 が追加され、 その数字に関してチェックが行われます。

文字列引数を指定してコールした場合、これらの関数は、その文字列の全 ての文字を調べ、その文字列の全ての文字が要求された基準に一致する場 合にのみ **TRUE** を返します。

文字列または整数以外のものを指定した場合は、直ちに **FALSE** が 返されます。

ctype 関数は、正規表現よりもつねに好ましく、さらに str_* および is_* のような いくつかの等価な関数よりも好ましいことに注意してください。 これは、ctype 関数がネーティブな C ライブラリを使用しており、処理が著しく 高速であるためです。

要件

常に利用可能な標準 C ライブラリ関数以外は不要です。

インストール手順

PHP 4.2.0以降、これらの関数はデフォルトで有効となりました。 以前のバージョンでは、configure に--enable-ctypeを指定してPHPをコンパイルする必要があります。--disable-ctypeによりctypeを無効にすること ができます。

Windows 版の *PHP* にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

> **注意**: ctypeの組込みサポートは、PHP 4.3.0で利用可能です。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

ctype_alnum

(PHP 4 >= 4.0.4, PHP 5)

ctype_alnum — 英数字かどうかを調べる

説明

bool **ctype_alnum** (string `$text`)

与えられた文字列 *text* のすべての文字が英字または 数字であるかどうかを調べます。標準の *C* ロケールの場合、 文字は *[A-Za-z]* となります。

パラメータ

text

> 調べる文字列。

返り値

text のすべての文字が英字または数字だった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 A ctype_alnum() の例 (デフォルトのロケールを使用)

```
<?php
$strings = array('AbCd1zyZ9', 'foo!#$bar');
foreach ($strings as $testcase) {
    if (ctype_alnum($testcase)) {
        echo "The string $testcase consists of all letters or digits.¥n";
    } else {
        echo "The string $testcase does not consist of all letters or digits.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string AbCd1zyZ9 consists of all letters or digits.
The string foo!#$bar does not consist of all letters or digits.
```

参考

- ctype_alpha()
- ctype_digit()
- setlocale()

---

ctype_alpha

(PHP 4 >= 4.0.4, PHP 5)

ctype_alpha — 英字かどうかを調べる

説明

bool **ctype_alpha** (string $text)

与えられた文字列 *text* のすべての文字が 英字であるかどうかを調べます。 標準の *C* ロケールの場合、文字は *[A-Za-z]* で、**ctype_alpha()** は $text が一文字のみの場合の *(ctype_upper($text) || ctype_lower($text))* と等価です。 しかし、他の言語には大文字でも小文字でもない文字が含まれています。

パラメータ

text

調べる文字列。

返り値

text のすべての文字が英字だった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_alpha()の例 (デフォルトのロケールを使用)

```
<?php
$strings = array('KjgWZC', 'arf12');
foreach ($strings as $testcase) {
    if (ctype_alpha($testcase)) {
        echo "文字列 $testcase は全て文字からなります。\n";
    } else {
        echo "文字列 $testcase は全てが文字から構成されているわけではありません。\n";
    }
}
?>
```

上の例の出力は以下となります。

```
文字列 KjgWZC は全て文字からなります。
文字列 arf12 は全てが文字から構成されているわけではありません。
```

参考

- ctype_upper()
- ctype_lower()
- setlocale()

---

ctype_cntrl

(PHP 4 >= 4.0.4, PHP 5)

ctype_cntrl — 制御文字かどうかを調べる

説明

bool **ctype_cntrl** (string $text)

与えられた文字列 *text* のすべての文字が制御文字であるかどうかを調べます。 制御文字とは、例えばラインフィードやタブ、エスケープなどです。

パラメータ

text

調べる文字列。

返り値

text のすべての文字が現在のロケールの制御文字だった場合に **TRUE**、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_cntrl() の例

```
<?php
$strings = array('string1' => "¥n¥r¥t", 'string2' => 'arf12');
foreach ($strings as $name => $testcase) {
    if (ctype_cntrl($testcase)) {
        echo "The string '$name' consists of all control characters.¥n";
    } else {
        echo "The string '$name' does not consist of all control characters.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string 'string1' consists of all control characters.
The string 'string2' does not consist of all control characters.
```

参考

- ctype_print()

---

ctype_digit

(PHP 4 >= 4.0.4, PHP 5)

ctype_digit — 数字かどうかを調べる

説明

bool **ctype_digit** (string $text)

与えられた文字列 *text* のすべての文字が 数字であるかどうかを調べます。

パラメータ

text

調べる文字列。

返り値

text のすべての文字が数字だった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_digit() の例

```
<?php
$strings = array('1820.20', '10002', 'wsl!12');
foreach ($strings as $testcase) {
    if (ctype_digit($testcase)) {
        echo "The string $testcase consists of all digits.¥n";
    } else {
        echo "The string $testcase does not consist of all digits.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string 1820.20 does not consist of all digits.
The string 10002 consists of all digits.
The string wsl!12 does not consist of all digits.
```

参考

- ctype_alnum()
- ctype_xdigit()

---

ctype_graph

(PHP 4 >= 4.0.4, PHP 5)

ctype_graph — 空白以外の印字可能な文字かどうかを調べる

説明

bool **ctype_graph** (string $text)

与えられた文字列 *text* のすべての文字が 実際に目に見える出力を行うかどうかを調べます。

パラメータ

text

調べる文字列。

返り値

text のすべての文字が印字可能で実際に目に見える 出力を行う（空白でない）場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_graph() の例

```
<?php
$strings = array('string1' => "asdf¥n¥r¥t", 'string2' => 'arf12', 'string3' => 'LKA#@%.54');
foreach ($strings as $name => $testcase) {
    if (ctype_graph($testcase)) {
        echo "The string '$name' consists of all (visibly) printable characters.¥n";
    } else {
        echo "The string '$name' does not consist of all (visibly) printable characters.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string 'string1' does not consist of all (visibly) printable characters.
The string 'string2' consists of all (visibly) printable characters.
The string 'string3' consists of all (visibly) printable characters.
```

参考

- ctype_alnum()
- ctype_print()
- ctype_punct()

---

ctype_lower

(PHP 4 >= 4.0.4, PHP 5)

ctype_lower — 小文字かどうかを調べる

説明

bool **ctype_lower** (string $text)

text のすべての文字が小文字だった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

パラメータ

text

調べる文字列。

返り値

text のすべての文字がカレントのロケールで 小文字だった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_lower() の例（デフォルトのロケールを利用）

```
<?php
$strings = array('aac123', 'qiutoas', 'QASsdks');
foreach ($strings as $testcase) {
    if (ctype_lower($testcase)) {
        echo "The string $testcase consists of all lowercase letters.¥n";
    } else {
        echo "The string $testcase does not consist of all lowercase letters.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string aac123 does not consist of all lowercase letters.
The string qiutoas consists of all lowercase letters.
The string QASsdks does not consist of all lowercase letters.
```

参考

- ctype_alpha()
- ctype_upper()
- setlocale()

---

ctype_print

(PHP 4 >= 4.0.4, PHP 5)

ctype_print — 印字可能な文字かどうかを調べる

説明

bool **ctype_print** (string $text)

与えられた文字列 *text* のすべての文字が 印字可能な文字であるかどうかを調べます。

パラメータ

text

調べる文字列。

返り値

text のすべての文字が（空白を含めて）実際に 出力を行う場合に **TRUE** 、*text* に制御文字 またはまったく出力も制御も行わない文字が含まれる場合に **FALSE** を返します。

例

Example#1 ctype_print() の例

```
<?php
$strings = array('string1' => "asdf¥n¥r¥t", 'string2' => 'arf12', 'string3' => 'LKA#@%.54');
foreach ($strings as $name => $testcase) {
    if (ctype_print($testcase)) {
        echo "The string '$name' consists of all printable characters.¥n";
```

```
    } else {
        echo "The string '$name' does not consist of all printable characters.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string 'string1' does not consist of all printable characters.
The string 'string2' consists of all printable characters.
The string 'string3' consists of all printable characters.
```

参考

- ctype_cntrl()
- ctype_graph()
- ctype_punct()

---

ctype_punct

(PHP 4 >= 4.0.4, PHP 5)

ctype_punct — 空白、英数字以外の出力可能な文字かどうかを調べる

説明

bool **ctype_punct** (string $text)

与えられた文字列 *text* のすべての文字が 句読点であるかどうかを調べます。

パラメータ

text

調べる文字列。

返り値

text のすべての文字が出力可能であり、 かつ文字でも数字でも空白でもなかった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_punct() の例

```
<?php
$strings = array('ABasdk!@!$#', '!@ # $', '*&$()');
foreach ($strings as $testcase) {
    if (ctype_punct($testcase)) {
        echo "The string $testcase consists of all punctuation.¥n";
    } else {
        echo "The string $testcase does not consist of all punctuation.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string ABasdk!@!$# does not consist of all punctuation.
The string !@ # $ does not consist of all punctuation.
The string *&$() consists of all punctuation.
```

参考

- ctype_cntrl()
- ctype_graph()

---

ctype_space

(PHP 4 >= 4.0.4, PHP 5)

ctype_space — 空白文字かどうか調べる

説明

bool **ctype_space** (string $text)

与えられた文字列 *text* のすべての文字が 空白文字であるかどうかを調べます。

パラメータ

text

調べる文字列。

返り値

text のすべての文字がなんらかの空白文字を 生成する場合に **TRUE** 、そうでない場合に **FALSE** を返します。 空白文字には、タブ・垂直タブ・改行・復帰・フォームフィード文字も 含まれます。

例

Example#1 ctype_space() の例

```
<?php
$strings = array('string1' => "¥n¥r¥t", 'string2' => "¥narf12", 'string3' => '¥n¥r¥t');
foreach ($strings as $name => $testcase) {
    if (ctype_space($testcase)) {
        echo "The string '$name' consists of all whitespace characters.¥n";
    } else {
        echo "The string '$name' does not consist of all whitespace characters.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string 'string1' consists of all whitespace characters.
The string 'string2' does not consist of all whitespace characters.
The string 'string3' does not consist of all whitespace characters.
```

参考

- ctype_cntrl()
- ctype_graph()
- ctype_punct()

---

ctype_upper

(PHP 4 >= 4.0.4, PHP 5)

ctype_upper — 大文字かどうか調べる

説明

bool **ctype_upper** (string $text)

与えられた文字列 *text* のすべての文字が 大文字であるかどうかを調べます。

パラメータ

text

調べる文字列。

返り値

text のすべての文字がカレントのロケールで 大文字だった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_upper() の例（デフォルトのロケールを利用）

```
<?php
$strings = array('AKLWC139', 'LMNSDO', 'akwSKWsm');
foreach ($strings as $testcase) {
    if (ctype_upper($testcase)) {
        echo "The string $testcase consists of all uppercase letters.¥n";
    } else {
        echo "The string $testcase does not consist of all uppercase letters.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string AKLWC139 does not consist of all uppercase letters.
The string LMNSDO consists of all uppercase letters.
The string akwSKWsm does not consist of all uppercase letters.
```

参考

- ctype_alpha()
- ctype_lower()
- setlocale()

---

ctype_xdigit

(PHP 4 >= 4.0.4, PHP 5)

ctype_xdigit — 16 進数を表す文字かどうかを調べる

説明

bool **ctype_xdigit** (string $text)

与えられた文字列 *text* のすべての文字が 16 進の '数字' であるかどうかを調べます。

パラメータ

text

調べる文字列。

返り値

text のすべての文字が 16 進の '数字' つまり 10 進の数字または *[A-Fa-f]* だった場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 ctype_xdigit() の例

```
<?php
$strings = array('AB10BC99', 'AR1012', 'ab12bc99');
foreach ($strings as $testcase) {
    if (ctype_xdigit($testcase)) {
        echo "The string $testcase consists of all hexadecimal digits.¥n";
    } else {
        echo "The string $testcase does not consist of all hexadecimal digits.¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
The string AB10BC99 consists of all hexadecimal digits.
The string AR1012 does not consist of all hexadecimal digits.
The string ab12bc99 consists of all hexadecimal digits.
```

参考

- ctype_digit()

目次

- ctype_alnum — 英数字かどうかを調べる
- ctype_alpha — 英字かどうかを調べる
- ctype_cntrl — 制御文字かどうかを調べる
- ctype_digit — 数字かどうかを調べる
- ctype_graph — 空白以外の印字可能な文字かどうかを調べる
- ctype_lower — 小文字かどうかを調べる
- ctype_print — 印字可能な文字かどうかを調べる
- ctype_punct — 空白、英数字以外の出力可能な文字かどうかを調べる
- ctype_space — 空白文字かどうか調べる
- ctype_upper — 大文字かどうか調べる
- ctype_xdigit — 16 進数を表す文字かどうかを調べる

CURL, Client URL Library 関数

導入

PHP は、Daniel Stenbergにより開発されたライブラリlibcurl をサポー トします。このライブラリにより、多くの異なったプロトコルで様々なサーバーと接続し、通信することが可能になります。libcurl は現在 http, https, ftp, gopher, telnet, dict, file, ldap プロトコルをサ ポートしてします。libcurl は HTTPS 認証、HTTP POST、HTTP PUT、FTP アップロード(これはPHPのftp拡張機能でも実行可能です)、HTTPフォーム によるアップロード、プロキシ、クッキー、ユーザ名+パスワードによる 認証もサポートします。

これらの関数は、PHP 4.0.2で追加されました。

要件

PHP の cURL 関数を使用するためには、» libcurl パッケージをインストールしておく必要があります。PHP は、 libcurl 7.0.2-beta 以降を必要とします。 PHP バージョン 4.2.3 以降、少なくとも libcurl バージョン 7.9.0 以降が必要となっています。 PHP 4.3.0 以降では、7.9.8 以降が必要です。 PHP 5.0.0 では、おそらく、libcurl 7.10.5 以降が必要となります。

インストール手順

PHP の cURL サポート機能を使用するには、--with-curl[=DIR] を付けて PHP をコンパイル しておく必要があります。ただし、DIR は、ディレクトリ lib および include を有するディレクトリの場所となります。 ディレクトリ "include" には、"curl" という名前のフォルダがある 必要があり、そのフォルダにはファイル *easy.h* および *curl.h* がある必要が あります。*libcurl.a* という名前のファイルがディレクトリ "lib" にある必要があります。PHP 4.3.0 以降、URL ストリームで cURL を使用するよう PHP を 設定するために --with-curlwrappers を指定できます。

> **注意**: **Win32 ユーザへの注意** このモジュールを Windows 環境で使用可能とするには、 *libeay32.dll* および *ssleay32.dll* が PATH の通った場所に存在する必要があります。 cURL のサイトにある *libcurl.dll* は不要です。

リソース型

この拡張モジュールで定義しているリソース型は cURL ハンドルおよび cURL マルチハンドルのふたつです。

定義済み定数

cURL の 定義済み定数 も参照ください。

例

PHP を cURL サポート機能付きでコンパイルすると、 curl 関数を使用可能となります。cURL 関数の基本的な使用法は、 curl_init()により cURL セッションを初期化、 curl_setopt() により転送時のオプションを設定、 続いてcurl_exec() により転送を実行し、 curl_close() によりセッションを終了するというものになります。 cURL 関数を使用して PHP ホームページをファイルに取得する例を示します。

Example#1 PHP の cURL モジュールを使用して example.com のホームページを取得する

```
<?php

$ch = curl_init("http://www.example.com/");
$fp = fopen("example_homepage.txt", "w");

curl_setopt($ch, CURLOPT_FILE, $fp);
curl_setopt($ch, CURLOPT_HEADER, 0);

curl_exec($ch);
curl_close($ch);
fclose($fp);
?>
```

定数

(No version information available, might be only in CVS)

定数 — Curl の定義済み定数

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

これらの定数についての説明や実際の使用法は、 curl_setopt() および curl_getinfo() のドキュメントを参照ください。

CURLOPT_AUTOREFERER (integer)
PHP 5.1.0 以降で使用可能です。

CURLOPT_COOKIESESSION (integer)
PHP 5.1.0 以降で使用可能です。

CURLOPT_DNS_USE_GLOBAL_CACHE (integer)

CURLOPT_DNS_CACHE_TIMEOUT (integer)

CURLOPT_FTP_SSL (integer)
PHP 5.2.0 以降で使用可能です。

CURLFTPSSL_TRY (integer)
PHP 5.2.0 以降で使用可能です。

CURLFTPSSL_ALL (integer)
PHP 5.2.0 以降で使用可能です。

CURLFTPSSL_CONTROL (integer)
PHP 5.2.0 以降で使用可能です。

CURLFTPSSL_NONE (integer)
PHP 5.2.0 以降で使用可能です。

CURLOPT_PRIVATE (integer)
PHP 5.2.4 以降で使用可能です。

CURLOPT_FTPSSLAUTH (integer)
PHP 5.1.0 以降で使用可能です。

CURLOPT_PORT (integer)

CURLOPT_FILE (integer)

CURLOPT_INFILE (integer)

CURLOPT_INFILESIZE (integer)

CURLOPT_URL (integer)

CURLOPT_PROXY (integer)

CURLOPT_VERBOSE (integer)

CURLOPT_HEADER (integer)

CURLOPT_HTTPHEADER (integer)

CURLOPT_NOPROGRESS (integer)

CURLOPT_NOBODY (integer)

CURLOPT_FAILONERROR (integer)

CURLOPT_UPLOAD (integer)

CURLOPT_POST (integer)

CURLOPT_FTPLISTONLY (integer)

CURLOPT_FTPAPPEND (integer)

CURLOPT_FTP_CREATE_MISSING_DIRS (integer)

CURLOPT_NETRC (integer)

CURLOPT_FOLLOWLOCATION (integer)
open_basedir あるいは safe_mode が有効な場合は、 この定数は使用できません。

CURLOPT_FTPASCII (integer)

CURLOPT_PUT (integer)

CURLOPT_MUTE (integer)

CURLOPT_USERPWD (integer)

CURLOPT_PROXYUSERPWD (integer)

CURLOPT_RANGE (integer)

CURLOPT_TIMEOUT (integer)

CURLOPT_TIMEOUT_MS (integer)

CURLOPT_TCP_NODELAY (integer)
PHP 5.2.1 以降で使用可能です。

CURLOPT_POSTFIELDS (integer)

CURLOPT_REFERER (integer)

CURLOPT_USERAGENT (integer)

CURLOPT_FTPPORT (integer)

CURLOPT_FTP_USE_EPSV (integer)

CURLOPT_LOW_SPEED_LIMIT (integer)

CURLOPT_LOW_SPEED_TIME (integer)

CURLOPT_RESUME_FROM (integer)

CURLOPT_COOKIE (integer)

CURLOPT_SSLCERT (integer)

CURLOPT_SSLCERTPASSWD (integer)

CURLOPT_WRITEHEADER (integer)

CURLOPT_SSL_VERIFYHOST (integer)

CURLOPT_COOKIEFILE (integer)

CURLOPT_SSLVERSION (integer)

CURLOPT_TIMECONDITION (integer)

CURLOPT_TIMEVALUE (integer)

CURLOPT_CUSTOMREQUEST (integer)

CURLOPT_STDERR (integer)

CURLOPT_TRANSFERTEXT (integer)

CURLOPT_RETURNTRANSFER (integer)

CURLOPT_QUOTE (integer)

CURLOPT_POSTQUOTE (integer)

CURLOPT_INTERFACE (integer)

CURLOPT_KRB4LEVEL (integer)

CURLOPT_HTTPPROXYTUNNEL (integer)

CURLOPT_FILETIME (integer)

CURLOPT_WRITEFUNCTION (integer)

CURLOPT_READFUNCTION (integer)

CURLOPT_PASSWDFUNCTION (integer)

CURLOPT_HEADERFUNCTION (integer)

CURLOPT_MAXREDIRS (integer)

CURLOPT_MAXCONNECTS (integer)

CURLOPT_CLOSEPOLICY (integer)

CURLOPT_FRESH_CONNECT (integer)

CURLOPT_FORBID_REUSE (integer)

CURLOPT_RANDOM_FILE (integer)

CURLOPT_EGDSOCKET (integer)

CURLOPT_CONNECTTIMEOUT (integer)

CURLOPT_CONNECTTIMEOUT_MS (integer)

CURLOPT_SSL_VERIFYPEER (integer)

CURLOPT_CAINFO (integer)

CURLOPT_CAPATH (integer)

CURLOPT_COOKIEJAR (integer)

CURLOPT_SSL_CIPHER_LIST (integer)

CURLOPT_BINARYTRANSFER (integer)

CURLOPT_NOSIGNAL (integer)

CURLOPT_PROXYTYPE (integer)

CURLOPT_BUFFERSIZE (integer)

CURLOPT_HTTPGET (integer)

CURLOPT_HTTP_VERSION (integer)

CURLOPT_SSLKEY (integer)

CURLOPT_SSLKEYTYPE (integer)

CURLOPT_SSLKEYPASSWD (integer)

CURLOPT_SSLENGINE (integer)

CURLOPT_SSLENGINE_DEFAULT (integer)

CURLOPT_SSLCERTTYPE (integer)

CURLOPT_CRLF (integer)

CURLOPT_ENCODING (integer)

CURLOPT_PROXYPORT (integer)

CURLOPT_UNRESTRICTED_AUTH (integer)

CURLOPT_FTP_USE_EPRT (integer)

CURLOPT_HTTP200ALIASES (integer)

CURLOPT_HTTPAUTH (integer)

CURLAUTH_BASIC (integer)

CURLAUTH_DIGEST (integer)

CURLAUTH_GSSNEGOTIATE (integer)

CURLAUTH_NTLM (integer)

CURLAUTH_ANY (integer)

CURLAUTH_ANYSAFE (integer)

CURLOPT_PROXYAUTH (integer)

CURLCLOSEPOLICY_LEAST_RECENTLY_USED (integer)

CURLCLOSEPOLICY_LEAST_TRAFFIC (integer)

CURLCLOSEPOLICY_SLOWEST (integer)

CURLCLOSEPOLICY_CALLBACK (integer)

CURLCLOSEPOLICY_OLDEST (integer)

CURLINFO_PRIVATE (integer)
PHP 5.2.4 以降で使用可能です。

CURLINFO_EFFECTIVE_URL (integer)

CURLINFO_HTTP_CODE (integer)

CURLINFO_HEADER_OUT (integer)
PHP 5.1.3 以降で使用可能です。

CURLINFO_HEADER_SIZE (integer)

CURLINFO_REQUEST_SIZE (integer)

CURLINFO_TOTAL_TIME (integer)

CURLINFO_NAMELOOKUP_TIME (integer)

CURLINFO_CONNECT_TIME (integer)

CURLINFO_PRETRANSFER_TIME (integer)

CURLINFO_SIZE_UPLOAD (integer)

CURLINFO_SIZE_DOWNLOAD (integer)

CURLINFO_SPEED_DOWNLOAD (integer)

CURLINFO_SPEED_UPLOAD (integer)

CURLINFO_FILETIME (integer)

CURLINFO_SSL_VERIFYRESULT (integer)

CURLINFO_CONTENT_LENGTH_DOWNLOAD (integer)

CURLINFO_CONTENT_LENGTH_UPLOAD (integer)

CURLINFO_STARTTRANSFER_TIME (integer)

CURLINFO_CONTENT_TYPE (integer)

CURLINFO_REDIRECT_TIME (integer)

CURLINFO_REDIRECT_COUNT (integer)

CURL_TIMECOND_IFMODSINCE (integer)

CURL_TIMECOND_IFUNMODSINCE (integer)

CURL_TIMECOND_LASTMOD (integer)

CURL_VERSION_IPV6 (integer)

CURL_VERSION_KERBEROS4 (integer)

CURL_VERSION_SSL (integer)

CURL_VERSION_LIBZ (integer)

CURLVERSION_NOW (integer)

CURLE_OK (integer)

CURLE_UNSUPPORTED_PROTOCOL (integer)

CURLE_FAILED_INIT (integer)

CURLE_URL_MALFORMAT (integer)

CURLE_URL_MALFORMAT_USER (integer)

CURLE_COULDNT_RESOLVE_PROXY (integer)

CURLE_COULDNT_RESOLVE_HOST (integer)

CURLE_COULDNT_CONNECT (integer)

CURLE_FTP_WEIRD_SERVER_REPLY (integer)

CURLE_FTP_ACCESS_DENIED (integer)

CURLE_FTP_USER_PASSWORD_INCORRECT (integer)

CURLE_FTP_WEIRD_PASS_REPLY (integer)

CURLE_FTP_WEIRD_USER_REPLY (integer)

CURLE_FTP_WEIRD_PASV_REPLY (integer)

CURLE_FTP_WEIRD_227_FORMAT (integer)

CURLE_FTP_CANT_GET_HOST (integer)

CURLE_FTP_CANT_RECONNECT (integer)

CURLE_FTP_COULDNT_SET_BINARY (integer)

CURLE_PARTIAL_FILE (integer)

CURLE_FTP_COULDNT_RETR_FILE (integer)

CURLE_FTP_WRITE_ERROR (integer)

CURLE_FTP_QUOTE_ERROR (integer)

CURLE_HTTP_NOT_FOUND (integer)

CURLE_WRITE_ERROR (integer)

CURLE_MALFORMAT_USER (integer)

CURLE_FTP_COULDNT_STOR_FILE (integer)

CURLE_READ_ERROR (integer)

CURLE_OUT_OF_MEMORY (integer)

CURLE_OPERATION_TIMEOUTED (integer)

CURLE_FTP_COULDNT_SET_ASCII (integer)

CURLE_FTP_PORT_FAILED (integer)

CURLE_FTP_COULDNT_USE_REST (integer)

CURLE_FTP_COULDNT_GET_SIZE (integer)

CURLE_HTTP_RANGE_ERROR (integer)

CURLE_HTTP_POST_ERROR (integer)

CURLE_SSL_CONNECT_ERROR (integer)

CURLE_FTP_BAD_DOWNLOAD_RESUME (integer)

CURLE_FILE_COULDNT_READ_FILE (integer)

CURLE_LDAP_CANNOT_BIND (integer)

CURLE_LDAP_SEARCH_FAILED (integer)

CURLE_LIBRARY_NOT_FOUND (integer)

CURLE_FUNCTION_NOT_FOUND (integer)

CURLE_ABORTED_BY_CALLBACK (integer)

CURLE_BAD_FUNCTION_ARGUMENT (integer)

CURLE_BAD_CALLING_ORDER (integer)

CURLE_HTTP_PORT_FAILED (integer)

CURLE_BAD_PASSWORD_ENTERED (integer)

CURLE_TOO_MANY_REDIRECTS (integer)

CURLE_UNKNOWN_TELNET_OPTION (integer)

CURLE_TELNET_OPTION_SYNTAX (integer)

CURLE_OBSOLETE (integer)

CURLE_SSL_PEER_CERTIFICATE (integer)

CURLE_GOT_NOTHING (integer)

CURLE_SSL_ENGINE_NOTFOUND (integer)

CURLE_SSL_ENGINE_SETFAILED (integer)

CURLE_SEND_ERROR (integer)

CURLE_RECV_ERROR (integer)

CURLE_SHARE_IN_USE (integer)

CURLE_SSL_CERTPROBLEM (integer)

CURLE_SSL_CIPHER (integer)

CURLE_SSL_CACERT (integer)

CURLE_BAD_CONTENT_ENCODING (integer)

CURLE_LDAP_INVALID_URL (integer)

CURLE_FILESIZE_EXCEEDED (integer)

CURLE_FTP_SSL_FAILED (integer)

CURLFTPAUTH_DEFAULT (integer)
PHP 5.1.0 以降で使用可能です。

CURLFTPAUTH_SSL (integer)
PHP 5.1.0 以降で使用可能です。

CURLFTPAUTH_TLS (integer)
PHP 5.1.0 以降で使用可能です。

CURLPROXY_HTTP (integer)

CURLPROXY_SOCKS5 (integer)

CURL_NETRC_OPTIONAL (integer)

CURL_NETRC_IGNORED (integer)

CURL_NETRC_REQUIRED (integer)

CURL_HTTP_VERSION_NONE (integer)

CURL_HTTP_VERSION_1_0 (integer)

CURL_HTTP_VERSION_1_1 (integer)

CURLM_CALL_MULTI_PERFORM (integer)

CURLM_OK (integer)

CURLM_BAD_HANDLE (integer)

CURLM_BAD_EASY_HANDLE (integer)

CURLM_OUT_OF_MEMORY (integer)

CURLM_INTERNAL_ERROR (integer)

CURLMSG_DONE (integer)

---

curl_close

(PHP 4 >= 4.0.2, PHP 5)

curl_close — cURL セッションを閉じる

説明

void **curl_close** (resource $ch)

cURL セッションを閉じ、全てのリソースを開放します。 cURL ハンドル *ch* も削除されます。

パラメータ

ch

curl_init() が返す cURL ハンドル。

返り値

値を返しません。

例

Example#1 新しい cURL セッションの初期化とウェブページの取得

```
<?php
// 新しい cURL リソースを作成します
$ch = curl_init();

// URL とその他のオプションを設定します
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, 0);

// URL を取得し、それをブラウザに渡します
curl_exec($ch);

// cURL リソースを閉じ、システムリソースを解放します
curl_close($ch);
?>
```

参考

- curl_init()
- curl_multi_close()

---

curl_copy_handle

(PHP 5)

curl_copy_handle — cURL ハンドルを、その設定も含めてコピーする

説明

resource **curl_copy_handle** (resource $ch)

cURL ハンドルをコピーし、同じ設定を保持します。

パラメータ

ch

curl_init() が返す cURL ハンドル。

返り値

新しい cURL ハンドルを返します。

例

Example#1 cURL ハンドルのコピー

```
<?php
// 新しい cURL リソースを作成します
$ch = curl_init();

// URL その他のオプションを適切に設定します
curl_setopt($ch, CURLOPT_URL, 'http://www.example.com/');
curl_setopt($ch, CURLOPT_HEADER, 0);

// ハンドルをコピーします
$ch2 = curl_copy_handle($ch);

// URL (http://www.example.com/) を取得し、ブラウザに渡します
curl_exec($ch2);

// cURL リソースを閉じ、システムリソースを開放します
curl_close($ch2);
curl_close($ch);
?>
```

---

curl_errno

(PHP 4 >= 4.0.3, PHP 5)

curl_errno — 直近のエラー番号を返す

説明

int **curl_errno** (resource $ch)

直近の cURL 処理に関するエラー番号を返します。

パラメータ

ch

curl_init() が返す cURL ハンドル。

返り値

エラー番号を返します。エラーが発生しない場合、 *0* (ゼロ) を返します。

参考

- curl_error()
- » Curl エラーコード

---

curl_error

(PHP 4 >= 4.0.3, PHP 5)

curl_error — 現在のセッションに関する直近のエラー文字列を返す

説明

string **curl_error** (resource $ch)

直近の cURL 操作に関するエラーメッセージをクリアテキストで返します。

パラメータ

ch

curl_init() が返す cURL ハンドル。

返り値

エラー番号、あるいはエラーが発生しなかった場合は '' (空文字) を返します。

参考

- curl_errno()
- » Curl エラーコード

---

curl_exec

(PHP 4 >= 4.0.2, PHP 5)

curl_exec — cURL セッションを実行する

説明

mixed **curl_exec** (resource $ch)

指定した cURL セッションを実行します。

この関数は、cURL セッションを初期化し、 オプションを全て設定した後にコールする必要があります。

パラメータ

ch

curl_init() が返す cURL ハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 しかし、オプション **CURLOPT_RETURNTRANSFER** が設定されていると、 成功した場合に取得結果、失敗した場合に **FALSE** を返します。

例

Example#1 ウェブページの取得

```
<?php
// 新規 cURL リソースを作成します
$ch = curl_init();

// URL や他の適当なオプションを設定します
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, 0);

// URL を取得し、ブラウザに渡します
curl_exec($ch);

// cURL リソースを閉じ、システムリソースを解放します
curl_close($ch);
?>
```

参考

- curl_multi_exec()

---

curl_getinfo

(PHP 4 >= 4.0.4, PHP 5)

curl_getinfo — 指定した伝送に関する情報を得る

説明

mixed **curl_getinfo** (resource $ch [, int $opt])

直近の転送に関する情報を取得します。

パラメータ

ch

curl_init() が返す cURL ハンドル。

opt

これは、以下のいずれかの定数となります。

- **CURLINFO_EFFECTIVE_URL** - 直近の有効な URL
- **CURLINFO_HTTP_CODE** - 最後に受け取った HTTP コード
- **CURLINFO_FILETIME** - ドキュメントを取得するのにかかった時間。 取得できなかった場合は -1
- **CURLINFO_TOTAL_TIME** - 直近の伝送にかかった秒数
- **CURLINFO_NAMELOOKUP_TIME** - 名前解決が完了するまでにかかった秒数
- **CURLINFO_CONNECT_TIME** - 接続を確立するまでにかかった秒数
- **CURLINFO_PRETRANSFER_TIME** - 開始からファイル伝送がはじまるまでにかかった秒数
- **CURLINFO_STARTTRANSFER_TIME** - 最初のバイトの伝送がはじまるまでの秒数
- **CURLINFO_REDIRECT_TIME** - 伝送が始まるまでのリダイレクト処理の秒数
- **CURLINFO_SIZE_UPLOAD** - アップロードされたバイト数
- **CURLINFO_SIZE_DOWNLOAD** - ダウンロードされたバイト数
- **CURLINFO_SPEED_DOWNLOAD** - 平均のダウンロード速度
- **CURLINFO_SPEED_UPLOAD** - 平均のアップロード速度
- **CURLINFO_HEADER_SIZE** - 受信したヘッダのサイズ
- **CURLINFO_HEADER_OUT** - 送信したリクエスト文字列。PHP 5.1.3 以降で使用可能
- **CURLINFO_REQUEST_SIZE** - 発行されたリクエストのサイズ。現在は HTTP リクエストの場合のみ
- **CURLINFO_SSL_VERIFYRESULT** - CURLOPT_SSL_VERIFYPEER を設定した際に 要求される SSL 証明書の認証結果
- **CURLINFO_CONTENT_LENGTH_DOWNLOAD** - ダウンロードされるサイズ。 Content-Length: フィールドの内容を取得する
- **CURLINFO_CONTENT_LENGTH_UPLOAD** - アップロードされるサイズ。
- **CURLINFO_CONTENT_TYPE** - ダウンロードされたオブジェクトの Content-type 。 NULL は、サーバが適切な Content-Type: ヘッダを返さなかったことを示す

返り値

opt を指定した場合は、その値を文字列で返します。 それ以外の場合は、以下の要素をもつ連想配列を返します (それぞれの要素が *opt* に対応します)。

- "url"
- "content_type"
- "http_code"
- "header_size"
- "request_size"
- "filetime"
- "ssl_verify_result"
- "redirect_count"
- "total_time"
- "namelookup_time"
- "connect_time"
- "pretransfer_time"
- "size_upload"
- "size_download"
- "speed_download"

- "speed_upload"
- "download_content_length"
- "upload_content_length"
- "starttransfer_time"
- "redirect_time"

---

curl_init

(PHP 4 >= 4.0.2, PHP 5)

curl_init — cURL セッションを初期化する

説明

resource **curl_init** ([string `$url`])

新規セッションを初期化し、cURL ハンドルを返します。このハンドルは、関数 curl_setopt(), curl_exec(), curl_close() で使用します。

パラメータ

`url`

`url` を指定した場合、オプション CURLOPT_URL がそのパラメータの値に設定されます。関数 curl_setopt() により、 この値をマニュアルで設定することも可能です。

返り値

成功した場合に cURL ハンドル、エラー時に **FALSE** を返します。

例

Example#1 新しい cURL セッションを初期化し、ウェブページを取得する

```
<?php
// 新しい cURL リソースを作成します
$ch = curl_init();

// URL や他の適当なオプションを設定します
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, 0);

// URL を取得し、ブラウザに渡します
curl_exec($ch);

// cURL リソースを閉じ、システムリソースを解放します
curl_close($ch);
?>
```

参考

- curl_close()
- curl_multi_init()

---

curl_multi_add_handle

(PHP 5)

curl_multi_add_handle — cURL マルチハンドルに、通常の cURL ハンドルを追加する

説明

int **curl_multi_add_handle** (resource `$mh` , resource `$ch`)

`ch` ハンドルを、マルチハンドル *`mh`* に追加します。

パラメータ

`mh`

curl_multi_init() が返す cURL マルチハンドル。

ch

curl_init() が返す cURL ハンドル。

返り値

成功した場合に 0、あるいはエラーコード **CURLM_XXX** のいずれかを返します。

例

Example#1 curl_multi_add_handle() の例

この例は、ふたつの cURL ハンドルを作成し、それをマルチハンドルに追加して並列実行します。

```
<?php
// cURL リソースを作成します
$ch1 = curl_init();
$ch2 = curl_init();

// URL およびその他適切なオプションを設定します。
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

// マルチ cURL ハンドルを作成します
$mh = curl_multi_init();

// ふたつのハンドルを追加します
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$running=null;
// ハンドルを実行します
do {
    curl_multi_exec($mh,$running);
} while($running > 0);

// すべてのハンドルを閉じます
curl_multi_remove_handle($ch1);
curl_multi_remove_handle($ch2);
curl_multi_close($mh);
?>
```

参考

- curl_multi_remove_handle()
- curl_multi_init()
- curl_init()

---

curl_multi_close

(PHP 5)

curl_multi_close — cURL ハンドルのセットを閉じる

説明

void **curl_multi_close** (resource $mh)

cURL ハンドルのセットを閉じます。

パラメータ

mh

curl_multi_init() が返す cURL マルチハンドル。

返り値

値を返しません。

例

Example#1 curl_multi_close() の例

この例は、ふたつの cURL ハンドルを作成し、それをマルチハンドルに追加して並列実行します。

```php
<?php
// cURL リソースを作成します
$ch1 = curl_init();
$ch2 = curl_init();

// URL およびその他適切なオプションを設定します。
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

// マルチ cURL ハンドルを作成します
$mh = curl_multi_init();

// ふたつのハンドルを追加します
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$running=null;
// ハンドルを実行します
do {
    curl_multi_exec($mh,$running);
} while ($running > 0)
// ハンドルを閉じます
curl_multi_remove_handle($ch1);
curl_multi_remove_handle($ch2);
curl_multi_close($mh);

?>
```

参考

- curl_multi_init()
- curl_close()

---

curl_multi_exec

(PHP 5)

curl_multi_exec — 現在の cURL ハンドルから、サブ接続を実行する

説明

int **curl_multi_exec** (resource `$mh` , int `&$still_running`)

スタック内の各ハンドルを処理します。 このメソッドは、ハンドルがデータの読み書きを要するかどうかにかかわらずコール可能です。

パラメータ

`mh`

curl_multi_init() が返す cURL マルチハンドル。

`still_running`

処理が実行中かどうかを表すフラグへの参照。

返り値

cURL 定義済み定数 で定義された cURL コードを返します。

注意: これは、マルチスタック全体に関するエラーのみを返します。この関数が **`CURLM_OK`** を返したとしても、各転送で個別にエラーが発生している可能性があります。

例

Example#1 curl_multi_exec() の例

この例は、ふたつの cURL ハンドルを作成し、それをマルチハンドルに追加して並列実行します。

```php
<?php
// cURL リソースを作成します
$ch1 = curl_init();
$ch2 = curl_init();

// URL およびその他適切なオプションを設定します。
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

// マルチ cURL ハンドルを作成します
```

```
$mh = curl_multi_init();

// ふたつのハンドルを追加します
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$running=null;
// ハンドルを実行します
do {
    $mrc = curl_multi_exec($mh, $running);
} while ($mrc == CURLM_CALL_MULTI_PERFORM);

// ハンドルを閉じます
curl_multi_remove_handle($mh, $ch1);
curl_multi_remove_handle($mh, $ch2);
curl_multi_close($mh);

?>
```

参考

- curl_multi_init()
- curl_exec()

---

curl_multi_getcontent

(PHP 5)

curl_multi_getcontent — **CURLOPT_RETURNTRANSFER** が設定されている場合に、cURL ハンドルの内容を返す

説明

string **curl_multi_getcontent** (resource $ch)

CURLOPT_RETURNTRANSFER に何らかのハンドルが設定されている場合に、 この関数はその cURL ハンドルの内容を文字列形式で返します。

パラメータ

ch

curl_init() が返す cURL ハンドル。

返り値

CURLOPT_RETURNTRANSFER が設定されている場合に、 cURL ハンドルの内容を返します。

参考

- curl_multi_init()

---

curl_multi_info_read

(PHP 5)

curl_multi_info_read — 現在の転送についての情報を表示する

説明

array **curl_multi_info_read** (resource $mh)

マルチハンドルに対して、個別の転送にメッセージ/情報が残っているかどうかを問い合わせます。 メッセージには、転送時のエラーコードや転送が完了したという情報が残っている可能性があります。

この関数を繰り返しコールすると、毎回新しい結果を返します。**FALSE** が返されると、その時点でもう取得する結果がないことを意味します。
msgs_in_queue がさす値は、 この関数をコールした後に残っているメッセージの数となります。

警告

返されたリソースがさすデータは、 curl_multi_remove_handle() をコールした後には残りません。

パラメータ

mh

curl_multi_init() が返す cURL マルチハンドル。

msgs_in_queue

まだキューの中に残っているメッセージの数。

返り値

成功した場合にメッセージの連想配列、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.2.0	*msgs_in_queue* が追加されました。

参考

- curl_multi_init()

---

curl_multi_init

(PHP 5)

curl_multi_init — 新規 cURL マルチハンドルを返す

説明

resource **curl_multi_init** (void)

複数の cURL ハンドルを並列で実行できるようにします。

パラメータ

mh

curl_multi_init() が返す cURL マルチハンドル。

返り値

成功した場合に cURL、失敗した場合に **FALSE** を返します。

例

Example#1 curl_multi_init() の例

この例は、ふたつの cURL ハンドルを作成し、それをマルチハンドルに追加して並列実行します。

```
<?php
// cURL リソースを作成します
$ch1 = curl_init();
$ch2 = curl_init();

// URL およびその他適切なオプションを設定します。
curl_setopt($ch1, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch1, CURLOPT_HEADER, 0);
curl_setopt($ch2, CURLOPT_URL, "http://www.php.net/");
curl_setopt($ch2, CURLOPT_HEADER, 0);

// マルチ cURL ハンドルを作成します
$mh = curl_multi_init();

// ふたつのハンドルを追加します
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);

$running=null;
// ハンドルを実行します
do {
    curl_multi_exec($mh,$running);
} while ($running > 0);

// ハンドルを閉じます
curl_multi_remove_handle($ch1);
curl_multi_remove_handle($ch2);
curl_multi_close($mh);
```

```
?>
```

参考

- curl_init()
- curl_multi_close()

---

curl_multi_remove_handle

(PHP 5)

curl_multi_remove_handle — cURL ハンドルのセットからマルチハンドルを削除する

説明

int **curl_multi_remove_handle** (resource $mh , resource $ch)

指定した *ch* ハンドルを、*mh* ハンドルから削除します。 *ch* ハンドルが削除されてからも、このハンドルで curl_exec() を実行できます。 使用中のハンドルを削除する際には、進行中の転送をきちんと停止します。

パラメータ

mh

curl_multi_init() が返す cURL マルチハンドル。

ch

curl_init() が返す cURL ハンドル。

返り値

成功した場合に cURL ハンドル、失敗した場合に **FALSE** を返します。

参考

- curl_init()
- curl_multi_init()
- curl_multi_add_handle()

---

curl_multi_select

(PHP 5)

curl_multi_select — cURL 拡張モジュールに関連付けられているすべてのソケットを取得し、「選択可能な」状態にする

説明

int **curl_multi_select** (resource $mh [, float $timeout])

cURL 拡張モジュールに関連付けられているすべてのソケットを取得し、「選択可能な」状態にします。

パラメータ

mh

curl_multi_init() が返す cURL マルチハンドル。

timeout

レスポンスを待つ秒数。

返り値

成功した場合は、記述子セットに含まれる記述子の数を返します。 失敗した場合は、この関数は **FALSE** を返します。

参考

- curl_multi_init()

---

curl_setopt_array

(PHP 5 >= 5.1.3)

curl_setopt_array — CURL 転送用の複数のオプションを設定する

説明

bool **curl_setopt_array** (resource $ch , array $options)

cURL セッション用の複数のオプションを設定します。 この関数が便利なのは大量の cURL オプションを設定する場合で、 何度も繰り返して curl_setopt() をコールせずにすみます。

パラメータ

ch

curl_init() が返す cURL ハンドル。

options

設定するオプションとその値を指定した配列。キーとして使用できるのは、 有効な curl_setopt() 定数か、 その定数に対応する整数値だけです。

返り値

すべてのオプションがうまく設定できた場合に **TRUE** を返します。 うまく設定できないオプションがあった時点で即時に **FALSE** が返され、*options* 配列に含まれるそれ以降のオプションは無視されます。

例

Example#1 新規に CURL セッションを初期化、ウェブページを取得する

```
<?php
// 新しい cURL リソースを作成します
$ch = curl_init();

// URL その他のオプションを適切に設定します
$options = array(CURLOPT_URL => 'http://www.example.com/',
                 CURLOPT_HEADER => false
                );

curl_setopt_array($ch, $options);

// URL の内容を取得し、ブラウザに渡します
curl_exec($ch);

// cURL リソースを閉じ、システムリソースを開放します
curl_close($ch);
?>
```

PHP 5.1.4 より前のバージョンでこの関数と同等の操作をするには、次のようにします。

Example#2 curl_setopt_array() の独自実装

```
<?php
if (!function_exists('curl_setopt_array')) {
    function curl_setopt_array(&$ch, $curl_options)
    {
        foreach ($curl_options as $option => $value) {
            if (!curl_setopt($ch, $option, $value)) {
                return false;
            }
        }
        return true;
    }
}
?>
```

参考

- curl_setopt()

---

curl_setopt

(PHP 4 >= 4.0.2, PHP 5)

curl_setopt — cURL 転送用オプションを設定する

説明

bool **curl_setopt** (resource `$ch` , int `$option` , mixed `$value`)

指定した cURL セッションハンドルのオプションを設定します。

パラメータ

ch

curl_init() が返す cURL ハンドル。

option

設定したい *CURLOPT_XXX* オプション。

value

option に設定する値。

value には、 *option* の以下の値に関して bool 値を指定する必要があります。

オプション	*value* への設定値	注記
CURLOPT_AUTOREFERER	**TRUE** を設定すると、*Location:* によるリダイレクトを たどる際には自動的に *Referer:* フィールドをリクエストに 追加します。	PHP 5.1.0 以降で有効です。
CURLOPT_BINARYTRANSFER	**TRUE** を設定すると、**CURLOPT_RETURNTRANSFER** が使用された場合に出力結果を何も加工せずに返します。	
CURLOPT_COOKIESESSION	**TRUE** を設定すると、クッキーの "セッション" を新しく開始します。 以前のセッションで読み込まれていた "セッションクッキー" は無視するよう、 libcurl に指示します。デフォルトでは、それがセッションクッキーであるか どうかにかかわらず libcurl はすべてのクッキーを読み込んで保存します。 セッションクッキーとは、有効期限が指定されておらず "セッション" の間のみ 有効であるクッキーのことです。	PHP 5.1.0 以降で有効です。
CURLOPT_CRLF	**TRUE** を設定すると、転送時に Unix 形式の改行を CRLF 形式に変換します。	
CURLOPT_DNS_USE_GLOBAL_CACHE	**TRUE** を設定すると、グローバル DNS キャッシュを利用します。 このオプションはスレッドセーフではありません。また、デフォルトで 有効になっています。	
CURLOPT_FAILONERROR	**TRUE** を設定すると、HTTP で 400 以上のコードが返ってきた際に 処理失敗と判断し、何もしません。デフォルトでは、コードの値を無視して ページの内容を取得します。	
CURLOPT_FILETIME	**TRUE** を設定すると、ドキュメントの更新日時を取得しようと試みます。 この値を取得するには、curl_getinfo() で *CURLINFO_FILETIME* オプションを用います。	
CURLOPT_FOLLOWLOCATION	**TRUE** を設定すると、サーバが HTTP ヘッダの一部として送ってくる *"Location: "* ヘッダの内容をたどります （これは再帰的に行われます。**CURLOPT_MAXREDIRS** が指定されていない限り、送ってくる *"Location: "* ヘッダの内容をずっとたどり続けることに注意しましょう）。	
CURLOPT_FORBID_REUSE	**TRUE** を設定すると、処理が終了した際に明示的に接続を切断します。 接続を再利用しません。	
CURLOPT_FRESH_CONNECT	**TRUE** を設定すると、キャッシュされている接続を利用せずに 新しい接続を確立します。	
CURLOPT_FTP_USE_EPRT	**TRUE** を設定すると、FTP のダウンロードに EPRT（および LPRT） を利用します。 **FALSE** の場合は EPRT・LPRT を無効にして PORT を利用します。	PHP 5.0.0 で追加されました。
CURLOPT_FTP_USE_EPSV	**TRUE** を設定すると、FTP 転送の際にまず EPSV コマンドの利用を 試みます。失敗した場合は PASV を利用します。**FALSE** を設定すると、 EPSV を無効にします。	
CURLOPT_FTPAPPEND	**TRUE** を設定すると、リモートファイルを上書きせずに追記します。	
CURLOPT_FTPASCII	**CURLOPT_TRANSFERTEXT** のエイリアスです。	
CURLOPT_FTPLISTONLY	**TRUE** を設定すると、FTP でディレクトリ名のみ表示します。	
CURLOPT_HEADER	**TRUE** を設定すると、ヘッダの内容も出力します。	
CURLOPT_HTTPGET	**TRUE** を設定すると、HTTP のリクエスト形式を GET に戻します。 GET はデフォルト設定なので、リクエスト形式が変更されている 場合にのみ必要となります。	

オプション	*value* への設定値	注記
CURLOPT_HTTPPROXYTUNNEL	**TRUE** を設定すると、指定された HTTP プロキシを利用します。	
CURLOPT_MUTE	**TRUE** を設定すると、cURL 関数に関連する出力を完全に抑えます。	
CURLOPT_NETRC	**TRUE** を設定すると、リモートサイトと接続を確立する際に用いる ユーザ名やパスワードを、*~/.netrc* から取得します。	
CURLOPT_NOBODY	**TRUE** を設定すると、出力からの本文を削除します。	
CURLOPT_NOPROGRESS	**TRUE** を設定すると、cURL 転送の進捗状況表示を無効にします。 **注意**: PHP は、このオプションを自動的に **TRUE** に設定します。 これを変更するのは、デバッグ時のみにすべきです。	
CURLOPT_NOSIGNAL	**TRUE** を設定すると、cURL 関数が PHP プロセスに送信するシグナルを 無視します。マルチスレッド SAPI ではデフォルトで on となっており、 そのためタイムアウトオプションもまだ利用されています。	cURL 7.10 および PHP 5.0.0 で追加されました。
CURLOPT_POST	**TRUE** を設定すると、HTTP POST を行います。POST は、*application/x-www-form-urlencoded* 形式で 行われます。これは一般的な HTML のフォームと同じ形式です。	
CURLOPT_PUT	**TRUE** を設定すると、HTTP PUT を行います。PUT するファイルは **CURLOPT_INFILE** および **CURLOPT_INFILESIZE** で指定されている必要があります。	
CURLOPT_RETURNTRANSFER	**TRUE** を設定すると、curl_exec() の返り値を 文字列で返します。通常はデータを直接出力します。	
CURLOPT_SSL_VERIFYPEER	**FALSE** を設定すると、cURL はサーバ証明書の検証を行いません。 別の証明書を **CURLOPT_CAINFO** オプションで 指定するか、**CURLOPT_CAPATH** オプションで 証明ディレクトリを指定します。 **CURLOPT_SSL_VERIFYPEER** を無効にする場合、**CURLOPT_SSL_VERIFYHOST** もまた **TRUE** あるいは **FALSE** にする必要があります（デフォルトは 2 です）。	cURL 7.10 以降、デフォルト値は **TRUE** です。また、cURL 7.10 以降、デフォルトでインストールされています。
CURLOPT_TRANSFERTEXT	**TRUE** を設定すると、FTP 転送を ASCII モードで行います。 LDAP の場合は、データを HTML ではなくプレーンテキストで取得します。 Windows システムでは *STDOUT* に対してバイナリモードを設定しないでください。	
CURLOPT_UNRESTRICTED_AUTH	**TRUE** を設定すると、（**CURLOPT_FOLLOWLOCATION** を利用して）場所をたどっていく際にユーザ名とパスワードを送信し続けます。 これは、たとえホスト名が変わっても続けられます。	PHP 5.0.0 で追加されました。
CURLOPT_UPLOAD	**TRUE** を設定すると、アップロードの準備をします。	
CURLOPT_VERBOSE	**TRUE** を設定すると、詳細な情報を出力します。情報は *STDERR* か、または **CURLOPT_STDERR** で指定したファイルに出力されます。	

value には、 *option* の以下の値に関して 整数値を指定する必要があります。

オプション	*value* への設定値	注記
CURLOPT_BUFFERSIZE	1 回の読み込みに用いるバッファのサイズ。しかしながら、必ず このバッファいっぱいまで読み込まれることを保証するものではありません。	cURL 7.10 および PHP 5.0.0 で追加されました。
CURLOPT_CLOSEPOLICY	*CURLCLOSEPOLICY_LEAST_RECENTLY_USED* あるいは *CURLCLOSEPOLICY_OLDEST* です。 他にも 3 つの *CURLCLOSEPOLICY_* 定数が存在しますが、 cURL はそれらをまだサポートしていません。	
CURLOPT_CONNECTTIMEOUT	接続の試行を待ち続ける秒数。0 は永遠に待ち続けることを意味します。	
CURLOPT_DNS_CACHE_TIMEOUT	DNS エントリをメモリ内に保持し続ける秒数。デフォルトでは 120（2 分）に設定されています。	
CURLOPT_FTPSSLAUTH	（使用可能な場合の）FTP 認証方法。 *CURLFTPAUTH_SSL*（まず SSL を試す）、*CURLFTPAUTH_TLS*（まず TLS を試す）あるいは *CURLFTPAUTH_DEFAULT*（cURL が決める）のいずれかです。	cURL 7.12.2 および PHP 5.1.0 で追加されました。
CURLOPT_HTTP_VERSION	*CURL_HTTP_VERSION_NONE* (デフォルト。 使用するバージョンを決めるのは cURL にまかせる)、 *CURL_HTTP_VERSION_1_0* (HTTP/1.0 を使用する)、 あるいは *CURL_HTTP_VERSION_1_1* (HTTP/1.1 を使用する) のいずれかです。	
CURLOPT_HTTPAUTH	使用する HTTP 認証方法。以下の中から選びます。 *CURLAUTH_BASIC* 、 *CURLAUTH_DIGEST* 、*CURLAUTH_GSSNEGOTIATE* 、 *CURLAUTH_NTLM* 、 *CURLAUTH_ANY* および *CURLAUTH_ANYSAFE* 。 2 つ以上の方法を組み合わせるには、ビット演算子 ¦（or） を使用します。このような場合、	PHP 5.0.0 で追加されました。

オプション	*value* への設定値	注記
	cURL はサーバがサポートしている方法を 問い合わせたうえで最適な方法を選択します。 *CURLAUTH_ANY* は *CURLAUTH_BASIC* ¦ *CURLAUTH_DIGEST* ¦ *CURLAUTH_GSSNEGOTIATE* ¦ *CURLAUTH_NTLM* のエイリアスです。 *CURLAUTH_ANYSAFE* は *CURLAUTH_DIGEST* ¦ *CURLAUTH_GSSNEGOTIATE* ¦ *CURLAUTH_NTLM* のエイリアスです。	
CURLOPT_INFILESIZE	ファイルをリモートサイトにアップロードする際のファイルサイズ。	
CURLOPT_LOW_SPEED_LIMIT	1 秒あたりのバイト数で、転送速度がこれより遅い期間が **CURLOPT_LOW_SPEED_TIME** 秒以上続いた場合に PHP は転送を終了します。	
CURLOPT_LOW_SPEED_TIME	転送速度が **CURLOPT_LOW_SPEED_LIMIT** より遅い期間がどれだけ続いた場合に転送を異常終了させるかを、 秒単位で指定します。	
CURLOPT_MAXCONNECTS	許可される持続的接続の最大数。もしこの値に達した場合、 どの接続を閉じるのかを **CURLOPT_CLOSEPOLICY** によって決定します。	
CURLOPT_MAXREDIRS	HTTP のリダイレクト先を追いかける最大値。 **CURLOPT_FOLLOWLOCATION** とあわせて使用します。	
CURLOPT_PORT	接続先のポート番号。	
CURLOPT_PROXYAUTH	プロキシ接続に使用する HTTP 認証の方法。 **CURLOPT_HTTPAUTH** で説明したのと同じオプションを 指定可能です。プロキシ認証でサポートされているのは、今のところ *CURLAUTH_BASIC* および *CURLAUTH_NTLM* のみです。	cURL 7.10.7 および PHP 5.1.0 で追加されました。
CURLOPT_PROXYPORT	プロキシ接続のポート番号。このポート番号は、 **CURLOPT_PROXY** で指定することも可能です。	PHP 5.0.0 で追加されました。
CURLOPT_PROXYTYPE	*CURLPROXY_HTTP* （デフォルト）あるいは *CURLPROXY_SOCKS5* 。	cURL 7.10 および PHP 5.0.0 で追加されました。
CURLOPT_RESUME_FROM	転送を途中から再開する場合のバイトオフセット。	
CURLOPT_SSL_VERIFYHOST	1 は SSL ピア証明書に一般名が存在するかどうかを調べます。 2 はそれに加え、その名前がホスト名と一致することを検証します。	
CURLOPT_SSLVERSION	使用する SSL のバージョン（2 あるいは 3）。デフォルトでは PHP が自動的に判断しますが、これを手動で設定する必要がある場合もあります。	
CURLOPT_TIMECONDITION	**CURLOPT_TIMEVALUE** の扱いを決定します。 **CURLOPT_TIMEVALUE** で指定した時刻以降に 変更されたページのみを返す場合は *CURL_TIMECOND_IFMODSINCE* を使用します。 **CURLOPT_HEADER** が **TRUE** だと仮定すると、 ページが変更されていない場合は *"304 Not Modified"* ヘッダが返されます。 *CURL_TIMECOND_ISUNMODSINCE* は反対の意味です。 デフォルトは *CURL_TIMECOND_IFMODSINCE* です。	PHP 5.1.0 で追加されました。
CURLOPT_TIMEOUT	cURL 関数の実行にかけられる時間の最大値。	
CURLOPT_TIMEVALUE	1970 年 1 月 1 日からの経過秒数。この値は **CURLOPT_TIMECONDITION** で使用されます。デフォルトでは *CURL_TIMECOND_IFMODSINCE* が設定されます。	

value は、 *option* パラメータの 以下の値に関して文字列である必要があります。

オプション	*value* への設定値	注記
CURLOPT_CAINFO	接続先を検証するための証明書を保持するファイル名。 これは **CURLOPT_SSL_VERIFYPEER** を使用する場合に のみ意味を持ちます。	
CURLOPT_CAPATH	複数の証明書ファイルを保持するディレクトリ。このオプションは **CURLOPT_SSL_VERIFYPEER** とともに使用します。	
CURLOPT_COOKIE	HTTP リクエストにおける *"Set-Cookie: "* ヘッダの内容。	
CURLOPT_COOKIEFILE	クッキーのデータを保持するファイルの名前。クッキーファイルは、 Netscape フォーマットあるいは HTTP ヘッダを単純にファイルにダンプしたものが使用可能です。	
CURLOPT_COOKIEJAR	接続を閉じる際に、すべての内部クッキーを保存するファイルの名前。	
CURLOPT_CUSTOMREQUEST	HTTP リクエストで *"GET"* あるいは *"HEAD"* 以外に 使用するカスタムメソッド。これが有用なのは、 *"DELETE"* やその他のあまり知られていない HTTP リクエストを実行する場合です。 使用可能な値は *"GET"*、 *"POST"*、 *"CONNECT"* などです。 HTTP リクエストの内容を 指定してエラーでなければ正常な応答として扱われます。 *"GET"* を指定しても HTTP リクエストの配送は 行いだということです。	cURL 7.1.1 および PHP 5.0.0 で追加されました。
CURLOPT_HTTPHEADER	設定する HTTP ヘッダフィールドの配列。 **注意**: 使用しようとしているメソッドをサーバがサポートしていることを 確かめるまで、これを使用しないでください。	

value には、 *option* の以下の値に関して 配列を指定する必要があります。

オプション	*value* への設定値	注記
CURLOPT_HTTP200ALIASES	*"HTTP/1.0"* および *"HTTP/1.1"* として扱われる *"HTTP"* ヘッダの配列。	PHP 5.1.3 で追加されました。

オプション	*value* への設定値	注記
CURLOPT_POSTQUOTE	FTP リクエストの実行後に、サーバ上で実行する FTP コマンドの配列。	
CURLOPT_QUOTE	FTP リクエストの前にサーバ上で実行する FTP コマンドの配列。	

value はストリームリソース（例えば fopen() が作成するもの）であり、以下の *option* パラメータに設定します。

オプション	*value* に設定する内容	注釈
CURLOPT_FILE	転送内容が書き込まれるファイル。デフォルトは *STDOUT*（ブラウザウィンドウ）。	
CURLOPT_INFILE	アップロード時に転送内容を読み込むファイル。	
CURLOPT_STDERR	*STDERR* の代わりにエラーを出力する場所。	
CURLOPT_WRITEHEADER	転送のヘッダ部分が書き込まれるファイル。	

value には、 *option* の以下の値に関して 有効なコールバック関数の名前を指定する必要があります。

オプション	*value* への設定値	注記
CURLOPT_HEADERFUNCTION	コールバック関数の名前で、このコールバック関数は 2 つの引数を とります。最初のパラメータは CURL リソースで、2 番目は書き込む ヘッダデータの文字列です。このコールバック関数を使用するにあたり、 ヘッダデータを書き込む処理を実装するのはあなたの役目となります。 書き込んだデータのバイト数を返します。	
CURLOPT_PASSWDFUNCTION	コールバック関数の名前で、このコールバック関数は 3 つの引数を とります。最初のパラメータは CURL リソースで、2 番目はパスワード プロンプトの文字列、そして 3 番目はパスワードの最大長です。 入力されたパスワードを文字列で返します。	
CURLOPT_READFUNCTION	コールバック関数の名前で、このコールバック関数は 2 つの引数を とります。最初のパラメータは cURL リソースで、2 番目は読み込む データの文字列です。このコールバック関数を使用するにあたり、 データを読み込む処理を実装するのはあなたの役目となります。 読み込んだデータのバイト数を返します。*EOF* シグナルを受け取った場合は 0 を返します。	
CURLOPT_WRITEFUNCTION	コールバック関数の名前で、このコールバック関数は 2 つの引数を とります。最初のパラメータは cURL リソースで、2 番目は書き込む データの文字列です。このコールバック関数を使用するにあたり、 データを書き込む処理を実装するのはあなたの役目となります。 書き込んだデータの正確なバイト数を返す必要があり、さもないと この関数は失敗します。	

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 新規に cURL セッションを初期化、ウェブページを取得する

```
<?php
// 新しい cURL リソースを作成します
$ch = curl_init();

// URL その他のオプションを適切に設定します
curl_setopt($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt($ch, CURLOPT_HEADER, false);

// URL の内容を取得し、ブラウザに渡します
curl_exec($ch);

// cURL リソースを閉じ、システムリソースを開放します
curl_close($ch);
?>
```

参考

- curl_setopt_array()

---

curl_version

(PHP 4 >= 4.0.2, PHP 5)

curl_version — cURL のバージョンを返す

説明

array **curl_version** ([int $age])

cURL のバージョンについての情報を返します。

パラメータ

age

返り値

以下の要素からなる連想配列を返します。

version_number	cURL の 24 ビットのバージョン番号
version	cURL バージョン番号を表す文字列
ssl_version_number	OpenSSL の 24 ビットのバージョン番号
ssl_version	OpenSSL バージョン番号を表す文字列
libz_version	zlib バージョンを表す文字列
host	cURL をビルドしたホストについての情報
age	
features	定数 *CURL_VERSION_XXX* のビットマスク
protocols	cURL がサポートするプロトコル名の配列

目次

Cybercash 支払関数

インストール手順

以下の関数は、インタプリタが --with-cybercash=[DIR] を付けて コンパイルされた場合ににみ利用可能です。

PHP 4.3.0 以降、この拡張モジュールは PHP から削除されました。 CyberCash は、現在 » PECL にあります。

CyberCash の最新情報を知りたい場合は、以下の » CyberCash Faq が便利です。 端的にいうと、CyberCash は VeriSign に買収されました。 CyberCash サービス自体は現在も存在するものの、VeriSign は利用者に 対して移行を勧めています。詳細は、上の faq および PECL のリンク先を 参照ください。

---

cybercash_base64_decode

(PHP 4 <= 4.2.3, PECL cybercash:1.18)

cybercash_base64_decode — Cybercash 用に、データの BASE64 デコードを行う

説明

string **cybercash_base64_decode** (string $inbuff)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cybercash_base64_encode

(PHP 4 <= 4.2.3, PECL cybercash:1.18)

cybercash_base64_encode — Cybercash 用に、データの BASE64 エンコードを行う

説明

string **cybercash_base64_encode** (string $inbuff)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cybercash_decr

(PHP 4 <= 4.2.3, PECL cybercash:1.18)

cybercash_decr — Cybercash の復号

説明

array **cybercash_decr** (string $wmk , string $sk , string $inbuff)

情報を復号します。

パラメータ

wmk

merchant キー。

sk

session キー。

inbuff

返り値

この関数は、要素 "errcode" を有する連想配列を返します。 "errcode" が **FALSE** の場合、"outbuff"(文字列)、 "outLth" (long) および "macbuff" (文字列)が配列の要素として返されます。

参考

- cybercash_encr()

---

cybercash_encr

(PHP 4 <= 4.2.3, PECL cybercash:1.18)

cybercash_encr — Cybercash の暗号化

説明

array **cybercash_encr** (string $wmk , string $sk , string $inbuff)

情報を暗号化します。

パラメータ

wmk

merchant キー。

sk

session キー。

inbuff

URL エンコードされた文字列で表す、値のペア。 *order-id=12345&amount=USD+12.34&card_number=.....* のようになります。

返り値

この関数は、要素 "errcode" を有する連想配列を返します。 "errcode" が **FALSE** の場合、"outbuff"(文字列)、 "outLth" (long) および "macbuff" (文字列)が配列の要素として返されます。

参考

- cybercash_decr()

目次

Credit Mutuel CyberMUT 関数

導入

この拡張モジュールにより、 » Credit Mutuel CyberMUT システム を用いたクレジットカードトランザクションが可能となります。

CynerMUT はフランスで有名な Web 支払サービスで、 Credit Mutuel bank により提供されています。フランス人以外には、 これらの関数は有用ではないでしょう。

これらの関数の使用法はほとんど元の関数と同じですが、 CreerFormulaireCM および CreerReponseCM が返すパラメータが異なります。 PHP 関数ではこれらの値は直接返されますが、元の関数ではリファレンスが 渡されます。

> **注意**: この拡張モジュールは Windows 環境では利用できません。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 4.3.0

要件

使用するプラットフォーム用の適当なSDKが必要です。 これは、CyberMUT に登録した後に送られてきます（Web 経由で依頼するか、または近くの Credit Mutuel に行ってみてください）。

> **注意**: これらの関数は、CyberMUT SDK へのリンクを提供するだけです。 必要なパラメータの詳細については、CyberMUT Developers Guide を必ず読んでください。

---

cybermut_creerformulairecm

(PHP 4 >= 4.0.5, PECL cybermut:1.0-1.1)

cybermut_creerformulairecm — 支払リクエスト用 HTML フォームを生成する

説明

string **cybermut_creerformulairecm** (string `$url_cm` , string `$version` , string `$tpe` , string `$price` , string `$ref_command` , string `$text_free` , string `$url_return` , string `$url_return_ok` , string `$url_return_err` , string `$language` , string `$code_company` , string `$text_button`)

cynermut_creerformulairecm() は、支払用 HTML フォームを生成するために使用されます。

参考

- cybermut_testmac()
- cybermut_creerreponsecm()

---

cybermut_creerreponsecm

(PHP 4 >= 4.0.5, PECL cybermut:1.0-1.1)

cybermut_creerreponsecm — 支払を確認し、領収書を生成する

説明

string **cybermut_creerreponsecm** (string `$sentence`)

支払いの確認メッセージが cybermut_testmac() によって正しく認証された場合、パラメータは "OK" です。 他のチェーンは全てエラーメッセージとみなされます。

返り値

支払の領収メッセージを返します。

参考

- cybermut_creerformulairecm()
- cybermut_testmac()

---

cybermut_testmac

(PHP 4 >= 4.0.5, PECL cybermut:1.0-1.1)

cybermut_testmac — 受信した確認用メッセージに虚偽のデータが 含まれていないことを確かめる

説明

bool **cybermut_testmac** (string `$code_mac` , string `$version` , string `$tpe` , string `$cdate` , string `$price` , string `$ref_command` , string `$text_free` , string `$code_return`)

cybermut_testmac() は、受信した確認メッセージに 虚偽のデータが含まれていないことを確かめるために使用されます。 パラメータ *code_return* および *text_free* に注意してください。 これらにはダッシュが含まれるためそのまま評価することができません。

参考

- cybermut_creerformulairecm()
- cybermut_creerreponsecm()

目次

Cyrus IMAP 管理関数

導入

注意: この拡張モジュールは Windows 環境では利用できません。

注意: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP

5.0.0.

インストール手順

Cyrus IMAP サポートを有効にしてこれらの関数を使用するには、 --with-cyrus オプションを指定して PHP をコンパイルする必要があります。

警告

IMAP、recode、 YAZ および Cyrus 拡張モジュールは、組み合わせて使用することはできません。これは、これらすべてが同一の内部シンボルを使用しているためです。

リソース型

この拡張モジュールでは、Cyrus IMAP 接続 ID を定義しています。これは cyrus_connect() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

CYRUS_CONN_NONSYNCLITERAL (integer)
CYRUS_CONN_INITIALRESPONSE (integer)
CYRUS_CALLBACK_NUMBERED (integer)
CYRUS_CALLBACK_NOLITERAL (integer)

---

cyrus_authenticate

(PHP 4 >= 4.0.7, PECL cyrus:1.0)

cyrus_authenticate — Cyrus IMAP サーバに対する認証を行う

説明

void **cyrus_authenticate** (resource `$connection` [, string `$mechlist` [, string `$service` [, string `$user` [, int `$minssf` [, int `$maxssf` [, string `$authname` [, string `$password`]]]]]]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cyrus_bind

(PHP 4 >= 4.0.7, PECL cyrus:1.0)

cyrus_bind — Cyrus IMAP 接続へのコールバックをバインドする

説明

bool **cyrus_bind** (resource `$connection` , array `$callbacks`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cyrus_close

(PHP 4 >= 4.0.7, PECL cyrus:1.0)

cyrus_close — Cyrus IMAP サーバへの接続を閉じる

説明

bool **cyrus_close** (resource `$connection`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cyrus_connect

(PHP 4 >= 4.0.7, PECL cyrus:1.0)

cyrus_connect — Cyrus IMAP サーバに接続する

説明

resource **cyrus_connect** ([string `$host` [, string `$port` [, int `$flags`]]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cyrus_query

(PHP 4 >= 4.0.7, PECL cyrus:1.0)

cyrus_query — Cyrus IMAP サーバへクエリを送信する

説明

array **cyrus_query** (resource `$connection` , string `$query`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

cyrus_unbind

(PHP 4 >= 4.0.7, PECL cyrus:1.0)

cyrus_unbind — アンバインドする ...

説明

bool **cyrus_unbind** (resource `$connection` , string `$trigger_name`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

日付・時刻関数

導入

以下の関数により、PHPスクリプトを実行するサーバから日付と時間を取 得することが可能となります。多くの異なる方法で日付や時間をフォー マットするためにこれらの関数を使用することができます。

> **注意**: これらの関数は、使用するサーバのロケールの設定に依存していること に注意してください。これらの関数を使用する際にはサマータイム (例えば、*.$date += 7*24*60*60*ではなく、 *$date = strtotime('+7 days', $date)* とする) および閏年を必ず考慮に入れるよう

にしてください。

注意: この節で参照しているタイムゾーンは、 サポートされるタイムゾーンのリスト で見つけられます。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

注意: 最新バージョンのタイムゾーンデータベースは、PECL の » timezonedb からインストールできます。Windows ユーザ向けには、コンパイル済みの DLL » php_timezonedb.dll が PECL4Win サイトからダウンロードできます。

実行時設定

php.ini の設定により動作が変化します。

Date/Time Configuration Options

名前	デフォルト	変更の可否	変更履歴
date.default_latitude	"31.7667"	PHP_INI_ALL	PHP 5.0.0 以降で利用可能
date.default_longitude	"35.2333"	PHP_INI_ALL	PHP 5.0.0 以降で利用可能
date.sunrise_zenith	"90.583333"	PHP_INI_ALL	PHP 5.0.0 以降で利用可能
date.sunset_zenith	"90.583333"	PHP_INI_ALL	PHP 5.0.0 以降で利用可能
date.timezone	"GMT"	PHP_INI_ALL	PHP 5.1.0 以降で利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

`date.default_latitude` float

デフォルトの緯度

`date.default_longitude` float

デフォルトの経度

`date.sunrise_zenith` float

デフォルトの日出点

`date.sunset_zenith` float

デフォルトの日没点

`date.timezone` string

環境変数 TZ が設定されていない場合、 全ての日付/時刻関数で使用されるデフォルトのタイムゾーン。 優先順については、date_default_timezone_get() のページで解説されています。サポートしているタイムゾーンについては サポートされるタイムゾーンのリスト を参照ください。

注意: 最初の 4 つの設定オプションは現時点で date_sunrise() と date_sunset() でのみ使用されます。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数は PHP 5.1.1 以降で定義されており、標準的な日付の書式を表します。 日付フォーマット関数（date() など）で使用します。

`DATE_ATOM` (string)
Atom (例: 2005-08-15T15:52:01+00:00)
`DATE_COOKIE` (string)
HTTP クッキー (例: Monday, 15-Aug-05 15:52:01 UTC)

DATE_ISO8601 (string)
ISO-8601 (例: 2005-08-15T15:52:01+0000)
DATE_RFC822 (string)
RFC 822 (例: Mon, 15 Aug 05 15:52:01 +0000)
DATE_RFC850 (string)
RFC 850 (例: Monday, 15-Aug-05 15:52:01 UTC)
DATE_RFC1036 (string)
RFC 1036 (例: Mon, 15 Aug 05 15:52:01 +0000)
DATE_RFC1123 (string)
RFC 1123 (例: Mon, 15 Aug 2005 15:52:01 +0000)
DATE_RFC2822 (string)
RFC 2822 (Mon, 15 Aug 2005 15:52:01 +0000)
DATE_RFC3339 (string)
DATE_ATOM と同じ (PHP 5.1.3 以降)
DATE_RSS (string)
RSS (Mon, 15 Aug 2005 15:52:01 +0000)
DATE_W3C (string)
World Wide Web コンソーシアム (例: 2005-08-15T15:52:01+00:00)

以下の定数は PHP 5.1.2 以降に存在し、関数 date_sunrise() および date_sunset() が返す結果の書式を指定します。

SUNFUNCS_RET_TIMESTAMP (integer)
タイムスタンプ
SUNFUNCS_RET_STRING (integer)
時:分 (例: 08:02)
SUNFUNCS_RET_DOUBLE (integer)
時刻を表す浮動小数点値 (例 8.75)

---

checkdate

(PHP 4, PHP 5)

checkdate — グレグリオ歴の日付/時刻の妥当性を確認します

説明

bool **checkdate** (int $month , int $day , int $year)

引数で指定された日付の妥当性をチェックします。 各パラメータが適切に指定されている場合に、妥当であると判断されます。

パラメータ

month

月は 1 から 12 の間となります。

day

日は、指定された *month* の日数の範囲内になります。*year* がうるう年の場合は、それも考慮されます。

year

年は 1 から 32767 の間となります。

返り値

指定した日付が有効な場合に **TRUE**、そうでない場合に **FALSE** を返します。

例

Example#1 checkdate() の例

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

上の例の出力は以下となります。

```
bool(true)
```

```
bool(false)
```

参考

- mktime()
- strtotime()

---

date_create

(PHP 5 >= 5.1.0)

date_create — 新しい DateTime オブジェクトを返す

説明

DateTime **date_create** ([string `$time` [, DateTimeZone `$timezone`]])
DateTime **DateTime::__construct** ([string `$time` [, DateTimeZone `$timezone`]])

パラメータ

`time`

strtotime() が理解する形式の文字列。デフォルトは "now" です。

`timezone`

その時間のタイムゾーン。

返り値

成功した場合に DateTime オブジェクト、失敗した場合に **`FALSE`** を返します。

---

date_date_set

(PHP 5 >= 5.1.0)

date_date_set — 日付を設定する

説明

void **date_date_set** (DateTime `$object` , int `$year` , int `$month` , int `$day`)
void **DateTime::setDate** (int `$year` , int `$month` , int `$day`)

パラメータ

`object`

DateTime オブジェクト。

`year`

その日付の年。

`month`

その日付の月。

`day`

その日付の日。

返り値

成功した場合に **`NULL`**、失敗した場合に **`FALSE`** を返します。

参考

- date_isodate_set()
- date_time_set()

---

date_default_timezone_get

(PHP 5 >= 5.1.0)

date_default_timezone_get — スクリプト中の日付/時刻関数で使用されるデフォルトタイムゾーンを取得する

説明

string **date_default_timezone_get** (void)

この関数は、デフォルトのタイムゾーンを以下の順で「推測」します。

- date_default_timezone_set() 関数を使用して 設定したタイムゾーン（もし何か設定されていれば）
- *TZ* 環境変数（もし空白でなければ）
- date.timezone ini オプション （設定されていれば）
- 「特殊な」推測（もし OS がサポートしていれば）
- 以上のどの方法でも推測できなかった場合は *UTC* を返します。

返り値

string を返します。

参考

- date_default_timezone_set()

---

date_default_timezone_set

(PHP 5 >= 5.1.0)

date_default_timezone_set — スクリプト中の日付/時刻関数で使用されるデフォルトタイムゾーンを設定する

説明

bool **date_default_timezone_set** (string `$timezone_identifier`)

date_default_timezone_set() は、日付/時刻関数で 使用されるデフォルトタイムゾーンを設定します。

> **注意**: PHP 5.1.0 以降（日付/時刻 関数が書き直されてから）、タイムゾーンを 正しく設定せずに日付/時刻関数をコールすると **E_NOTICE** が発生し、またシステムの設定や *TZ* 環境変数を 使用すると **E_STRICT** が発生するようになりました。

スクリプト内でこの関数を使用してデフォルトタイムゾーンを設定するかわりに、 INI 設定 date.timezone でデフォルトタイムゾーンを設定することもできます。

パラメータ

timezone_identifier

> タイムゾーン ID 。例えば *UTC* や *Europe/Lisbon* など。有効な ID の一覧は サポートされるタイムゾーンのリスト にあります。

返り値

この関数は、*timezone_identifier* が 無効なものである場合に **FALSE**、それ以外の場合に **TRUE** を返します。

変更履歴

バージョン	説明

バージョン	説明
5.1.2	*timezone_identifier* パラメータの内容を 検証するようになりました。

参考

- date_default_timezone_get()

---

date_format

(PHP 5 >= 5.1.0)

date_format — 指定した書式でフォーマットした日付を返す

説明

string **date_format** (DateTime `$object` , string `$format`)
string **DateTime::format** (string `$format`)

パラメータ

object

DateTime オブジェクト。

format

date() が理解できる書式指定文字列。

返り値

成功した場合にフォーマット済みの日付、失敗した場合に **FALSE** を返します。

参考

- date()

---

date_isodate_set

(PHP 5 >= 5.1.0)

date_isodate_set — ISO 日付を設定する

説明

void **date_isodate_set** (DateTime `$object` , int `$year` , int `$week` [, int `$day`])
void **DateTime::setISODate** (int `$year` , int `$week` [, int `$day`])

パラメータ

object

DateTime オブジェクト。

year

その日付の年。

week

その日付の週。

day

その日付の日。

返り値

成功した場合に **NULL**、失敗した場合に **FALSE** を返します。

参考

- date_date_set()

---

date_modify

(PHP 5 >= 5.1.0)

date_modify — タイムスタンプを変更する

説明

void **date_modify** (DateTime `$object` , string `$modify`)
void **DateTime::modify** (string `$modify`)

パラメータ

object

DateTime オブジェクト。

modify

strtotime() が理解できる書式の文字列。

返り値

成功した場合に **NULL**、失敗した場合に **FALSE** を返します。

例

Example#1 date_modify() の例

```
<?php
$date = new DateTime("2006-12-12");
$date->modify("+1 day");
echo $date->format("Y-m-d");
?>
```

上の例の出力は以下となります。

```
2006-12-13
```

参考

- strtotime()

---

date_offset_get

(PHP 5 >= 5.1.0)

date_offset_get — 夏時間のオフセットを返す

説明

int **date_offset_get** (DateTime `$object`)
int **DateTime::getOffset** (void)

パラメータ

object

DateTime オブジェクト。

返り値

成功した場合に DST オフセット秒数、失敗した場合に **FALSE** を返します。

---

date_parse

(PHP 5 >= 5.1.3)

date_parse — 指定した日付に関する詳細な情報を連想配列で返す

説明

array **date_parse** (string `$date`)

パラメータ

date

strtotime() が理解できる書式の日付。

返り値

成功した場合に日付情報を含む配列、 失敗した場合に **FALSE** を返します。

エラー / 例外

日付フォーマットにエラーがある場合は、 'errors' 要素にエラーメッセージが格納されます。

例

Example#1 date_parse() の例

```
<?php
print_r(date_parse("2006-12-12  10:00:00.5"));
?>
```

上の例の出力は以下となります。

```
Array
(
     [year] => 2006
     [month] => 12
     [day] => 12
     [hour] => 10
     [minute] => 0
     [second] => 0
     [fraction] => 0.5
     [warning_count] => 0
     [warnings] => Array()
     [error_count] => 0
     [errors] => Array()
     [is_localtime] =>
)
```

参考

- getdate()

---

date_sun_info

(PHP 5 >= 5.1.2)

date_sun_info — 日の出/日の入り時刻と薄明かり (twilight) の開始/終了時刻の情報を含む配列を返す

説明

array **date_sun_info** (int `$time` , float `$latitude` , float `$longitude`)

パラメータ

time

タイムスタンプ。

latitude

緯度を表す度数。

longitude

経度を表す度数。

返り値

成功した場合に配列、失敗した場合に **FALSE** を返します。

例

Example#1 date_sun_info() の例

```
<?php
$sun_info = date_sun_info(strtotime("2006-12-12"), 31.7667, 35.2333);
foreach ($sun_info as $key => $val) {
    echo "$key: " . date("H:i:s", $val) . "¥n";
}
?>
```

上の例の出力は以下となります。

```
sunrise:  05:52:11
sunset:  15:41:21
transit:  10:46:46
civil_twilight_begin:  05:24:08
civil_twilight_end:  16:09:24
nautical_twilight_begin:  04:52:25
nautical_twilight_end:  16:41:06
astronomical_twilight_begin:  04:21:32
astronomical_twilight_end:  17:12:00
```

参考

- date_sunrise()
- date_sunset()

---

date_sunrise

(PHP 5)

date_sunrise — 指定した日付と場所についての日の出時刻を返す

説明

mixed **date_sunrise** (int $timestamp [, int $format [, float $latitude [, float $longitude [, float $zenith [, float $gmt_offset]]]]])

date_sunrise() は、与えられた日付 (*timestamp* で指定する) と場所についての日の出の時刻を返します。

パラメータ

timestamp

日の出時刻を取得する日の *timestamp* 。

format

***format* 定数**

定数	説明	例
SUNFUNCS_RET_STRING	結果を string で返します。	16:46
SUNFUNCS_RET_DOUBLE	結果を float で返します。	16.78243132
SUNFUNCS_RET_TIMESTAMP	結果を integer (タイムスタンプ) で返します。	1095034606

latitude

デフォルトは北緯で、南緯は負の値で表します。 *date.default_latitude* も参照ください。

longitude

デフォルトは東経で、西経は負の値で表します。 *date.default_longitude* も参照ください。

zenith

デフォルトは *date.sunrise_zenith* です。

gmtoffset

時間単位で指定します。

返り値

日の出時刻を、指定した *format* で返します。 失敗した場合には **FALSE** を返します。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。

例

Example#1 date_sunrise() の例

```
<?php

/* ポルトガル リスボンの日の出時刻を計算する
緯度: 北緯 38.4
経度: 西経 9
天頂 ~= 90
時差: +1 GMT
*/

echo date("D M d Y"). ', sunrise time : ' .date_sunrise(time(), SUNFUNCS_RET_STRING, 38.4, -9, 90, 1);

?>
```

上の例の出力は、たとえば以下のようになります。

```
Mon Dec 20 2004, sunrise time : 08:54
```

参考

- date_sunset()

---

date_sunset

(PHP 5)

date_sunset — 指定した日付と場所についての日の入り時刻を返す

説明

mixed **date_sunset** (int $timestamp [, int $format [, float $latitude [, float $longitude [, float $zenith [, float $gmt_offset]]]]])

date_sunset() は、与えられた日付 (*timestamp* で指定する) と場所についての日の入り時刻を返します。

パラメータ

timestamp

日の入り時刻を取得する日の *timestamp* 。

format

format 定数

定数	説明	例
SUNFUNCS_RET_STRING	結果を string で返します。	16:46
SUNFUNCS_RET_DOUBLE	結果を float で返します。	16.78243132
SUNFUNCS_RET_TIMESTAMP	結果を integer (タイムスタンプ) で返します。	1095034606

latitude

デフォルトは北緯で、南緯は負の値で表します。 *date.default_latitude* も参照ください。

longitude

デフォルトは東経で、西経は負の値で表します。 *date.default_longitude* も参照ください。

zenith

デフォルトは *date.sunrise_zenith* です。

gmtoffset

時間単位で指定します。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。

返り値

日の入り時刻を、指定した *format* で返します。 失敗した場合には **FALSE** を返します。

例

Example#1 date_sunset() の例

```
<?php

/* ポルトガル リスボンの日の入り時刻を計算する
緯度: 北緯 38.4
経度: 西経 9
天頂 ~= 90
時差: +1 GMT
*/

echo date("D M d Y"). ', sunset time : ' .date_sunset(time(), SUNFUNCS_RET_STRING, 38.4, -9, 90, 1);

?>
```

上の例の出力は、たとえば以下のようになります。

```
Mon Dec 20 2004, sunset time : 18:13
```

参考

- date_sunrise()

---

date_time_set

(PHP 5 >= 5.1.0)

date_time_set — 時刻を設定する

説明

void **date_time_set** (DateTime `$object` , int `$hour` , int `$minute` [, int `$second`])
void **DateTime::setTime** (int `$hour` , int `$minute` [, int `$second`])

パラメータ

`object`

DateTime オブジェクト。

`hour`

その時刻の時。

`minute`

その時刻の分。

`second`

その時刻の秒。

返り値

成功した場合に **`NULL`**、失敗した場合に **`FALSE`** を返します。

参考

- date_date_set()

---

date_timezone_get

(PHP 5 >= 5.1.0)

date_timezone_get — 指定した DateTime に関連するタイムゾーンを返す

説明

DateTimeZone **date_timezone_get** (DateTime `$object`)
DateTimeZone **DateTime::getTimezone** (void)

パラメータ

`object`

DateTime オブジェクト。

返り値

成功した場合に DateTimeZone オブジェクト、失敗した場合に **`FALSE`** を返します。

参考

- date_timezone_set()

---

date_timezone_set

(PHP 5 >= 5.1.0)

date_timezone_set — DateTime オブジェクトのタイムゾーンを設定する

説明

void **date_timezone_set** (DateTime `$object` , DateTimeZone `$timezone`)

void **DateTime::setTimezone** (DateTimeZone `$timezone`)

パラメータ

`object`

DateTime オブジェクト。

`timezone`

指定したいタイムゾーン。

返り値

成功した場合に **`NULL`**、失敗した場合に **`FALSE`** を返します。

参考

- date_timezone_get()

---

date

(PHP 4, PHP 5)

date — ローカルの日付/時刻を書式化する

説明

string **date** (string `$format` [, int `$timestamp`])

指定された引数 *`timestamp`* を、与えられた フォーマット文字列によりフォーマットし、日付文字列を返します。 タイムスタンプが与えられない場合は、現在の時刻が使われます。 つまり *`timestamp`* はオプションであり そのデフォルト値は time() の値です。

パラメータ

`format`

出力される日付文字列の書式。以下のオプションを参照ください。

以下の文字が *format* パラメータ文字列として認識されます

format 文字	説明	戻り値の例
日	---	---
d	日。二桁の数字（先頭にゼロがつく場合も）	*01* から *31*
D	曜日。3文字のテキスト形式。	*Mon* から *Sun*
j	日。先頭にゼロをつけない。	*1* から *31*
l (lowercase 'L')	曜日。フルスペル形式。	*Sunday* から *Saturday*
N	ISO-8601 形式の、曜日の数値表現 (PHP 5.1.0 で追加)。	*1*（月曜日）から *7*（日曜日）
S	英語形式の序数を表すサフィックス。2 文字。	*st*, *nd*, *rd* または *th*。 *j*と一緒に使用する ことができる。
w	曜日。数値。	*0* (日曜)から *6* (土曜)
z	年間の通算日。数字。(ゼロから開始)	*0* から *365*
週	---	---
W	ISO-8601 月曜日に始まる年単位の週番号 (PHP 4.1.0 で追加)	例: *42* (年の第 42 週目)
月	---	---
F	月。フルスペルの文字。	*January* から *December*
m	月。数字。先頭にゼロをつける。	*01* から *12*
M	月。3 文字形式。	*Jan* から *Dec*
n	月。数字。先頭にゼロをつけない。	*1* から *12*
t	指定した月の日数。	*28* から *31*
年	---	---
L	閏年であるかどうか。	*1*なら閏年。*0*なら閏年ではない。
o	ISO-8601 形式の年。これは *Y* ほぼ同じだが、ISO 週番号 （*W*）が前年あるいは翌年に属する場合がある点で 異なる（PHP 5.1.0 で追加）。	例: *1999* あるいは *2003*
Y	年。4 桁の数字。	例: *1999*または*2003*
y	年。2 桁の数字。	例: *99* または *03*
時	---	---
a	午前または午後（小文字）	*am* または *pm*
A	午前または午後（大文字）	*AM* または *PM*
B	Swatch インターネット時間	*000* から *999*
g	時。12時間単位。先頭にゼロを付けない。	*1* から *12*
G	時。24時間単位。先頭にゼロを付けない。	*0* から *23*
h	時。数字。12 時間単位。	*01* から *12*
H	時。数字。24 時間単位。	*00* から *23*
i	分。先頭にゼロをつける。	*00* to *59*
s	秒。先頭にゼロをつける。	*00* から *59*
u	ミリ秒 (PHP 5.2.2 で追加)。	例: *54321*
タイムゾーン	---	---
e	タイムゾーン識別子（PHP 5.1.0 で追加）	例: *UTC*, *GMT*, *Atlantic/Azores*
I (capital i)	サマータイム中か否か	*1*ならサマータイム中。 *0*ならそうではない。
O	グリニッジ標準時 (GMT) との時差	例: *+0200*
P	グリニッジ標準時 (GMT) との時差。時間と分をコロンで区切った形式 (PHP 5.1.3 で追加)。	例: *+02:00*
T	タイムゾーンの略称	例: *EST*, *MDT* ...
Z	タイムゾーンのオフセット秒数。 UTC の西側のタイムゾーン用のオフセットは常に負です。そして、 UTC の東側のオフセットは常に正です。	*-43200* から *50400*
全ての日付/時刻	---	---
c	ISO 8601 日付 (PHP 5 で追加されました)	2004-02-12T15:19:21+00:00
r	» RFC 2822 フォーマットされた日付	例: *Thu, 21 Dec 2000 16:01:07 +0200*
U	Unix Epoch (1970 年 1 月 1 日 0 時 0 分 0 秒) からの秒数	time() も参照

フォーマット文字列中の認識されない文字は、そのまま表示されます。 *Z* 形式は、 gmdate() で使用した場合、常に *0* を返します。

> **注意**: この関数が受け付けるのは integer のタイムスタンプだけです。したがって、書式指定文字 *u* が有用となるのは date_create() で作成したタイムスタンプを用いて date_format() を使用した場合のみです。

timestamp

オプションのパラメータ *timestamp* は、 integer 型の Unix タイムスタンプです。 *timestamp* が指定されなかった場合のデフォルト値は、現在の時刻です。言い換えると、デフォルトは time() の返り値となります。

返り値

日付を表す文字列を返します。 *timestamp* に数字以外が使用された場合は **FALSE** が返され、*E_WARNING* レベルのエラーが発生します。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
5.1.0	有効なタイムスタンプの範囲は、通常 Fri, 13 Dec 1901 20:45:54 GMT から Tue, 19 Jan 2038 03:14:07 GMT までです (これらの日付は、32 ビット符号付き整数の最小および最大値に 対応します)。 しかし、PHP 5.1 より前のバージョンでは、システム環境によっては (例: Windows) この範囲が 1970 年 1 月 1 日から 2038 年 1 月 19 日 までに制限されます。
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。
5.1.1	PHP 5.1.1 以降、 *format* パラメータで標準的な 日付/時刻フォーマットを指定する際に有用な 定数がいくつか追加されました。

例

Example#1 date() の例

```
<?php
// 使用するデフォルトのタイムゾーンを指定します。PHP 5.1 以降で使用可能です。
date_default_timezone_set('UTC');


// 結果は、たとえば Monday のようになります。
echo date("l");

// 結果は、たとえば Monday 8th of August 2005 03:12:46 PM のようになります。
echo date('l jS ¥of F Y h:i:s A');

// 結果は July 1, 2000 is on a Saturday となります。
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000));

/* 書式指定パラメータに、定数を使用します。 */
// 結果は、たとえば Mon, 15 Aug 2005 15:12:46 UTC のようになります。
echo date(DATE_RFC822);

// 結果は、たとえば 2000-07-01T00:00:00+00:00 のようになります。
echo date(DATE_ATOM, mktime(0, 0, 0, 7, 1, 2000));
?>
```

前にバックスラッシュを付けてエスケープすることにより、 フォーマット文字列として認識される文字が展開されることを防止することができます。バックスラッシュ付きの文字は既に特別なシーケンスであり、 バックスラッシュもエスケープすることが必要となる可能性があります。

Example#2 date() の文字をエスケープする

```
<?php
// Wednesday the 15th のように出力
echo date("l ¥¥t¥h¥e jS");
?>
```

date() と mktime() の両方を用いて、未来または過去の日付を知ることができます。

Example#3 date() と mktime() の例

```
<?php
$tomorrow  = mktime(0, 0, 0, date("m")  , date("d")+1, date("Y"));
$lastmonth = mktime(0, 0, 0, date("m")-1, date("d"),   date("Y"));
$nextyear  = mktime(0, 0, 0, date("m"),   date("d"),   date("Y")+1);
?>
```

> **注意**: サマータイムがあるため、日付や月の秒数を単純にタイムスタンプに 可減算するよりもより信頼性があります。

date() フォーマットのいくつかの例を示します。 現在の実装で特別な意味がある文字や今後の PHP のバージョンで意味が 割り付けられるであろう文字については、望ましくない結果を避けるために エスケープする必要があることに注意してください。エスケープを する際には、改行文字 \n のような文字を回避するために シングルクォートを使用してください。

Example#4 date() のフォーマット指定

```
<?php
// 今日は March 10th, 2001, 5:16:18 pm であるとします。

$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, n, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 10-03-01, 1631 1618 6 Fripm01
$today = date('¥i¥t ¥i¥s ¥t¥h¥e jS ¥d¥a¥y.');   // It is the 10th day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10 15:16:08 MST 2001
$today = date('H:m:s ¥m ¥i¥s¥ ¥m¥o¥n¥t¥h');     // 17:03:17 m is month
$today = date("H:i:s");                         // 17:16:17
?>
```

他の言語で日付をフォーマットするためには、**date()** のかわりに setlocale() および strftime() 関数を使用する必要があります。

注意

> **注意**: 日付の文字列表現からタイムスタンプを生成するには、 strtotime() が使用できるでしょう。 さらに、いくつかのデータベースは（MySQL の » UNIX_TIMESTAMP 関数の ような）日付フォーマットからタイムスタンプへの変換関数を有しています。

ヒント

PHP 5.1 以降、*$_SERVER['REQUEST_TIME']* によってリクエスト開始時のタイムスタンプが取得できるようになりました。

参考

- getlastmod()
- gmdate()
- mktime()
- strftime()
- time()

---

getdate

(PHP 4, PHP 5)

getdate — 日付/時刻情報を取得する

説明

array **getdate** ([int `$timestamp`])

timestamp に関する日付情報を有する連想配列を返します。 timestamp が指定されない場合は、現在のローカルな時間に関する情報を返します。

パラメータ

timestamp

> オプションのパラメータ *timestamp* は、 integer 型の Unix タイムスタンプです。 *timestamp* が指定されなかった場合のデフォルト値は、現在の時刻です。言い換えると、デフォルトは time() の返り値となります。

返り値

typestamp に関連する情報を連想配列で返します。 返される連想配列の内容は、次のようになります。

返される連想配列のキー

キー	説明	戻り値の例
"seconds"	秒。数値。	*0* から *59*
"minutes"	分。数値。	*0* から *59*
"hours"	時。数値	*0* から *23*
"mday"	月単位の日。数値	*1* から *31*
"wday"	曜日。数値。	*0* (日曜) から *6* (土曜)
"mon"	月。数値。	*1* から *12*
"year"	年。4桁の数値。	例: *1999* あるいは *2003*
"yday"	年単位の日。数値。	*0* から *365*
"weekday"	曜日。フルスペルの文字。	*Sunday* から *Saturday*
"month"	月。フルスペルの文字。	*January* から *December*
0	UNIX時（1970年1月1日）からの秒数。 time()の戻り値と同様。 date() でも使用される。	システムによって違うが、通常は*-2147483648* から *2147483647*.

例

Example#1 getdate() の例

```
<?php
$today  = getdate();
print_r($today);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [seconds] => 40
     [minutes] => 58
     [hours]   => 21
     [mday]    => 17
     [wday]    => 2
     [mon]     => 6
     [year]    => 2003
     [yday]    => 167
      [weekday] => Tuesday
     [month]   => June
     [0]        => 1055901520
)
```

参考

- date()
- time()
- setlocale()

---

gettimeofday

(PHP 4, PHP 5)

gettimeofday — 現在の時間を得る

説明

mixed **gettimeofday** ([bool `$return_float`])

この関数は、gettimeofday(2) へのインターフェイスです。 この関数は、システムコールから返されたデータを有する連想配列を返します。

パラメータ

`return_float`

TRUE を指定すると、配列ではなく float で返します。

返り値

デフォルトでは配列を返します。*return_float* が設定されている場合は float を返します。

配列のキー:

- "sec" - UNIX エポックからの秒
- "usec" - マイクロ秒
- "minuteswest" - グリニッジ基準の分
- "dsttime" - 夏時間補正の型

変更履歴

バージョン	説明
5.1.0	*return_float* パラメータが追加されました。

例

Example#1 gettimeofday() の例

```
<?php
print_r(gettimeofday());

echo gettimeofday(true);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [sec] => 1073504408
     [usec] => 238215
     [minuteswest] => 0
     [dsttime] => 1
)

1073504408.23910
```

---

gmdate

(PHP 4, PHP 5)

gmdate — GMT/UTC の日付/時刻を書式化する

説明

string **gmdate** (string $format [, int $timestamp])

date() 関数と同じですが、返される時刻が グリニッジ標準時 (GMT) であるところが異なります。

パラメータ

format

出力される文字列の書式。date() 関数の書式オプションを参照ください。

timestamp

オプションのパラメータ *timestamp* は、 integer 型の Unix タイムスタンプです。 *timestamp* が指定されなかった場合のデフォルト値は、現在の時刻です。言い換えると、デフォルトは time() の返り値となります。

返り値

日付を表す文字列を返します。 *timestamp* に数字以外が使用された場合は **FALSE** が返され、*E_WARNING* レベルのエラーが発生します。

変更履歴

バージョン	説明
5.1.0	有効なタイムスタンプの範囲は、通常 Fri, 13 Dec 1901 20:45:54 GMT から Tue, 19 Jan 2038 03:14:07 GMT までです (これらの日付は、32 ビット符号付き整数の最小および最大値に 対応します)。 しかし、PHP 5.1 より前のバージョンでは、システム環境によっては (例: Windows) この範囲が 1970 年 1 月 1 日から 2038 年 1 月 19 日 までに制限されます。

バージョン	説明
5.1.1	PHP 5.1.1 以降、 *format* パラメータで標準的な 日付/時刻フォーマットを指定する際に有用な 定数がいくつか追加されました。

例

Example#1 gmdate() の例

フィンランド (GMT +0200) で実行した場合、一行目の出力は "Jan 01 1998 00:00:00"、二行目の出力は "Dec 31 1997 22:00:00" となります。

```
<?php
echo date("M d Y H:i:s", mktime(0, 0, 0, 1, 1, 1998));
echo gmdate("M d Y H:i:s", mktime(0, 0, 0, 1, 1, 1998));
?>
```

参考

- date()
- mktime()
- gmmktime()
- strftime()

---

gmmktime

(PHP 4, PHP 5)

gmmktime — GMT 日付から Unix タイムスタンプを取得する

説明

int **gmmktime** ([int `$hour` [, int `$minute` [, int `$second` [, int `$month` [, int `$day` [, int `$year` [, int `$is_dst`]]]]]]])

指定されるパラメータが GMT 日付を表すこと以外は mktime() と同じです。 **gmmktime()** は内部で mktime() を使用しているので、ローカル時刻として有効な値のみが使用可能です。

mktime() と同様に、 引数は右から順に省略することができます。 省略された引数は、GMT の日付と時刻に従って、現在の値にセットされます。

パラメータ

hour

時

minute

分

second

秒

month

月

day

日

year

年

is_dst

常に GMT 日付を表すので、*is_dst* は結果に影響を及ぼしません。

返り値

Unix タイムスタンプを integer で返します。

変更履歴

バージョン	説明
5.1.0	PHP 5.1.0 で、*is_dst* パラメータは廃止されました。 その代わりに、新しいタイムゾーン処理機能が使用されます。

例

Example#1 Windows 環境での gmmktime()

```
<?php
gmmktime(0, 0, 0, 1, 1, 1970); // valid in GMT and west, invalid in east
?>
```

参考

- mktime()
- date()
- time()

---

gmstrftime

(PHP 4, PHP 5)

gmstrftime — ロケールの設定に基づいて GMT/UTC 時刻/日付をフォーマットする

説明

string **gmstrftime** (string `$format` [, int `$timestamp`])

グリニッジ標準時を返すこと以外は、 strftime() と同じ動作をします。例えば、東部標準時 (GMT -0500) で実行した場合、以下の最初の行は "Dec 31 1998 20:00:00" を出力し、二行目は "Jan 01 1999 01:00:00" を出力します。

例

Example#1 gmstrftime() の例

```
<?php
setlocale(LC_TIME, 'en_US');
echo strftime("%b %d %Y %H:%M:%S", mktime(20, 0, 0, 12, 31, 98)) . "¥n";
echo gmstrftime("%b %d %Y %H:%M:%S", mktime(20, 0, 0, 12, 31, 98)) . "¥n";
?>
```

参考

- strftime()

---

idate

(PHP 5)

idate — ローカルな時刻/日付を整数として整形する

説明

int **idate** (string `$format` [, int `$timestamp`])

指定された引数 *timestamp* を、 与えられたフォーマット文字列によりフォーマットし、日付数値を返します。 タイムスタンプが与えられない場合は、現在のローカル時刻が使われます。 つまり、*timestamp* はオプションであり、 そのデフォルト値は time() の値です。

関数 date() と異なり、**idate()** は *format* パラメータ中は一文字しか受け取りません。

パラメータ

format

以下の文字が *format* パラメータ文字列として認識されます

format 文字	説明
B	Swatch ビート/インターネット時間
d	月の日
h	時 (12 時間単位)
H	時 (24 時間単位)
i	分
I	夏時間の適用中は *1*、 そうでなければ *0* を返す
L	閏年なら *1*、 そうでなければ *0* を返す
m	月数
s	秒
t	現在の月の日数
U	Unix Epoch (January 1 1970 00:00:00 GMT) からの秒数。 これは time() と同じです
w	曜日 (日曜日は *0*)
W	ISO-8601 形式。月曜日から始まる年単位の週番号
y	年 (1 桁あるいは 2 桁の数値 - 下の「注意」を確認ください)
Y	年 (4 桁)
z	年間の通算日
Z	タイムゾーンのオフセット秒数

timestamp

オプションのパラメータ *timestamp* は、 integer 型の Unix タイムスタンプです。 *timestamp* が指定されなかった場合のデフォルト値は、現在の時刻です。言い換えると、デフォルトは time() の返り値となります。

返り値

整数値を返します。

idate() が返す値の型は常に integer であり、 先頭に "0" がくることはありません。そのため、**idate()** の返す結果が予想より少ない桁数になることもあります。以下の例を参照ください。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。

例

Example#1 idate() の例

```
<?php
$timestamp  = strtotime('1st  January  2004'); //1072915200

// これは、年を 2 桁で表示します。
// しかし、ここでは年が "0" から始まるので、
// "4" だけが表示されます。
echo idate('y', $timestamp);
?>
```

参考

- date()
- time()

---

localtime

(PHP 4, PHP 5)

localtime — ローカルタイムを得る

説明

array **localtime** ([int `$timestamp` [, bool `$is_associative`]])

localtime() 関数は、C 言語の 同名の関数コールにより返される構造体と同じ内容の配列を返します。

パラメータ

`timestamp`

オプションのパラメータ *`timestamp`* は、 integer 型の Unix タイムスタンプです。 *`timestamp`* が指定されなかった場合のデフォルト値は、現在の時刻です。言い換えると、デフォルトは time() の返り値となります。

`is_associative`

FALSE に設定されるか省略された場合は、 配列は通常の数値を添字とした配列として返されます。 **TRUE** に設定された場合は、 **localtime()** は C 言語の関数 localtime のコールにより返される構造体の全ての要素を有する連想配列を返します。 この連想配列の各キーの名前は次のようになります。

- "tm_sec" - 秒
- "tm_min" - 分
- "tm_hour" - 時
- "tm_mday" - 月の日付 月は 0 (1 月) から 11 (12 月) で、曜日は 0 (日曜日) から 6 (土曜日) となります。
- "tm_mon" - 月。0 が 1 月を表します。
- "tm_year" - 1900 年からの年数。
- "tm_wday" - 曜日。
- "tm_yday" - 年単位の日付。
- "tm_isdst" - 夏時間の適用中かどうか。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。

例

Example#1 time() の例

```
<?php
$localtime  = localtime();
$localtime_assoc  = localtime(time(), true);
print_r($localtime);
print_r($localtime_assoc);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => 24
     [1] => 3
     [2] => 19
     [3] => 3
     [4] => 3
     [5] => 105
     [6] => 0
     [7] => 92
     [8] => 1
)

Array
(
     [tm_sec] => 24
     [tm_min] => 3
     [tm_hour] => 19
     [tm_mday] => 3
     [tm_mon] => 3
     [tm_year] => 105
     [tm_wday] => 0
     [tm_yday] => 92
```

```
    [tm_isdst] => 1
)
```

---

microtime

(PHP 4, PHP 5)

microtime — 現在の Unix タイムスタンプをマイクロ秒まで返す

説明

mixed **microtime** ([bool `$get_as_float`])

microtime() は、現在の Unix タイムスタンプをマイクロ秒単位で返します。 この関数は、gettimeofday() システムコールをサポートする オペレーティングシステムでのみ使用できます。

パラメータ

`get_as_float`

オプション引数を付けずにコールされた場合、この関数は文字列 "msec sec" を返します。ただし、sec は Unix エポック (1970 年 1 月 1 日 0:00:00 GMT) から計算した秒数、msec はマイクロ秒の部分です。 文字列のそれぞれの部分は秒単位で返されます。

`get_as_float` を **`TRUE`** に設定すると、結果を float (秒単位) で返します。

変更履歴

バージョン	説明
5.0.0	*`get_as_float`* パラメータが追加されました。

例

Example#1 microtime() でタイマスクリプト実行

```php
<?php
/**
 * PHP 5の動作を模擬する簡単な関数
 */
function microtime_float()
{
    list($usec, $sec) = explode(" ", microtime());
    return ((float)$usec + (float)$sec);
}

$time_start = microtime_float();

// しばらくスリープ
usleep(100);

$time_end = microtime_float();
$time = $time_end - $time_start;

echo "Did nothing in $time seconds¥n";
?>
```

Example#2 PHP 5 におけるタイマスクリプト実行

```php
<?php
$time_start = microtime(true);

// しばらくスリープ
usleep(100);

$time_end = microtime(true);
$time = $time_end - $time_start;

echo "Did nothing in $time seconds¥n";
?>
```

参考

- time()

---

mktime

(PHP 4, PHP 5)

mktime — 日付を Unix のタイムスタンプとして取得する

説明

int **mktime** ([int $hour [, int $minute [, int $second [, int $month [, int $day [, int $year [, int $is_dst]]]]]]])

与えられた引数に従って UNIX のタイムスタンプを返します。 このタイムスタンプは、Unix epoch(1970年1月1日00:00:00 GMT)から 指定された時刻までの通算秒を表す長整数です。

引数は右から順に省略することができます。省略された引数は、 ローカルの日付と時刻に従って、現在の値にセットされます。

パラメータ

hour

時

minute

分

second

秒

month

月

day

日

year

年。2 桁または 4 桁の値を指定可能で、 0-69 の間の値は 2000-2069 に、70-100 は 1970-2000 にマップされます。 今日最も一般的なシステム、すなわち time_t が 32 ビットの符号付き整数である システムでは *year* として有効な範囲は 1901 から 2038 の間です。しかし、PHP 51.0 より前のバージョンではこの範囲が 1970 から 2038 に制限されているシステム (たとえば Windows) もありました。

is_dst

このパラメータはサマータイム (DST) の時に1にセットされ、 そうでない時に0、サマータイムであるかどうかが不明である場合に-1 にセットされます。 不明な場合、PHP はサマータイムかどうか推測しようとします。 PHP を実行しているシステムで DST が有効になっている、もしくは *is_dst* が 1 にセットされている場合、 一部の時刻は有効になりません。 もし DST が有効で例えば 2:00 の場合、2:00 から 3:00 までの全ての時刻は 無効になり、**mktime()** は不確定な値 (通常は負数) を返します。いくつかのシステム (例えば Solaris 8) は真夜中に DST を有効にしますので、DST が有効なときの 0:30 は前日の 23:30 と評価されます。

注意: PHP 5.1.0 以降、このパラメータは廃止されました。その結果、 新しいタイムゾーン処理機能がかわりに使用されます。

返り値

mktime() は与えられた引数の Unix タイムスタンプを返します。引数が不正な場合、 この関数は **FALSE** を返します（PHP 5.1 より前のバージョンでは *-1* を返していました）。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
3.0.10	*is_dst* パラメータが追加されました
5.1.0	*is_dst* パラメータは廃止されました。 エラー時には *-1* ではなく **FALSE** を返すようになりました。 年月日がすべてゼロとすることが可能になりました。
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。

例

Example#1 mktime() の例

mktime() は入力日付の有効性を確認しており、 範囲外の入力を自動的に修正して計算してくれるので便利です。 例えば、以下の各行はいずれも文字列 "Jan-01-1998"を出力します。

```
<?php
echo date("M-d-Y", mktime(0, 0, 0, 12, 32, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 13, 1, 1997));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 1998));
echo date("M-d-Y", mktime(0, 0, 0, 1, 1, 98));
?>
```

Example#2 次月の最後の日

指定した月の最後の日は次の月の -1 番目の日ではなく、"0" 番目の日 として表すことが可能です。以下の例はともに文字列"The last day in Feb 2000 is: 29" を出力します。

```
<?php
$lastday = mktime(0, 0, 0, 3, 0, 2000);
echo strftime("Last day in Feb 2000 is: %d", $lastday);
$lastday = mktime(0, 0, 0, 4, -31, 2000);
echo strftime("Last day in Feb 2000 is: %d", $lastday);
?>
```

注意

警告

PHP 5.1.0 より前のバージョンでは、負の値のタイムスタンプは Windows の どのバージョンにおいてもサポートされていませんでした。 したがって、年の有効範囲は 1970 年から 2038 年の間のみに限定されていました。

参考

- gmmktime()
- date()
- time()

---

strftime

(PHP 4, PHP 5)

strftime — ロケールの設定に基づいてローカルな日付・時間をフォーマットする

説明

string **strftime** (string `$format` [, int `$timestamp`])

指定した *timestamp* または timestamp が指定されていない場合に現在のローカル時間を用いて、 指定したフォーマット文字列に基づき文字列をフォーマットして返します。 月および曜日の名前、およびその他の言語依存の文字列は、 setlocale() で設定された現在のロケールを尊重して表示されます。

使用する C ライブラリで、すべての変換指定子がサポートされているとは限りません。 そのような場合、該当する変換指定子は PHP の **strftime()** ではサポートされません。 また、全てのプラットフォームが負のタイムスタンプをサポートしているわけではないため、 Unix Epoch (1970 年 1 月 1 日）以前の日付を扱えないかもしれません。 つまり、Windows や一部の Linux ディストリビューション、そしてその他のごく一部の オペレーティングシステム上では %e、%T、%R および %D (あるいはこれ以外も) が *Jan 1, 1970* より前の日付について動作しないということです。 Windwos システム上でサポートされる変換指定子の概要については、 » MSDN のウェブサイト に掲載されています。

パラメータ

format

次の変換指定子は、フォーマット文字列として認識されます。

- %a - 現在のロケールに基づく短縮された曜日の名前
- %A - 現在のロケールに基づく完全な曜日の名前
- %b - 現在のロケールに基づく短縮された月の名前
- %B - 現在のロケールに基づく完全な月の名前
- %c - 現在のロケールに基づく適当な日付と時間の表現
- %C - 世紀 (年を 100 で割り、整数に丸めたもの。00 から 99)
- %d - 日付を 10 進数で (01 から 31)。
- %D - %m/%d/%y と同じ
- %e - 月単位の日付を 10 進数で表したもの。日付が 1 桁の場合は、 前に空白を一つ付けます (' 1' から '31')。

- %g - 世紀以外は %G と同じ。
- %G - ISO 週番号 (%V を参照) に対応する 4 桁の年。 これは ISO 週番号が前年もしくは次年に属するかによって使用される年が 異なる事を除き %Y と同じフォーマットと値です。
- %h - %bと同じ。
- %H - 時間を 24 時間表示の 10 進数で (00 から 23 まで)。
- %I - 時間を 12 時間表示の 10 進数で (01 から 12 まで)。
- %j - 年間での日付を 10 進数で表現 (001 から 366)。
- %m - 月を 10 進数で表現 (01 から 12)。
- %M - 分を 10 進数で表現。
- %n - 改行文字。
- %p - 指定した時間により `am' または `pm'、 または現在のロケールの、それに対応する文字列。
- %r - a.m. および p.m. 表記で表した時間。
- %R - 24 時間表記で表した時間。
- %S - 秒を 10 進数で表現。
- %t - タブ文字。
- %T - 現在の時間。%H:%M:%S に等しい。
- %u - 10 進数表記の曜日。1 から 7 の範囲で表し、1 が月曜日。

  警告

  Sun Solaris では日曜日を 1 と表しているようですが、 ISO 9889:1999 (現在の C 言語の標準規格) では「1 は月曜日」と明確に指定されています。

- %U - 年間で何番目の週であるかを 10 進数で表現。 年間で最初の日曜を最初の週の最初の日として数えます。
- %V - ISO 8601:1988 で規定された現在の年の週番号の 10 進数表現で、 01 から 53 までの範囲となります。1 は最初の週で、その週は現在の年に 最低 4 日はあります。週は月曜日から始まります (指定したタイムスタンプの週番号に対応する年を表すには、 %G あるいは %g をしてください)。
- %W - 現在の年で何番目の週であるかを 10 進数で表現。 年間で最初の月曜を最初の週の最初の日として数えます。
- %w - 曜日を 10 進数で表現。日曜は 0 になります。
- %x - 時間を除いた日付を現在のロケールに基づき表現します。
- %X - 日付を除いた時間を現在のロケールに基づき表現します。
- %y - 世紀の部分を除いた年を10進数として表現 (00 から 99 までの範囲)。
- %Y - 世紀を含む年を 10 進数で表現。
- %Z あるいは %z - タイムゾーンまたはその名前または短縮形。
- %% - 文字リテラル`%'。

このパラメータの最大長は 1023 文字です。

timestamp

オプションのパラメータ *timestamp* は、 integer 型の Unix タイムスタンプです。 *timestamp* が指定されなかった場合のデフォルト値は、現在の時刻です。言い換えると、デフォルトは time() の返り値となります。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。

例

この例は、それぞれのロケールがシステムにインストールされている場合にのみ動作します。

Example#1 strftime() のロケールの例

```
<?php
setlocale(LC_TIME, "C");
echo strftime("%A");
setlocale(LC_TIME, "fi_FI");
echo strftime(" in Finnish is %A,");
setlocale(LC_TIME, "fr_FR");
echo strftime(" in French %A and");
setlocale(LC_TIME, "de_DE");
echo strftime(" in German %A.¥n");
?>
```

Example#2 ISO 8601:1988 の週番号の例

```
<?php
```

```
/*       December 2002 / January 2003
ISOWk   M    Tu   W    Thu F    Sa   Su
-----  ----------------------------
51       16   17   18   19   20   21   22
52       23   24   25   26   27   28   29
1        30   31    1    2    3    4    5
2         6    7    8    9   10   11   12
3        13   14   15   16   17   18   19    */

// Outputs: 12/28/2002 - %V,%G,%Y = 52,2002,2002
echo "12/28/2002 - %V,%G,%Y = " . strftime("%V,%G,%Y", strtotime("12/28/2002")) . "¥n";

// Outputs: 12/30/2002 - %V,%G,%Y = 1,2003,2002
echo "12/30/2002 - %V,%G,%Y = " . strftime("%V,%G,%Y", strtotime("12/30/2002")) . "¥n";

// Outputs: 1/3/2003 - %V,%G,%Y = 1,2003,2003
echo "1/3/2003 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/3/2003")) . "¥n";

// Outputs: 1/10/2003 - %V,%G,%Y = 2,2003,2003
echo "1/10/2003 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/10/2003")) . "¥n";


/*       December 2004 / January 2005
ISOWk   M    Tu   W    Thu F    Sa   Su
-----  ----------------------------
51       13   14   15   16   17   18   19
52       20   21   22   23   24   25   26
53       27   28   29   30   31    1    2
1         3    4    5    6    7    8    9
2        10   11   12   13   14   15   16    */

// Outputs: 12/23/2004 - %V,%G,%Y = 52,2004,2004
echo "12/23/2004 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("12/23/2004")) . "¥n";

// Outputs: 12/31/2004 - %V,%G,%Y = 53,2004,2004
echo "12/31/2004 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("12/31/2004")) . "¥n";

// Outputs: 1/2/2005 - %V,%G,%Y = 53,2004,2005
echo "1/2/2005 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/2/2005")) . "¥n";

// Outputs: 1/3/2005 - %V,%G,%Y = 1,2005,2005
echo "1/3/2005 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/3/2005")) . "¥n";

?>
```

注意

> **注意**: ISO 8601:1988 week numbers に基づいている %G と %V は、 もしナンバリングシステムが完全に理解されていなければ 期待とは違う結果をもたらします。 このマニュアルの %V の例を参照ください。

参考

- setlocale()
- mktime()
- strptime()
- » Open Group による **strftime()** の定義

---

strptime

(PHP 5 >= 5.1.0)

strptime — strftime() が生成した日付/時刻をパースする

説明

array **strptime** (string `$date` , string `$format`)

strptime() は *date* をパースした結果を配列で返します。エラー時には **FALSE** を返します。

月名や曜日、そしてその他の言語依存な文字列は setlocale() (LC_TIME) で設定された現在の ロケールを考慮して返します。

> **注意**: この関数は Windows 環境にはまだ実装されていません。

パラメータ

date (string)

> パースする文字列（例: strftime() の返り値）。

format (string)

date で使用されているフォーマット （例: strftime() で使用されていたものと同じ）。

フォーマットオプションについての詳細は strftime() のページを参照ください。

返り値

成功した場合は配列、失敗した場合は **FALSE** を返します。

配列で返されるパラメータ

パラメータ	説明
tm_sec	分未満の秒数 (0-61)
tm_min	時間未満の分数 (0-59)
tm_hour	0 時以降の時間数 (0-23)
tm_mday	月内の日数 (1-31)
tm_mon	1 月から数えた月数 (0-11)
tm_year	1900 年以降の年
tm_wday	日曜日からの日数 (0-6)
tm_yday	1 月 1 日からの日数 (0-365)
unparsed	*date* の中で、指定された *format* で判断できなかった部分

例

Example#1 strptime() の例

```
<?php
$format = '%d/%m/%Y %H:%M:%S';
$strf = strftime($format);

echo "$strf¥n";

print_r(strptime($strf, $format));
?>
```

上の例の出力は、たとえば以下のようになります。

```
03/10/2004 15:54:19

Array
(
    [tm_sec] => 19
    [tm_min] => 54
    [tm_hour] => 15
    [tm_mday] => 3
    [tm_mon] => 9
    [tm_year] => 104
    [tm_wday] => 0
    [tm_yday] => 276
    [unparsed] =>
)
```

参考

- strftime()

---

strtotime

(PHP 4, PHP 5)

strtotime — 英文形式の日付を Unix タイムスタンプに変換する

説明

int **strtotime** (string $time [, int $now])

この関数は US 英文形式の日付を含む文字列が指定されることを期待しており、 *now* で与えられたその形式から Unix タイムスタンプ (1970 年 1 月 1 日 00:00:00 GMT からの経過秒数) への変換を試みます。 *now* が指定されていない場合は現在日時に変換します。

この関数は、タイムスタンプを計算するために (利用可能であれば) 環境変数 TZ を使用します。PHP 5.1.0 以降では、日付/時刻関数で 使用されるタイムゾーンを設定する簡単な方法があります。 その方法については date_default_timezone_get() 関数のページを参照ください。

注意: 年を 2 桁の数値で指定した場合、その値が 00-69 なら 2000-2069 に、 70-99 なら 1970-1999 にそれぞれ変換されます。

パラメータ

time

パースする文字列で、GNU » Date Input Formats 形式に準拠したもの。PHP 5.0.0 より前のバージョンではマイクロ秒を 含めることはできませんでした。PHP 5.0.0 以降ではマイクロ秒を 含めることが可能ですが、その値は無視されます。

now

返される値を計算するために使用されるタイムスタンプ。

返り値

成功時はタイムスタンプ、そうでなければ **FALSE** を返します。 PHP 5.1.0 以前ではこの関数は失敗時に *-1* を返します。

エラー / 例外

すべての日付/時刻関数は、有効なタイムゾーンが設定されていない場合に **E_NOTICE** を発生させます。また、システム設定のタイムゾーンあるいは環境変数 *TZ* を使用した場合には **E_STRICT** を発生させます。 date_default_timezone_set() も参照ください。

変更履歴

バージョン	説明
5.1.0	失敗時に *-1* の代わりに **FALSE** を返すようになりました。
5.1.0	タイムゾーンがおかしい場合に **E_STRICT** や **E_NOTICE** が発生するようになりました。

例

Example#1 A strtotime() の例

```
<?php
echo strtotime("now"), "¥n";
echo strtotime("10 September 2000"), "¥n";
echo strtotime("+1 day"), "¥n";
echo strtotime("+1 week"), "¥n";
echo strtotime("+1 week 2 days 4 hours 2 seconds"), "¥n";
echo strtotime("next Thursday"), "¥n";
echo strtotime("last Monday"), "¥n";
?>
```

Example#2 失敗のチェック

```
<?php
$str = 'Not Good';

// PHP 5.1.0 以前では、false の代わりに -1 と比較する
if (($timestamp = strtotime($str)) === false) {
    echo "The string ($str) is bogus";
} else {
    echo "$str == " . date('l dS ¥o¥f F Y h:i:s A', $timestamp);
}
?>
```

注意

警告

5.0.2 までの PHP 5 では、*"now"* と他の関連する時刻は 誤って当日の真夜中から計算されます。 他のバージョンでは、これは正しく現在時刻から計算されます。

警告

バージョン 4.4.0 より前の PHP では、*"next"* が誤って +2 として計算されます。これを解決するには、代わりに *"+1"* を使用します。

注意: タイムスタンプの有効な範囲は、通常、Fri, 13 Dec 1901 20:45:54 GMT から Tue, 19 Jan 2038 03:14:07 GMT までです (これらは、32 ビット符号付整数の最大及び最小に一致します)。 さらに全てのプラットフォームが負のタイムスタンプをサポートしているわけではありませんので、日付の範囲は Unix エポック以前にはならないかも知れません。 これは、例えば Windows やいくつかの Linux ディストリビューション、 いくつかの他のオペレーティングシステムでは 1970 年 1 月 1 日以前の日付では動作しない事を意味しています。 PHP 5.1.0 以降のバージョンでは、この制限はなくなっています。

参考

- strptime()

time

(PHP 4, PHP 5)

time — 現在の Unix タイムスタンプを返す

説明

int **time** (void)

現在時刻を Unix エポック (1970 年 1 月 1 日 00:00:00 GMT) からの通算秒として返します。

例

Example#1 time() の例

```
<?php
$nextWeek  = time() + (7 * 24 * 60 * 60);
                  // 7 日 * 24 時間 * 60 分 * 60 秒
echo 'Now:         '. date('Y-m-d') ."¥n";
echo 'Next  Week:  '. date('Y-m-d', $nextWeek) ."¥n";
// あるいは strtotime() を使用します
echo 'Next  Week:  '. date('Y-m-d', strtotime('+1  week')) ."¥n";
?>
```

上の例の出力は、たとえば以下のようになります。

```
Now:         2005-03-30
Next  Week:  2005-04-06
Next  Week:  2005-04-06
```

注意

ヒント

PHP 5.1 以降、*$_SERVER['REQUEST_TIME']* によってリクエスト開始時のタイムスタンプが取得できるようになりました。

参考

- date()
- microtime()

---

timezone_abbreviations_list

(PHP 5 >= 5.1.0)

timezone_abbreviations_list — 夏時間、オフセットおよびタイムゾーン名を含む連想配列を返す

説明

array **timezone_abbreviations_list** (void)
array **DateTimeZone::listAbbreviations** (void)

返り値

成功した場合に配列、失敗した場合に **FALSE** を返します。

例

Example#1 timezone_abbreviations_list() の例

```
<?php
$timezone_abbreviations  = DateTimeZone::listAbbreviations();
print_r($timezone_abbreviations["acst"]);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [0] => Array
        (
            [dst] => 1
```

```
            [offset] => -14400
             [timezone_id] => America/Porto_Acre
        )

    [1] => Array
        (
            [dst] => 1
             [offset] => -14400
              [timezone_id] => America/Eirunepe
        )

    [2] => Array
        (
            [dst] => 1
             [offset] => -14400
              [timezone_id] => America/Rio_Branco
        )

    [3] => Array
        (
            [dst] => 1
             [offset] => -14400
              [timezone_id] => Brazil/Acre
        )

)
```

参考

- timezone_identifiers_list()

---

timezone_identifiers_list

(PHP 5 >= 5.1.0)

timezone_identifiers_list — すべてのタイムゾーン識別子を含む配列を返す

説明

array **timezone_identifiers_list** (void)
array **DateTimeZone::listIdentifiers** (void)

返り値

成功した場合に配列、失敗した場合に **FALSE** を返します。

例

Example#1 timezone_identifiers_list() の例

```php
<?php
$timezone_identifiers  = DateTimeZone::listIdentifiers();
for ($i=0; $i < 5; $i++) {
    echo "$timezone_identifiers[$i]¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmera
```

参考

- timezone_abbreviations_list()

---

timezone_name_from_abbr

(PHP 5 >= 5.1.3)

timezone_name_from_abbr — 略称からタイムゾーン名を返す

説明

string **timezone_name_from_abbr** (string $abbr [, int $gmtOffset [, int $isdst]])

パラメータ

abbr

タイムゾーンの略称。

gmtOffset

GMT からのオフセット秒数。デフォルトは -1 で、 この場合は *abbr* に対応するタイムゾーンのうち最初に見つかったものを返します。 それ以外の場合は指定したオフセットを探し、 そこで見つからなかった場合には 他のオフセットも含めて最初に見つかったものを返します。

isdst

夏時間の識別子。*abbr* が存在しなかった場合は、 *offset* と *isdst* をもとにタイムゾーンを探します。

返り値

成功した場合にタイムゾーン名、失敗した場合に **FALSE** を返します。

例

Example#1 timezone_name_from_abbr() の例

```
<?php
echo timezone_name_from_abbr("CET") . "¥n";
echo timezone_name_from_abbr("", 3600, 0) . "¥n";
?>
```

上の例の出力は、たとえば以下のようになります。

```
Europe/Berlin
Europe/Paris
```

参考

- timezone_abbreviations_list()

---

timezone_name_get

(PHP 5 >= 5.1.0)

timezone_name_get — タイムゾーンの名前を返す

説明

string **timezone_name_get** (DateTimeZone $object)
string **DateTimeZone::getName** (void)

パラメータ

object

DateTimeZone オブジェクト。

返り値

成功した場合にタイムゾーン名、失敗した場合に **FALSE** を返します。

---

timezone_offset_get

(PHP 5 >= 5.1.0)

timezone_offset_get — GMT からのタイムゾーンのオフセットを返す

説明

int **timezone_offset_get** (DateTimeZone `$object` , DateTime `$datetime`)
int **DateTimeZone::getOffset** (DateTime `$datetime`)

この関数は、*datetime* パラメータで指定した 日付/時刻 についての GMT へのオフセットを返します。GMT オフセットの計算の際には、 使用する DateTime オブジェクトに含まれるタイムゾーン情報を使用します。

パラメータ

object

DateTimeZone オブジェクト。

datetime

オフセットを計算する 日付/時刻 を含む DateTime。

返り値

成功した場合にタイムゾーンのオフセット秒数、失敗した場合に **FALSE** を返します。

例

Example#1 timezone_offset_get() の例

```
<?php
// ふたつのタイムゾーンオブジェクトを作成します。ひとつは台北（台湾）、
// そしてもうひとつは東京（日本）のものです。
$dateTimeZoneTaipei  = new DateTimeZone("Asia/Taipei");
$dateTimeZoneJapan  = new DateTimeZone("Asia/Tokyo");

// 同一の Unix タイムスタンプを持つふたつの DateTime オブジェクトを作成します。
// しかしアタッチするタイムゾーンはそれぞれ異なります。
$dateTimeTaipei  = new DateTime("now", $dateTimeZoneTaipei);
$dateTimeJapan  = new DateTime("now", $dateTimeZoneJapan);

// $dateTimeTaipei オブジェクトに含まれる 日付/時刻 の GMT オフセットを計算します。
// しかし、タイムゾーンの規則は東京のもの ($dateTimeZoneJapan)
// を使用します。
$timeOffset  = $dateTimeZoneJapan->getOffset($dateTimeTaipei);

// これは int(32400) となります (Sat Sep 8 01:00:00 1951 JST 以降の日付の場合)。
var_dump($timeOffset);
?>
```

---

timezone_open

(PHP 5 >= 5.1.0)

timezone_open — 新しい DateTimeZone オブジェクトを返す

説明

DateTimeZone **timezone_open** (string `$timezone`)
DateTimeZone **DateTimeZone::__construct** (string `$timezone`)

パラメータ

timezone

タイムゾーンを表す正式名 (例: Europe/Prague) あるいは略称 (例: CET)。

返り値

成功した場合に DateTimeZone オブジェクト、失敗した場合に **FALSE** を返します。

---

timezone_transitions_get

(PHP 5 >= 5.2.0)

timezone_transitions_get — タイムゾーンの変遷を返す

説明

array **timezone_transitions_get** (DateTimeZone `$object`)
array **DateTimeZone::getTransitions** (void)

パラメータ

object

DateTimeZone オブジェクト。

返り値

成功した場合にタイムゾーンの遷移を表す連想配列の配列、 失敗した場合に **FALSE** を返します。

例

Example#1 timezone_transitions_get() の例

```
<?php
$timezone  = new DateTimeZone("CET");
print_r(reset($timezone->getTransitions()));
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [ts] => -1693706400
     [time] => 1916-04-30T22:00:00+0000
    [offset] => 7200
    [isdst] => 1
    [abbr] => CEST
)
```

目次

- time — 現在の Unix タイムスタンプを返す
- timezone_abbreviations_list — 夏時間、オフセットおよびタイムゾーン名を含む連想配列を返す
- timezone_identifiers_list — すべてのタイムゾーン識別子を含む配列を返す
- timezone_name_from_abbr — 略称からタイムゾーン名を返す
- timezone_name_get — タイムゾーンの名前を返す
- timezone_offset_get — GMT からのタイムゾーンのオフセットを返す
- timezone_open — 新しい DateTimeZone オブジェクトを返す
- timezone_transitions_get — タイムゾーンの変遷を返す

DB++ 関数

警告

この拡張モジュールは、 *実験的* なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

導入

db++ は、ドイツの企業 » Concept asa により作成された高性能でかつメモリ使用量とディスク使用量が 小さいことを特徴とするリレーショナルデータベースシステムです。 db++ では、SQL は補助的なインターフェイス原語として提供されており、 元来の SQL データベースであるわけではありませんが、SQL よりも 関係代数にはるかに強く影響された固有の AQL クエリ言語が提供されています。

Concept asa は、常にオープンソース言語のサポートに関心を持って 来ており、db++ は Perl、Tcl コールのインターフェイスを何年も前から有し、Tcl を内部的なストアドプロシージャ言語として使用しています。

要件

この拡張モジュールは外部クライアントライブラリに依存しており、 この拡張モジュールを使用したいシステム上に db++ クライアントが インストールされている必要があります。

» Concept asa が、 Linux および他のいくつかの Unix 版の » db++ デモ版 および » ドキュメント を 提供しています。また、Windows 版の db++ もありますが、 この拡張モジュールでは(まだ)サポートしていません。

インストール手順

この拡張モジュールをビルドするためには、db++ クライアントライブラリおよび ヘッダファイルがシステムにインストールされていることが必要です (db++ のインストールアーカイブには、これらがデフォルトで含まれています)。 この拡張モジュールをビルドするためには、**configure** の際にオプション --with-dbplus を設定する必要があります。

configure は、クライアントライブラリおよびヘッダファイルを デフォルトのパス */usr/dbplus*、 */usr/local/dbplus* および */opt/dblus* から探します。もし db++ を別の場所にインストールしている場合、 以下のようにしてインストール場所を **configure** オプションに 指定する必要があります。
--with-dbplus=/your/installation/path

実行時設定

設定ディレクティブは定義されていません。

リソース型

dbplus_relation

多くの db++ 関数は、*dbplus_relation* リソースを 操作または返します。 *dbplus_relation* は、保存された関係または クエリの結果として生成された関係へのハンドルです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

db++ エラーコード

DB++ エラーコード

PHP 定数	db++ 定数	意味
DBPLUS_ERR_NOERR (integer)	ERR_NOERR	Null エラー条件
DBPLUS_ERR_DUPLICATE (integer)	ERR_DUPLICATE	重複するタプルを挿入した
DBPLUS_ERR_EOSCAN (integer)	ERR_EOSCAN	rget()からスキャン終了
DBPLUS_ERR_EMPTY (integer)	ERR_EMPTY	関係が空(サーバ)
DBPLUS_ERR_CLOSE (integer)	ERR_CLOSE	サーバをクローズできない
DBPLUS_ERR_WLOCKED (integer)	ERR_WLOCKED	レコードは書き込みロックされている
DBPLUS_ERR_LOCKED (integer)	ERR_LOCKED	関係は既にロックされている
DBPLUS_ERR_NOLOCK (integer)	ERR_NOLOCK	関係をロックできない
DBPLUS_ERR_READ (integer)	ERR_READ	関係の読み込みエラー
DBPLUS_ERR_WRITE (integer)	ERR_WRITE	関係の書き込みエラー
DBPLUS_ERR_CREATE (integer)	ERR_CREATE	create() システムコールが失敗
DBPLUS_ERR_LSEEK (integer)	ERR_LSEEK	lseek() システムコールが失敗
DBPLUS_ERR_LENGTH (integer)	ERR_LENGTH	最大長を越えるタプル
DBPLUS_ERR_OPEN (integer)	ERR_OPEN	open() システムコールが失敗
DBPLUS_ERR_WOPEN (integer)	ERR_WOPEN	関係は既に書き込みオープンされている
DBPLUS_ERR_MAGIC (integer)	ERR_MAGIC	ファイルは関係でない
DBPLUS_ERR_VERSION (integer)	ERR_VERSION	ファイルは非常に古い関係である
DBPLUS_ERR_PGSIZE (integer)	ERR_PGSIZE	関係は異なったページサイズを使用している
DBPLUS_ERR_CRC (integer)	ERR_CRC	不正な CRC がスーパーページにある
DBPLUS_ERR_PIPE (integer)	ERR_PIPE	パイプ上の関係は lseek() を要求している
DBPLUS_ERR_NIDX (integer)	ERR_NIDX	セカンダリインデックスが多すぎる
DBPLUS_ERR_MALLOC (integer)	ERR_MALLOC	malloc() コールが失敗した
DBPLUS_ERR_NUSERS (integer)	ERR_NUSERS	最大ユーザ数エラー
DBPLUS_ERR_PREEXIT (integer)	ERR_PREEXIT	無効な使用法により発生
DBPLUS_ERR_ONTRAP (integer)	ERR_ONTRAP	シグナルにより発生
DBPLUS_ERR_PREPROC (integer)	ERR_PREPROC	プリプロセッサにおけるエラー
DBPLUS_ERR_DBPARSE (integer)	ERR_DBPARSE	パーサ上のエラー
DBPLUS_ERR_DBRUNERR (integer)	ERR_DBRUNERR	dbにおける実行エラー
DBPLUS_ERR_DBPREEXIT (integer)	ERR_DBPREEXIT	prexit() * プロシージャにより発生した終了条件
DBPLUS_ERR_WAIT (integer)	ERR_WAIT	少し待つ(simple のみ)
DBPLUS_ERR_CORRUPT_TUPLE (integer)	ERR_CORRUPT_TUPLE	クライアントが壊れたタプルを送信した
DBPLUS_ERR_WARNING0 (integer)	ERR_WARNING0	simple ルーチンが、修正済みの致命的でないエラーを発見した
DBPLUS_ERR_PANIC (integer)	ERR_PANIC	サーバは実際に実行中断していないが、全てのクライアントに ERR_PANIC が送信された
DBPLUS_ERR_FIFO (integer)	ERR_FIFO	fifo を作成できない
DBPLUS_ERR_PERM (integer)	ERR_PERM	不許可
DBPLUS_ERR_TCL (integer)	ERR_TCL	TCL_error
DBPLUS_ERR_RESTRICTED (integer)	ERR_RESTRICTED	ユーザ二人のみ
DBPLUS_ERR_USER (integer)	ERR_USER	アプリケーションプログラマによるライブラリの使用エラー
DBPLUS_ERR_UNKNOWN (integer)	ERR_UNKNOWN	

dbplus_add

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_add — リレーションにタプルを追加する

説明

int **dbplus_add** (resource `$relation` , array `$tuple`)

タプルをリレーション *relation* に追加します。

パラメータ

relation

tuple

属性/値の組の配列であり、 指定した *relation* に挿入されます。

実行に成功すると *tuple* 配列には新規に生成されたタプルの 完全なデータが含まれ、そこにはシーケンスのような暗黙の内に設定 される全てのドメインが含まれます。

返り値

この関数は、成功した場合に **DBPLUS_ERR_NOERR**、 失敗した場合に db++ エラーコードを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_errcode()

---

dbplus_aql

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_aql — AQL クエリを実行する

説明

resource **dbplus_aql** (string `$query` [, string `$server` [, string `$dbpath`]])

AQL *query* を、指定した *server* および *dbpath* で実行します。

パラメータ

query

AQL A... Query Language に関するより詳細な情報については、 オリジナルの db++ マニュアルで提供されています。

server

dbpath

返り値

成功時に関係ハンドルを返します。結果データは、 dbplus_next() および **dbplus_current()** をコールすることにより、このリレーションから取得可能です。 他のリレーションアクセス関数は結果のリレーションで動作しません。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_chdir

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_chdir — データベース仮想カレントディレクトリを設定/取得する

説明

string **dbplus_chdir** ([string `$newdir`])

dbplus_open() でリレーションファイルを探す際の 仮想カレントディレクトリを変更します。

パラメータ

`newdir`

リレーションファイルの新しいディレクトリ。このパラメータを省略すると、 現在の作業ディレクトリで探します。

返り値

カレントディレクトリの絶対パスを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_close

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_close — リレーションを閉じる

説明

mixed **dbplus_close** (resource `$relation`)

dbplus_open() でオープンしたリレーションを閉じます。

パラメータ

`relation`

dbplus_open() でオープンしたリレーション。

返り値

成功した場合に **`TRUE`**、失敗した場合に **`DBPLUS_ERR_UNKNOWN`** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_curr

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_curr — リレーションからカレントタプルを取得する

説明

int **dbplus_curr** (resource `$relation` , array `&$tuple`)

指定した *`relation`* について、カレントタプルに関するデータを読み込みます。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

tuple

データがこのパラメータに連想配列として返されます。

返り値

この関数は、成功時にゼロ (すなわちDBPLUS_ERR_NOERR)、失敗時に db++ エラーコードを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_first()
- dbplus_prev()
- dbplus_next()
- dbplus_last()
- dbplus_errcode()

---

dbplus_errcode

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_errcode — 指定したエラーコードまたは直近のエラーに関するエラー文字列を取得する

説明

string **dbplus_errcode** ([int $errno])

指定したエラーコードに対応するエラー文字列を返します。

パラメータ

errno

エラーコード。指定しなかった場合は、直近の db++ 操作の結果コードを使用します。

返り値

エラーメッセージを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_errno

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_errno — 直近の操作に関するエラーコードを取得する

説明

int **dbplus_errno** (void)

直近の db++ 操作からのエラーコードを返します。

返り値

エラーコードを表す整数値を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_errcode()

---

dbplus_find

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_find — リレーションに拘束を設定する

説明

int **dbplus_find** (resource `$relation` , array `$constraints` , mixed `$tuple`)

指定したリレーション *relation* に拘束を設定します。

この後、dbplus_curr() または dbplus_next() のような関数をコールすると、 指定した拘束にマッチするタプルのみが取得されます。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

constraints

拘束は、ドメイン名、比較演算子、比較値を含む複合文字列です。 パラメータ配列 *constraints* は文字列の配列で、 この各要素は、ドメイン、演算子、値を含むか、 複数の三要素からなる文字列の配列です。

比較演算子は、以下の文字列のどれかとなります。 正規表現による検索には、'==', '>', '>=', '<', '<=', '!=', '~'、ビット演算には 'BAND' または 'BOR' を使用します。

tuple

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_unselect()

---

dbplus_first

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_first — リレーションから最初のタプルを取得する

説明

int **dbplus_first** (resource `$relation` , array `&$tuple`)

指定したリレーション *relation* の最初のタプルのデータを読み込み、 連想配列として *tuple* に返します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

tuple

返り値

成功した場合に **DBPLUS_ERR_NOERR**、 失敗した場合に db++ エラーコードを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_curr()
- dbplus_prev()
- dbplus_next()
- dbplus_last()
- dbplus_errcode()

---

dbplus_flush

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_flush — リレーションに行った全ての変更をフラッシュする

説明

int **dbplus_flush** (resource `$relation`)

直近のフラッシュ以降にリレーション *relation* に適用されたすべての変更をディスクに書き込みます。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

成功した場合に **DBPLUS_ERR_NOERR**、 失敗した場合に db++ エラーコードを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_errcode()

---

dbplus_freealllocks

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_freealllocks — このクライアントにより保持された全てのロックを解放する

説明

int **dbplus_freealllocks** (void)

このクライアントによりロックされたタプルを全て解放します。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_getlock()
- dbplus_freelock()
- dbplus_freerlocks()

---

dbplus_freelock

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_freelock — タプルの書き込みロックを解放する

説明

int **dbplus_freelock** (resource `$relation` , string `$tuple`)

dbplus_getlock() より以前に得た指定 *`tuple`* の書き込みロックを解放します。

パラメータ

`relation`

dbplus_open() でオープンしたリレーション。

`tuple`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_getlock()
- dbplus_freerlocks()
- dbplus_freealllocks()

---

dbplus_freerlocks

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_freerlocks — 指定したリレーションに関する全てのタプルロックを解放する

説明

int **dbplus_freerlocks** (resource `$relation`)

指定した *relation* で保持されている全てのタプルロックを解放します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_getlock()
- dbplus_freelock()
- dbplus_freealllocks()

---

dbplus_getlock

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_getlock — タプルの書き込みロックを取得する

説明

int **dbplus_getlock** (resource `$relation` , string `$tuple`)

指定した *tuple* へ書き込みロックを要求します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

tuple

返り値

成功時にゼロ、失敗時にゼロでないコード、特に **DBPLUS_ERR_WLOCKED** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_freelock()
- dbplus_freerlocks()
- dbplus_freealllocks()

---

dbplus_getunique

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_getunique — あるリレーションのユニークな ID 番号を取得する

説明

int **dbplus_getunique** (resource `$relation` , int `$uniqueid`)

指定した *relation* に関してユニークであることが保証 された数を取得し、*uniqueid* で指定した変数にその数を代入します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

uniqueid

返り値

成功した場合に **DBPLUS_ERR_NOERR**、 失敗した場合に db++ エラーコードを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_info

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_info — リレーションについての情報を取得する

説明

int **dbplus_info** (resource `$relation` , string `$key` , array `&$result`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

relation

dbplus_open() が返すリレーション。

key

result

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_last

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_last — リレーションから直近のタプルを取得する

説明

int **dbplus_last** (resource `$relation` , array `&$tuple`)

指定した *relation* から直近のタプルに関するデータを読み込み、 *tuple* に連想配列として代入します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

tuple

返り値

成功した場合に **DBPLUS_ERR_NOERR**、 失敗した場合に db++ エラーコードを返します。 code on failure.

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_first()
- dbplus_curr()
- dbplus_prev()
- dbplus_next()

---

dbplus_lockrel

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_lockrel — リレーションに書き込みロックを要求する

説明

int **dbplus_lockrel** (resource `$relation`)

指定した *relation* に書き込みロックを要求します。

他のクライアントからリレーションに対してクエリを実行することはできますが、 ロックされている間は更新できません。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_next

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_next — リレーションから次のタプルを取得する

説明

int **dbplus_next** (resource `$relation` , array `&$tuple`)

指定した *relation* の次のタプルからデータを読み込み、 *tuple* に連想配列として返します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

tuple

返り値

成功した場合に **DBPLUS_ERR_NOERR**、 失敗した場合に db++ エラーコードを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_first()
- dbplus_curr()
- dbplus_prev()
- dbplus_last()

---

dbplus_open

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_open — リレーションファイルをオープンする

説明

resource **dbplus_open** (string `$name`)

指定したリレーションファイルをオープンします。

パラメータ

name

ファイル名またはリレーション、絶対パス名とします。 これは、指定したパス名及びサーバ上の絶対リレーションファイル パス名を常にマップします。

返り値

成功時にリレーションファイルリソース(カーソル)が返されます。これは、 このリレーションを参照する以降の全てのコマンドで使用されます。 失敗時にはゼロが返されます。実際のエラーコードは、 dbplus_errno() により取得可能です。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_prev

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_prev — リレーションから前のタプルを取得する

説明

int **dbplus_prev** (resource `$relation` , array `&$tuple`)

指定した *relation* から前のタプルのデータを読み込み、 *tuple* に連想配列として返します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

tuple

返り値

成功した場合に **DBPLUS_ERR_NOERR**、 失敗した場合に db++ エラーコードを返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_first()
- dbplus_curr()
- dbplus_next()
- dbplus_last()

---

dbplus_rchperm

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rchperm — リレーションの許可属性を変更する

説明

int **dbplus_rchperm** (resource `$relation` , int `$mask` , string `$user` , string `$group`)

mask , *user* ,*group* で 指定した許可属性に変更します。これらの値はオペレーティングシステムに依存します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

mask

user

group

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rcreate

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rcreate — 新しい DB++ リレーションを作成する

説明

resource **dbplus_rcreate** (string `$name` , mixed `$domlist` [, bool `$overwrite`])

新しいリレーションを作成します。同名の既存のリレーションは、 そのリレーションが現在使用されておらず、 *overwrite* に TRUE が設定されている場合にのみ上書きされます。

パラメータ

name

domlist

ドメインの組み合わせ。単一のドメイン名を表す文字列か、 ドメイン名の配列となります。

このパラメータにはドメインを示す文字列の配列で 新規リレーションのドメインを指定する必要があります。 ドメイン指定文字列は、このリレーションへのユニークなドメイン名、スラッシュ、 型指定文字からなります。利用可能な型指定子及びその意味については、 db++ のドキュメント、特に dbcreate(1) のマニュアルを参照ください。

注意: この関数には、複数のドメインを空白で区切った文字列を指定することもできます。 しかし、配列を使用することを推奨します。

overwrite

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rcrtexact

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rcrtexact — インデックスを含む、リレーションの空のコピーを作成する

説明

mixed **dbplus_rcrtexact** (string `$name` , resource `$relation` [, bool `$overwrite`])

dbplus_rcrtexact() は、 指定した *relation* の空のコピーを *name* という名前で作成します。

パラメータ

name

relation

コピーするリレーション。dbplus_open() でオープンしたもの。

overwrite

このパラメータが **TRUE** であり 他のプロセスがそのリレーションを使用していない場合にのみ、 同じ名前の既存のリレーションが上書きされます。

返り値

成功した場合にリソース、失敗した場合に **DBPLUS_ERR_UNKNOWN** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rcrtlike

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rcrtlike — デフォルトのインデックスで、リレーションの空のコピーを作成する

説明

mixed **dbplus_rcrtlike** (string `$name` , resource `$relation` [, int `$overwrite`])

dbplus_rcrtlike() は、 指定した *relation* の空のコピーを *name* という名前で作成しますが、デフォルトの インデックスを使用します。

パラメータ

name

relation

コピーするリレーション。dbplus_open() でオープンしたもの。

overwrite

このパラメータが **TRUE** であり 他のプロセスがそのリレーションを使用していない場合にのみ、 同じ名前の既存のリレーションが上書きされます。

返り値

成功した場合にリソース、失敗した場合に **DBPLUS_ERR_UNKNOWN** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_resolve

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_resolve — リレーションのホスト情報を取得する

説明

array **dbplus_resolve** (string `$relation_name`)

dbplus_resolve() は指定された *relation_name* を解決し、内部サーバ ID・ 実際のホスト名およびホスト上のデータベースのパスを取得します。

パラメータ

relation_name

リレーション名。

返り値

キー *sid*、*host* および *host_path* を含む配列、あるいはエラー時に **FALSE** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_tcl()

---

dbplus_restorepos

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_restorepos — 位置を復元する

説明

int **dbplus_restorepos** (resource `$relation` , array `$tuple`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

tuple

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rkeys

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rkeys — リレーションの主キーを新規に指定する

説明

mixed **dbplus_rkeys** (resource `$relation` , mixed `$domlist`)

dbplus_rkeys() は、 *relation* に指定された現在の主キーを、 *domlist* で指定されたドメインに変更します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

domlist

ドメインの組み合わせ。ドメイン名を含む文字列あるいは 複数のドメイン名を含む配列を渡します。

返り値

成功した場合にリソース、失敗した場合に **DBPLUS_ERR_UNKNOWN** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_ropen

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_ropen — リレーションファイルをローカルにオープンする

説明

resource **dbplus_ropen** (string `$name`)

dbplus_ropen() は、クライアント/サーバ間の オーバーヘッドを防いですばやくアクセスするために、リレーション *`file`* をローカルにオープンします。 アクセスは読み込み専用となり、返されるリレーションを利用できるのは **dbplus_current()** および dbplus_next() のみとなります。

パラメータ

`name`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rquery

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rquery — ローカル (raw) AQL クエリを実行する

説明

resource **dbplus_rquery** (string `$query` [, string `$dbpath`])

dbplus_rquery() は、db++ クライアントライブラリに組み込まれた AQL インタプリタを使用してローカル(raw) AQL クエリを実行します。 **dbplus_rquery()** は dbplus_aql() より高速ですが、 ローカルのデータについてのみしか実行できません。

パラメータ

`query`

`dbpath`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rrename

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rrename — リレーションの名前を変更する

説明

int **dbplus_rrename** (resource `$relation` , string `$name`)

dbplus_rrename() は、 *relation* の名前を *name* に変更します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

name

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rsecindex

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rsecindex — リレーションに新規セカンダリインデックスを作成する

説明

mixed **dbplus_rsecindex** (resource `$relation` , mixed `$domlist` , int `$type`)

dbplus_rsecindex() は、 *relation* に新しいセカンダリインデックスを作成します。 このインデックスには *domlist* で指定したドメインが含まれ、型は *type* です。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

domlist

ドメイン名の組み合わせ。単一のドメイン名を含む文字列あるいは 複数のドメイン名を含む配列を渡します。

type

返り値

成功した場合にリソース、失敗した場合に **DBPLUS_ERR_UNKNOWN** を返します。

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_runlink

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_runlink — ファイルシステムからリレーションを削除する

説明

int **dbplus_runlink** (resource `$relation`)

dbplus_unlink() は、 *relation* を閉じて削除します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_rzap

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_rzap — リレーションから全てのタプルを削除する

説明

int **dbplus_rzap** (resource `$relation`)

dbplus_rzap() は、 *relation* からタプルをすべて削除します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_savepos

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_savepos — 位置を保存する

説明

int **dbplus_savepos** (resource `$relation`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_setindex

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_setindex — インデックスを設定する

説明

int **dbplus_setindex** (resource `$relation` , string `$idx_name`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

`relation`

dbplus_open() でオープンしたリレーション。

`idx_name`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_setindexbynumber

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_setindexbynumber — インデックスを数字で設定する

説明

int **dbplus_setindexbynumber** (resource `$relation` , int `$idx_number`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

`relation`

dbplus_open() でオープンしたリレーション。

`idx_number`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_sql

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_sql — SQL クエリを実行する

説明

resource **dbplus_sql** (string `$query` [, string `$server` [, string `$dbpath`]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

`query`

`server`

`dbpath`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_tcl

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_tcl — サーバ側で TCL コードを実行する

説明

string **dbplus_tcl** (int `$sid` , string `$script`)

db++ サーバは、各クライアント接続について TCL インタプリタを準備します。 このインタプリタは、ある種のストアドプロシージャとしてクライアントに 指定された TCL コードをサーバで実行するもので、 クライアント/サーバ間のデータ伝送及びコンテキスト切替を回避し、 データベース処理性能を改善するために使用されます。

dbplus_tcl() は、TCL *`script`* コードが実行される、クライアント接続 ID を必要とします。dbplus_resolve() が、この接続 ID を返します。この関数は、TCL コードの返り値または TCL コードが失敗 した場合には、TCL エラーメッセージを返します。

パラメータ

`sid`

`script`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_resolve()

---

dbplus_tremove

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_tremove — タプルを削除し、新規カレントタプルを返す

説明

int **dbplus_tremove** (resource `$relation` , array `$tuple` [, array `&$current`])

dbplus_tremove() は、関係の中のタプルに完全に 一致する場合、 *`tuple`* を削除します。 *`current`* が指定された場合、 **dbplus_tremove()** をコールした後、新規のカレントの タプルのデータが代入されます。

パラメータ

`relation`

dbplus_open() でオープンしたリレーション。

`tuple`

`current`

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_undo

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_undo — 元に戻す

説明

int **dbplus_undo** (resource `$relation`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

`relation`

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_undoprepare

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_undoprepare — 元に戻す準備をする

説明

int **dbplus_undoprepare** (resource `$relation`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_unlockrel

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_unlockrel — リレーションの書き込みロックを中断する

説明

int **dbplus_unlockrel** (resource `$relation`)

以前 dbplus_lockrel() により得られた書き込みロックを開放します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_unselect

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_unselect — リレーションから制約を削除する

説明

int **dbplus_unselect** (resource `$relation`)

dbplus_unselect() をコールすると、 dbplus_find() により *relation* に設定された制約が削除されます。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_update

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_update — リレーション内の指定したタプルを更新する

説明

int **dbplus_update** (resource `$relation` , array `$old` , array `$new`)

dbplus_update() は、*old* で指定したタプル *new* からのデータで置換します。 置換が行われるのは、*old* が *relation* の中のタプルに完全に一致する場合のみです。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

old

古いタプルの内容。

new

新しいタプルの内容。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

dbplus_xlockrel

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_xlockrel — リレーションの排他的ロックを要求する

説明

int **dbplus_xlockrel** (resource `$relation`)

relation に他のクライアントからの読み込み アクセスさえ拒否する排他的ロックを要求します。

パラメータ

relation

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- dbplus_xunlockrel()

---

dbplus_xunlockrel

(PHP 4 >= 4.0.7, PECL dbplus:0.9)

dbplus_xunlockrel — リレーションの排他的ロックを解放する

説明

int **dbplus_xunlockrel** (resource `$relation`)

以前に dbplus_xlockrel() で取得した排他的ロックを開放します。

パラメータ

`relation`

dbplus_open() でオープンしたリレーション。

返り値

注意

警告

この関数は、 *実験的* なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

目次

- dbplus_rsecindex — リレーションに新規セカンダリインデックスを作成する
- dbplus_runlink — ファイルシステムからリレーションを削除する
- dbplus_rzap — リレーションから全てのタプルを削除する
- dbplus_savepos — 位置を保存する
- dbplus_setindex — インデックスを設定する
- dbplus_setindexbynumber — インデックスを数字で設定する
- dbplus_sql — SQL クエリを実行する
- dbplus_tcl — サーバ側で TCL コードを実行する
- dbplus_tremove — タプルを削除し、新規カレントタプルを返す
- dbplus_undo — 元に戻す
- dbplus_undoprepare — 元に戻す準備をする
- dbplus_unlockrel — リレーションの書き込みロックを中断する
- dbplus_unselect — リレーションから制約を削除する
- dbplus_update — リレーション内の指定したタプルを更新する
- dbplus_xlockrel — リレーションの排他的ロックを要求する
- dbplus_xunlockrel — リレーションの排他的ロックを解放する

（dbm 型の）データベース抽象化レイヤ関数

導入

これらの関数は、Berkeley DB 型データベースへのアクセス用の基本関数を 構成します。

これらの関数は、複数のファイルベースのデータベース用の抽象化レイヤです。 その機能は、» Sleepycat Software's DB2 でサポートされているような近代的なデータベースの 機能のサブセットに制限されています（IBM の DB2 と混同しないでください。 このデータベースは、ODBC 関数 を通じてサポートされます）。

要件

各機能の動作は、使用するデータベースの実装に依存します。 dba_optimize() および dba_sync() のような関数は、特定のデータベースでは動作しますが、他のデータベースでは 機能しません。サポートされるdbaハンドラをダウンロードし、インストールする 必要があります。

DBAハンドラのリスト

ハンドラ	注意
dbm	dbm は最も古い(元祖の)Berkeley DB 型データベースです。可能なら ば使用しない方が良いでしょう。DB2 および gdbm に組み込まれている 互換性を保つための関数は、サポートされません。これは、実装されて いるのがソースレベルの互換性のみであり、元の dbm フォーマットを 処理することができないためです。
ndbm	ndbm は、dbm に比べて新しく、dbm よりも柔軟です。 dbm 固有の制約の多くをまだ有しています（そのため、 推奨されません）。
gdbm	Gdbm は、» GNU データベースマネージャ です。
db2	DB2 は、» Sleepycat Software's DB2 です。これは、「スタンドアローンおよびクライアント/サーバー アプリケーションの両方で高性能な組込みデータベースサポートを提供する プログラム用ツールキット」として記述されています。
db3	DB3 は、» Sleepycat Software's DB3 です。
db4	DB4 は、» Sleepycat Software's DB4 です。PHP 4.3.2 以降で利用可能です。
cdb	cdb は「固定データベースの作成・読込み用の、高速で高信頼性の 軽量型パッケージ」です。これは qmail の作者によるものであり、» http://cr.yp.to/cdb.html にあります。 固定であるため、読込操作のみがサポートされます。 PHP 4.3.0 以降、内部的な cdb ライブラリにより（更新ではなく） 書き込みがサポートされています。
cdb_make	PHP 4.3.0 以降、付属する cdb ライブラリを使用する場合に cdb ファイルの（更新ではなく）作成をサポートします。
flatfile	これは、PHP 4.3.0 以降で利用可能で、古い dbm 拡張モジュールとの互換性の ためだけのものであり、使用するべきではありません。 しかし、ファイルがこの形式で作成された場所でこれを使用する ことができます。これは、configure が外部ライブラリを見付ける ことができない場合に生じます。
inifile	これは PHP 4.3.3 以降で利用可能で、PHP スクリプトから php.ini ファイルを書き換えられるようにします。ini ファイルを扱う場合は、 array(0=>group,1=>value_name) のような配列、あるいは "[group]value_name" のような文字列を渡します（group はオプションです）。 dba_firstkey() や dba_nextkey() はキーを文字列形式で返しますが、PHP 5 以降で利用できる dba_key_split() を用いるとそれを配列形式に 変換できます。その際に **FALSE** を失うこともありません。
qdbm	これは PHP 5.0.0 以降で有効です。qdbm ライブラリは » http://qdbm.sourceforge.net にあります。

dba_open() または dba_popen() 関数を実行する際、引数にハンドラ名の一つを指定する必要があります。 実際に利用可能なハンドラのリストは、phpinfo() または dba_handlers() をコールした際に表示されます。

インストール手順

設定オプション --enable-dba=shared を使用することにより、dbm 形式のデータベースをサポートする動的に ロード可能なモジュールを有効にして PHP を構築することができます。 また、PHP の configure 行に設定スイッチ *--with-XXXX* を指定することにより、少なくとも以下のハンドラの一つのサポートを 追加する必要があります。

警告

PHP の configuring とコンパイルを済ませたら、コマンドラインから次の テストを実行する必要があります: *php run-tests.php ext/dba* これは、指定したハンドラの組み合わせが動作するかどうかを調べます。 いちばん問題のあるのは *dbm* と *ndbm* の組み合わせで、これらは多くの場合何らかの衝突を引き起こします。 その原因は、いくつかのシステムではこれらのライブラリが他のライブラリの 一部となっていることで。configure 時のテストでは個々のハンドラについての 設定不備は調べられますが、それらの組み合わせについてはテストできません。

サポートされるDBAハンドラ

ハンドラ	configure のスイッチ
dbm	dbm のサポートを有効にするには、 --with-dbm[=DIR] を追加します。 **注意**: dbm は一般的にはラッパで、しばしば失敗を引き起こします。 その動作をしっかり把握しており、ほんとうにそれを必要とする場合のみ dbm を使用するようにすべきです。
ndbm	ndbm のサポートを有効にするには、 --with-ndbm[=DIR] を追加します。 **注意**: ndbm は一般的にはラッパで、しばしば失敗を引き起こします。 その動作をしっかり把握しており、ほんとうにそれを必要とする場合のみ ndbm を使用するようにすべきです。
gdbm	gdbm のサポートを有効にするには、 --with-gdbm[=DIR] を追加します。
db2	db2 のサポートを有効にするには、 --with-db2[=DIR] を追加します。 **注意**: db2 は db3 および db4 とは同時に使えません。
db3	db3 のサポートを有効にするには、 --with-db3[=DIR] を追加します。 **注意**: db3 は db2 および db4 とは同時に使えません。
db4	db4 のサポートを有効にするには、 --with-db4[=DIR] を追加します。 **注意**: db4 は db2 および db3 とは同時に使えません。 **注意**: これは PHP 4.3.2 で追加されました。これ以前のバージョンでは、 --with-db3=DIR を使用し、 DIR に db4 ライブラリのパスを指定する必要する必要があります。 バージョン 4.3.0 より前の PHP では、4.1 以降のバージョンの db を使用することはできません。また、バージョン 4.1 から 4.1.24 までの db ライブラリは、どの PHP のバージョンでも利用できません。
cdb	cdb のサポートを有効にするには、 --with-cdb[=DIR] を追加します。 **注意**: PHP 4.3.0 以降、付属する cdb ライブラリを使用するために DIR を省略することができます。この場合、cdb_make ハンドラが追加されます。 これにより cdb ファイルを作成したり、PHPのストリームを用いて ネットワーク上の cdb ファイルにアクセスできるようになります。
flatfile	flatfile のサポートを有効にするには、 --with-flatfile を追加します。 **注意**: これは、古い dbm 拡張モジュール との互換性のために PHP 4.3.0 で追加されました。 このハンドラは、他のハンドラで必要とされるライブラリをひとつも インストールすることができない場合、そして、付属する cdb ハンドラを使用することができない場合にのみ使用してください。
inifile	inifile のサポートを有効にするには、 --with-inifile を追加します。 **注意**: これは PHP 5.0.0 で追加されました。これにより Microsoft 形式の *.ini* ファイル（*php.ini* のような）を 読み書きできるようになります。
qdbm	qdbm のサポートを有効にするには、 --with-qdbm[=DIR] を追加します。 **注意**: qdbm は dbm および gdbm とは同時に使えません。 **注意**: これは PHP 5.0.0 で追加されました。qdbm ライブラリは » http://qdbm.sourceforge.net にあります。

注意: PHP 4.3.0 までは、db2 および db3 ハンドラの両方を追加することが できましたが、内部的に使用できるのは片方だけでした。これは、両方のファイル形式を使用することができないことを意味します。 PHP 5.0.0 以降、このような設定ミスを回避するよう設定の確認が行われます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

関数 dba_open() および dba_popen() は、指定したデータベースファイルに アクセスするためのハンドルを返します。このハンドルは、他の全ての dba 関数コールで使用されます。

定義済み定数

定数は定義されていません。

例

Example#1 DBA の例

```
<?php

$id = dba_open("/tmp/test.db", "n", "db2");

if (!$id) {
    echo "dba_open  failed¥n";
    exit;
}

dba_replace("key", "This is an example!", $id);

if (dba_exists("key", $id)) {
    echo dba_fetch("key", $id);
    dba_delete("key", $id);
}

dba_close($id);
?>
```

DBA はバイナリセーフであり、いかなる制限も受けません。しかし、 使用するデータベースの実装による全ての制約を継承します。

全てのファイルベースのデータベースは、完全に使用可能なものについて 新規に作成されたデータベースのファイルモードを設定する手段を、 提供する必要があります。 ファイルモードは、通常 dba_open() または dba_popen() に 4 番目の引数として渡されます。

dba_firstkey() および dba_nextkey() 関数を用いて全てのエントリに 連続的にアクセスすることができます。アクセスする際にデータベースを 変更できない可能性があります。

Example#2 データベースへのアクセス

```
<?php

// データベースをオープンする

$key = dba_firstkey($id);

while ($key != false) {
    if (true) {         //  他の操作を後で行うためにキーを記憶する
        $handle_later[] = $key;
    }
    $key = dba_nextkey($id);
}

foreach ($handle_later as $val) {
    dba_delete($val, $id);
}

?>
```

---

dba_close

(PHP 4, PHP 5)

dba_close — DBA データベースを閉じる

説明

void **dba_close** (resource $handle)

dba_close() は確立されたデータベースを閉じ、 により指定したデータベースハンドルの全てのリソースを開放します。

パラメータ

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

値を返しません。

参考

- dba_open()
- dba_popen()

---

dba_delete

(PHP 4, PHP 5)

dba_delete — キーが指す DBA エントリを削除する

説明

bool **dba_delete** (string $key , resource $handle)

dba_delete() は、指定されたエントリを データベースから削除します。

パラメータ

key

削除するエントリのキー。

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dba_exists()
- dba_fetch()
- dba_insert()
- dba_replace()

---

dba_exists

(PHP 4, PHP 5)

dba_exists — キーが存在するかどうかを確認する

説明

bool **dba_exists** (string $key , resource $handle)

dba_exists() は、指定した *key* がデータベースに存在するかどうかを確認します。

パラメータ

key

確認を行うキー。

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

キーが存在する場合に **TRUE** 、それ以外の場合に **FALSE** を返します。

参考

- dba_delete()
- dba_fetch()
- dba_insert()
- dba_replace()

---

dba_fetch

(PHP 4, PHP 5)

dba_fetch — キーが指すデータを取得する

説明

string **dba_fetch** (string $key , resource $handle)
string **dba_fetch** (string $key , int $skip , resource $handle)

dba_fetch() は、*handle* が指すデータベースから *key* が指すデータを 取得します。

パラメータ

key

データを取得するキー。

注意: inifile を利用する際には、この関数は配列を受け入れることが可能です。 インデックス 0 にはグループ名、インデックス 1 には値の名前を格納します。 dba_key_split() も参照ください。

skip

dba データベースを使用する際に無視する キー / 値 の組み合わせ数。 複数のキーに同じ名前をつけることがサポートされていないデータベースです、 この値は無視されます。

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

キー / データの組が見つかった場合にそれに関連する文字列、 それ以外の場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3	複数のキーに同じ名前をつけられる cdb の機能に対応するため、 *skip* パラメータが使用可能になりました。

参考

- dba_exists()
- dba_delete()
- dba_insert()
- dba_replace()
- dba_key_split()

---

dba_firstkey

(PHP 4, PHP 5)

dba_firstkey — 最初のキーを取得する

説明

string **dba_firstkey** (resource $handle)

dba_firstkey() はデータベースの最初のキーを返し、 内部キーポインタをリセットします。 この関数によりデータベース全体を連続的にサーチすることが可能になります。

パラメータ

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

成功した場合にキー、失敗した場合に **FALSE** を返します。

参考

- dba_nextkey()
- dba_key_split()
- DBA の例の例 2

---

dba_handlers

(PHP 4 >= 4.3.0, PHP 5)

dba_handlers — 利用可能なハンドラの一覧を得る

説明

array **dba_handlers** ([bool $full_info])

dba_handlers() は、この拡張モジュールにより サポートされる全てのハンドラの一覧を返します。

パラメータ

full_info

結果の全ての情報を表示するかどうかを切り替えます。 デフォルトは **FALSE** です。

返り値

データベースハンドラの配列を返します。*full_info* が **TRUE** の場合、この配列はハンドラ名をキー、そのバージョンを値とする 連想配列となります。それ以外の場合はハンドラ名を値に持つ数値添字の配列と なります。

注意: cdb ライブラリが用いられている場合、 *cdb* および *cdb_make* が表示されます。

例

Example#1 dba_handlers() の例

```
<?php

echo "Available DBA handlers:¥n";
foreach (dba_handlers(true) as $handler_name => $handler_version) {
  // バージョン番号を見やすくする
  $handler_version = str_replace('$', '', $handler_version);
  echo "  -  $handler_name:  $handler_version¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
Available DBA handlers:
  - cdb: 0.75, Revision: 1.3.2.3
  - cdb_make: 0.75, Revision: 1.2.2.4
  - db2: Sleepycat Software: Berkeley DB 2.7.7: (08/20/99)
  - inifile: 1.0, Revision: 1.6.2.3
```

```
- flatfile: 1.0, Revision: 1.5.2.4
```

---

dba_insert

(PHP 4, PHP 5)

dba_insert — エントリを挿入する

説明

bool **dba_insert** (string $key , string $value , resource $handle)

dba_insert() は、 データベースに *key* および *value* で記述されるエントリを挿入します。

パラメータ

key

挿入するエントリのキー。もしこのキーが既にデータベースに存在する場合、 この関数は失敗します。既存のキーを置き換える場合は、dba_replace() を使用してください。

value

挿入する値。

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dba_exists()
- dba_delete()
- dba_fetch()
- dba_replace()

---

dba_key_split

(PHP 5)

dba_key_split — 文字列形式のキーを配列形式に分割する

説明

mixed **dba_key_split** (mixed $key)

dba_key_split() は、キー（文字列形式） を配列に分割します。

パラメータ

key

文字列形式のキー。

返り値

array(0 => group, 1 => value_name) 形式の配列を返します。 *key* が **NULL** あるいは **FALSE** の場合には この関数は **FALSE** を返します。

参考

- dba_firstkey()

- dba_nextkey()
- dba_fetch()

---

dba_list

(PHP 4 >= 4.3.0, PHP 5)

dba_list — オープンされている全データベースファイルのリストを得る

説明

array **dba_list** (void)

dba_list() は、オープンされている全ての データベースファイルの一覧を返します。

返り値

resourceid => filename 形式の連想配列を返します。

---

dba_nextkey

(PHP 4, PHP 5)

dba_nextkey — 次のキーを取得する

説明

string **dba_nextkey** (resource `$handle`)

dba_nextkey() は、 が指すデータベースの次のキーを返し、内部キーポインタを進めます。

パラメータ

`handle`

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

成功した場合にキー、失敗した場合に **`FALSE`** を返します。

参考

- dba_firstkey()
- dba_key_split()
- DBA の例の例 2

---

dba_open

(PHP 4, PHP 5)

dba_open — データベースをオープンする

説明

resource **dba_open** (string `$path` , string `$mode` [, string `$handler` [, mixed `$...`]])

dba_open() は、*`handler`* を用いて *`mode`* を指定し、*`path`* にデータベースインスタンスを生成します。

パラメータ

`path`

通常のファイルシステムのパス。

mode

既存のデータベースへの読み込みアクセスには "r"、読込み/書き込み アクセスには "w" を指定します。 読み込み/書き込みアクセスおよび存在しない場合にデータベースの作成を するには "c" を、そして作成、削除、読込み/書き込みアクセス用には "n" を指定します。

さらに、次の文字でデータベースのロック方法を指定することができます。 *.lck* でデータベースをロックする場合には "l"、 データベースファイル自体をロックする場合は "d" を使用してください。 アプリケーション全体で統一した方法を用いることが重要です。

アクセスのテストを行う際にロックのために待ちたくない場合、"t" を 3 番目の文字に追加することができます。明らかにデータベースのロックが 不要な場合には、"l" や "d" の代わりに "-" を使用してロックを 行わないことができます。"d"、"l" または "-" のどれも指定されない場合、 "d"が指定されたものとしてデータベースファイルをロックします。

注意: ひとつのデータベースファイルに同時に書き込めるのは、ひとつだけです。 dba を Web サーバで使用している際に複数のリクエストが書き込み処理を 行う必要がある場合、他の書き込みが終わってからでないと次の書き込みを 行うことができません。また、書き込み中に読み込むことはできません。 dba 拡張モジュールは、このようなことを防止するためにロックを使用します。 以下の表を参照してください:

DBA のロック処理

既にオープンしているデータベース	*mode* = "rl"	*mode* = "rlt"	*mode* = "wl"	*mode* = "wlt"	*mode* = "rd"	*mode* = "rdt"	*mode* = "wd"	*mode* = "wdt"
not open	ok	ok	ok	ok	ok	ok	ok	ok
mode = "rl"	ok	ok	wait	false	illegal	illegal	illegal	illegal
mode = "wl"	wait	false	wait	false	illegal	illegal	illegal	illegal
mode = "rd"	illegal	illegal	illegal	illegal	ok	ok	wait	false
mode = "wd"	illegal	illegal	illegal	illegal	wait	false	wait	false

- ok: 2 番目のコールは成功します。
- wait: 2 番目のコールは、最初のデータベースで dba_close() がコールされるまで待ちます。
- false: 2 番目のコールは、false を返します。
- illegal: *mode* パラメータで "l" および "d" を同時に使用することはできません。

handler

path にアクセスする際に使用する ハンドラの名前。 **dba_open()** に指定したすべてのオプションが 渡され、その機能を用いることができます。

返り値

成功した場合に正のハンドル、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	ネットワーク接続されたデータベースファイルをオープンすることができます。 しかし、（http や ftp のような）ソケット接続が使用された場合、 リソース自体ではなくこの接続がロックされます。このような場合、 このリソースに関してのロック処理は単に無視されることになり、 他の解決策を見付ける必要があることに留意する必要があります。
4.3.0	ロック処理と *mode* 修正子 "l", "d", "-", "t" が追加されました。 以前のバージョンの PHP では、GDBM 以外のデータベースハンドラで 同時データベースアクセスに対する保護を行うためには、セマフォを 使用する必要がありました。 System V セマフォサポート を参照ください。
4.3.5 以前	オープンモード 'c' はいくつかの内部ハンドラでは正常に動作せず、 既存のデータベースにデータを追加するのではなく、データベースを 切り捨ててしまっていました。また、dbm と ndbm は典型的な設定で モード 'c' の処理が正常に動作しません（これは修正できません）。

参考

- dba_popen()
- dba_close()

---

dba_optimize

(PHP 4, PHP 5)

dba_optimize — データベースを最適化する

説明

bool **dba_optimize** (resource $handle)

dba_optimize() は、指定されたデータベースを最適化します。

パラメータ

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dba_sync()

---

dba_popen

(PHP 4, PHP 5)

dba_popen — データベースを持続的にオープンする

説明

resource **dba_popen** (string $path , string $mode [, string $handler [, mixed $...]])

dba_popen() は、*handler* を用いて *mode* で *path* に持続的なデータベースインスタンスを確立します。

パラメータ

path

通常のファイルシステムのパス。

mode

既存のデータベースへの読み込みアクセスには "r"、読込み/書き込み アクセスには "w" を指定します。 読み込み/書き込みアクセスおよび存在しない場合にデータベースの作成を するには "c" を、そして作成、削除、読込み/書き込みアクセス用には "n" を指定します。

handler

path にアクセスする際に使用する ハンドラの名前。 **dba_popen()** に指定したすべてのオプションが 渡され、その機能を用いることができます。

返り値

成功した場合に正のハンドル、失敗した場合に **FALSE** を返します。

参考

- dba_open()
- dba_close()

---

dba_replace

(PHP 4, PHP 5)

dba_replace — エントリを置換または挿入する

説明

bool **dba_replace** (string $key , string $value , resource $handle)

dba_replace() は、*handle* で 指定したデータベースに *key* および *value* で記述されるエントリを 置換または挿入します。

パラメータ

key

置換するエントリのキー。

value

置換される値。

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dba_exists()
- dba_delete()
- dba_fetch()
- dba_insert()

---

dba_sync

(PHP 4, PHP 5)

dba_sync — データベースを同期する

説明

bool **dba_sync** (resource `$handle`)

dba_sync() は データベースを同期します。サポートされる場合には、 おそらくディスクへの物理的アクセスへのきっかけとなることでしょう。

パラメータ

handle

dba_open() あるいは dba_popen() によって返されたデータベースハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dba_optimize()

目次

- dba_replace — エントリを置換または挿入する
- dba_sync — データベースを同期する

---

dBase 関数

導入

これらの関数は、dBase 形式(dbf) のデータベースに保存されたレコードに アクセスすることを可能にします。

dBase ファイルは、固定長の単純なシーケンシャルファイルです。 レコードはファイルの終りに追加され、削除されたレコードは dbase_pack() が コールされるまで保持されます。

dBase フィールドの型として有効なものは以下のとおりです。

有効なフィールド型

フィールド	dBase 型	フォーマット	補足情報
M	Memo	n/a	この型は PHP ではサポートされていません。 このフィールドは無視されます。
D	Date	*YYYYMMDD*	フィールドの最大長は 8 です。
N	Number	数字	長さと精度（小数点以下の桁数）を指定する必要があります。
C	String	文字列	長さを指定する必要があります。データを取得する際、指定した長さに 満たない文字列にはその右側に空白文字が付加されます。
L	Boolean	*T* あるいは *Y* が **TRUE**、 *F* あるいは *N* が **FALSE** です。	読み書きは整数値（1 または 0）で行われます。
F	Float	浮動小数点数値	この型のフィールドのサポートは PHP 5.2.0 で追加されました。

警告

これらの関数では、インデックスおよびメモフィールドのサポートは 行われません。ロックのサポートも行われません。ふたつの Web サーバ プロセスが同時に同じ dBase ファイルを修正しようとした場合、 データベースはほぼ確実に駄目になってしまうでしょう。

商用データベースとして dBase ファイルを使用しないことを推奨します。 本物の SQL サーバーを替わりに選んでください。通常、» MySQL または » Postgres PHP で使用されています。dBase サポートの目的は、 Web データベースのデータのインポート・エクスポートを行うことです。 なぜなら、Windows の表計算ソフトや統合ソフトでこのフォーマットが 通常サポートされているからです。

インストール手順

付属する dbase ライブラリを有効にしてこれらの関数を使用するには、 --enable-dbase オプションを 指定して PHP をコンパイルする必要があります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

例

このリファレンスの例の多くは dBase データベースを必要とします。ここでは dbase_create() の例で作成した */tmp/test.dbf* を利用しています。

定義済み定数

定数は定義されていません。

---

dbase_add_record

(PHP 4, PHP 5)

dbase_add_record — データベースにレコードを追加する

説明

bool **dbase_add_record** (int `$dbase_identifier` , array `$record`)

与えられたデータをデータベースに追加します。

パラメータ

`dbase_identifier`

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

`record`

データの配列。要素の数がデータベースのフィールド数と一致している 必要があります。それ以外の場合は**dbase_add_record()** は失敗します。

注意: dbase_get_record() が返す値をこのパラメータに 使用する際は、*deleted* という名前のキーを リセットすることを忘れないでください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 dBase データベースにレコードを追加する

```
<?php

// read-write モードでオープンする
$db = dbase_open('/tmp/test.dbf', 2);

if ($db) {
  dbase_add_record($db, array(
       date('Ymd'),
       'Maxim  Topolov',
       '23',
       'max@example.com',
       'T'));
  dbase_close($db);
}

?>
```

参考

- dbase_delete_record()
- dbase_replace_record()

---

dbase_close

(PHP 4, PHP 5)

dbase_close — データベースを閉じる

説明

bool **dbase_close** (int `$dbase_identifier`)

指定されたデータベースリンク ID を閉じます。

パラメータ

`dbase_identifier`

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 dBase データベースファイルを閉じる

```
<?php

// read-only モードでオープンする
$db = dbase_open('/tmp/test.dbf', 0);

if ($db) {
  // データを読む ..

  dbase_close($db);
}

?>
```

参考

- dbase_open()
- dbase_create()

---

dbase_create

(PHP 4, PHP 5)

dbase_create — データベースを作成する

説明

int **dbase_create** (string `$filename` , array `$fields`)

dbase_create() は、指定された定義で dBase データベースを作成します。

> **注意**: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。
>
> **注意**: この関数の動作は、open_basedir の設定に依存します。

パラメータ

`filename`

> データベースの名前。dBase がデータを格納するファイルの 相対パスあるいは絶対パス。

`fields`

> 配列の配列で、各配列はデータベースにおけるひとつのフィールドの フォーマットを記述します。各フィールドは、名前・フィールド型を 示す文字・長さ（オプション）・精度（オプション）から構成されます。
>
> > **注意**: フィールド名の長さは、10 文字までに制限されています。

返り値

データベースの作成に成功した場合にリンク ID 、エラーが発生した場合に **FALSE** を返します。

例

Example#1 dBase データベースファイルの作成

```
<?php

// データベースの定義
$def = array(
   array("date",     "D"),
   array("name",     "C",  50),
   array("age",      "N",   3, 0),
   array("email",    "C", 128),
   array("ismember", "L")
);

// 作成
if (!dbase_create('/tmp/test.dbf', $def)) {
   echo "Error, can't create the database¥n";
}

?>
```

参考

- dbase_open()
- dbase_close()

---

dbase_delete_record

(PHP 4, PHP 5)

dbase_delete_record — データベースからレコードを削除する

説明

bool **dbase_delete_record** (int `$dbase_identifier` , int `$record_number`)

データベースから削除したいレコードをマークします。

> **注意**: 実際にデータベースからレコードを削除するには、 dbase_pack() もコールする必要があります。

パラメータ

`dbase_identifier`

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

`record_number`

1 からデータベース内のレコード数 （dbase_numrecords() が返す） までの範囲の整数値。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- dbase_add_record()
- dbase_replace_record()

---

dbase_get_header_info

(PHP 5)

dbase_get_header_info — データベースのヘッダ情報を得る

説明

array **dbase_get_header_info** (int `$dbase_identifier`)

指定したデータベースリンク ID のカラム構造についての情報を返します。

パラメータ

`dbase_identifier`

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

返り値

データベースの各カラムについての要素を格納した配列を返します。 配列の添字は 0 から始まります。

配列の各要素には、以下に述べるような形式の連想配列で カラムの情報が格納されます。

name
: カラム名。

type
: カラムのデータ型を可読形式（例: date、boolean など）で格納。

length

カラムのバイト数。

precision
: カラムの数値の精度。

format
: カラムに対して推奨される printf() でのフォーマット指定。

offset
: 行の最初からのバイトオフセット。

ヘッダ情報が読み込めない場合には **FALSE** を返します。

例

Example#1 dBase データベースファイルの情報の表示

```
<?php
// dbase ファイルのパス
$db_path = "/tmp/test.dbf";

// dbase ファイルのオープン
$dbh = dbase_open($db_path, 0)
   or die("Error! Could not open dbase database file '$db_path'.");

// カラム情報の取得
$column_info = dbase_get_header_info($dbh);

// 情報の表示
print_r($column_info);
?>
```

---

dbase_get_record_with_names

(PHP 4, PHP 5)

dbase_get_record_with_names — データベースからレコードを連想配列として得る

説明

array **dbase_get_record_with_names** (int `$dbase_identifier` , int `$record_number`)

dBase データベースから、レコードを連想配列として取得します。

パラメータ

`dbase_identifier`

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

`record_number`

レコードのインデックス。

返り値

レコードの連想配列を返します。これには *deleted* という名前のキーも含まれており、もしレコードに削除マークがつけられている 場合にはその値が 1 となります （dbase_delete_record() を参照ください）。

各フィールドは、適切な PHP の型に変換されます。ただし以下の例外を除きます。

- 日付は文字列のままとなります。
- オーバーフローを発生するような整数（> 32 ビット）は、 文字列として返されます。

エラー時には **dbase_get_record_with_names()** は **FALSE** を返します。

例

Example#1 データベースに登録されているメンバの一覧表示

```
<?php
// read-only モードでオープンする
$db = dbase_open('/tmp/test.dbf', 0);

if ($db) {
  $record_numbers = dbase_numrecords($db);
  for ($i = 1; $i <= $record_numbers; $i++) {
      $row = dbase_get_record_with_names($db, $i);
      if ($row['ismember'] == 1) {
```

```
            echo "Member #$i: " . trim($row['name']) . "¥n";
        }
    }
}
?>
```

参考

- dbase_get_record()

---

dbase_get_record

(PHP 4, PHP 5)

dbase_get_record — データベースからレコードを配列形式で得る

説明

array **dbase_get_record** (int `$dbase_identifier` , int `$record_number`)

データベースから、レコードを配列形式で取得します。

パラメータ

`dbase_identifier`

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

`record_number`

レコードのインデックス。

返り値

レコードの連想配列を返します。これには *deleted* という名前のキーも含まれており、もしレコードに削除マークがつけられている 場合にはその値が 1 となります （dbase_delete_record() を参照ください）。

各フィールドは、適切な PHP の型に変換されます。ただし以下の例外を除きます。

- 日付は文字列のままとなります。
- オーバーフローを発生するような整数（> 32 ビット）は、 文字列として返されます。

エラー時には **dbase_get_record()** は **FALSE** を返します。

参考

- dbase_get_record_with_names()

---

dbase_numfields

(PHP 4, PHP 5)

dbase_numfields — データベースのフィールド数を得る

説明

int **dbase_numfields** (int `$dbase_identifier`)

指定したデータベースにおけるフィールド（カラム）数を取得します。

注意: フィールド番号は 0 から dbase_numfields($db)-1 までで、 一方レコード番号は 1 から dbase_numrecords($db) までです。

パラメータ

`dbase_identifier`

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

返り値

データベースのフィールド数を返します。エラーが発生した場合には **FALSE** を返します。

例

Example#1 dbase_numfields() の例

```
<?php

$rec  = dbase_get_record($db, $recno);
$nf   = dbase_numfields($db);
for ($i = 0; $i < $nf; $i++) {
   echo $rec[$i], "¥n";
}

?>
```

参考

- dbase_numrecords()

---

dbase_numrecords

(PHP 4, PHP 5)

dbase_numrecords — データベースのレコード数を得る

説明

int **dbase_numrecords** (int `$dbase_identifier`)

指定したデータベースのレコード（行）数を取得します。

> **注意**: レコード番号は 1 から dbase_numrecords($db) までで、 一方フィールド番号は 0 から dbase_numfields($db)-1 までです。

パラメータ

`dbase_identifier`

> データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

返り値

データベースのレコード数を返します。エラーが発生した場合には **FALSE** を返します。

例

Example#1 データベースの全レコードをループする

```
<?php

// read-only モードでオープンする
$db = dbase_open('/tmp/test.dbf', 0);

if ($db) {
  $record_numbers  = dbase_numrecords($db);
  for ($i = 1; $i <= $record_numbers; $i++) {
     // 各レコードに対して、ここで何かをする
  }
}

?>
```

参考

- dbase_num_fields()

---

dbase_open

(PHP 4, PHP 5)

dbase_open — データベースをオープンする

説明

int **dbase_open** (string `$filename` , int `$mode`)

dbase_open() は、指定したアクセスモードで dBase データベースをオープンします。

> **注意**: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

> **注意**: この関数の動作は、open_basedir の設定に依存します。

パラメータ

`filename`

> データベースの名前。dBase がデータを格納するファイルの 相対パスあるいは絶対パス。

`mode`

> open() システムコールに対応する整数値（通常、0 は読み込みのみ可・ 1 は書き込みのみ可・2 は読み書き両方可を意味します）。
>
> > **注意**: dBase ファイルを write-only モードでオープンしようとすると、 ヘッダ情報を読み込むことができないために関数は失敗します。 つまり、*mode* に 1 を指定することはできません。

例

Example#1 dBase データベースファイルのオープン

```
<?php

// read-only モードでオープンする
$db = dbase_open('/tmp/test.dbf', 0);

if ($db) {
  // データを読み込む ..

  dbase_close($db);
}

?>
```

返り値

データベースのオープンに成功した場合にリンク ID 、エラーが発生した場合に **FALSE** を返します。

参考

- dbase_create()
- dbase_close()

---

dbase_pack

(PHP 4, PHP 5)

dbase_pack — データベースを圧縮する

説明

bool **dbase_pack** (int `$dbase_identifier`)

dbase_delete_record() を使用して削除マークがつけられた レコードを完全に削除することで、指定したデータベースを圧縮します。

パラメータ

`dbase_identifier`

> データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 dBase データベースを空にする

```php
<?php

// read-write モードでオープンする
$db = dbase_open('/tmp/test.dbf', 2);

if ($db) {
  $record_numbers  = dbase_numrecords($db);
  for ($i = 1; $i <= $record_numbers; $i++) {
      dbase_delete_record($db, $i);
  }
  // データベースを消す
  dbase_pack($db);
}

?>
```

参考

- dbase_delete_record()

---

dbase_replace_record

(PHP 4, PHP 5)

dbase_replace_record — データベースのレコードを置換する

説明

bool **dbase_replace_record** (int `$dbase_identifier` , array `$record` , int `$record_number`)

データベースの指定したレコードを、指定した値で置換します。

パラメータ

dbase_identifier

データベースのリンク ID 。dbase_open() あるいは dbase_create() によって返されます。

record

データの配列。要素の数がデータベースのフィールド数と一致している 必要があります。それ以外の場合は**dbase_replace_record()** は失敗します。

> **注意**: dbase_get_record() が返す値をこのパラメータに 使用する際は、*deleted* という名前のキーを リセットすることを忘れないでください。

record_number

1 からデータベース内のレコード数 （dbase_numrecords() が返す） までの範囲の整数値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 データベースのレコードの更新

```php
<?php

// read-write モードでオープンする
$db = dbase_open('/tmp/test.dbf', 2);

if ($db) {
  // 変更前の行を取得する
  $row = dbase_get_record_with_names($db, 1);

  // 'deleted' エントリを削除する
  unset($row['deleted']);

  // 現在のタイムスタンプでフィールドを更新する
  $row['date'] = date('Ymd');

  // レコードを置換する
  dbase_replace_record($db, $row, 1);
  dbase_close($db);
}

?>
```

参考

- dbase_add_record()
- dbase_delete_record()

目次

DBM 関数 [非推奨]

導入

これらの関数により、レコードを dbm 形式 のデータベースに 格納できるようになります。この形式のデータベース (組み込みの フラットファイルライブラリと同様に、Berkeley DB・» GDBM・その他のシステムライブラリにより サポートされています) は、(リレーショナルデータベースにより サポートされる事細かなレコード形式ではなく)キーと値の組み合わせを 格納します。

> **注意**: dbm サポートは古くなっており、 (dbm形式の)データベース抽象化レイヤ関数 を代わりに使用することが推奨されています。

> **注意**: この拡張モジュールは PHP から削除されました。PHP 5.0.0 以降のバージョンには存在しません。

要件

この関数を使用するには、使用するデータベースのサポートを指定して PHP をコンパイルする必要があります。サポートされるデータベースの 一覧を参照ください。

インストール手順

以下の関数を使用するには、--with-db オプションを指定し、dbm サポートを 有効にして PHP をコンパイルする必要があります。 さらに、使用するデータベースのサポート トを確認する必要があります。また、いくつかのシステムライブラリを 使用することが可能です。

実行時設定

設定ディレクティブは定義されていません。

リソース型

関数 dbmopen() は、データベース ID を返します。 この ID は、他の dbm 関数で使用されます。

定義済み定数

定数は定義されていません。

例

Example#1 DBM の例

```
<?php

$dbm = dbmopen("lastseen", "w");
if (dbmexists($dbm, $userid)) {
    $last_seen = dbmfetch($dbm, $userid);
```

```
} else {
    dbminsert($dbm, $userid, time());
}
do_stuff();
dbmreplace($dbm, $userid, time());
dbmclose($dbm);

?>
```

---

dblist

(PHP 4)

dblist — 使用されている DBM 互換ライブラリの一覧を返す

説明

string **dblist** (void)

GDBM のバージョンを返します。

返り値

GDBM のバージョンを表す文字列を返します。

例

Example#1 コマンドラインで情報を取得する

```
[marcus@zaphod marcus]$ php -r 'echo dblist();'
This is GDBM version 1.8.0, as of May 19, 1999.
```

---

dbmclose

(PHP 4)

dbmclose — dbm データベースを閉じる

説明

bool **dbmclose** (resource `$dbm_identifier`)

指定したデータベースをアンロックして閉じます。

パラメータ

`dbm_identifier`

dbmopen() が返す DBM リンク識別子。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dbmopen()

---

dbmdelete

(PHP 4)

dbmdelete — DBM データベースから値を削除する

説明

bool **dbmdelete** (resource `$dbm_identifier` , string `$key`)

データベース中の、*key* に対応する値を削除します。

パラメータ

dbm_identifier

dbmopen() が返す DBM リンク識別子。

key

キーを表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 データベース中にキーが存在しない場合は **FALSE** を返します。

参考

- dbminsert()

---

dbmexists

(PHP 4)

dbmexists — DBM データベース中に、キーに対応する値があるかどうかを調べる

説明

bool **dbmexists** (resource `$dbm_identifier` , string `$key`)

キー *key* が指す値が存在すれば **TRUE** を返します。

パラメータ

dbm_identifier

dbmopen() が返す DBM リンク識別子。

key

キーを表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dbmfetch()

---

dbmfetch

(PHP 4)

dbmfetch — DBM データベースからキーの値を取り出す

説明

string **dbmfetch** (resource `$dbm_identifier` , string `$key`)

キー *key* が指す値を返します。

パラメータ

dbm_identifier

dbmopen() が返す DBM リンク識別子。

key

キーを表す文字列。

返り値

key に関連付けられた値を返します。

---

dbmfirstkey

(PHP 4)

dbmfirstkey — DBM データベースから最初のキーを取り出す

説明

string **dbmfirstkey** (resource `$dbm_identifier`)

データベースから最初のキーを取り出します。データベースは順序を 保証しないハッシュテーブルを使って構築されている可能性があるので、 返される値の順序は保証されないことに注意してください。

パラメータ

dbm_identifier

dbmopen() が返す DBM リンク識別子。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dbmnextkey()

---

dbminsert

(PHP 4)

dbminsert — DBM データベースへ値を挿入する

説明

int **dbminsert** (resource `$dbm_identifier` , string `$key` , string `$value`)

データベースに、指定したキー(key)に対応する値を追加します。

パラメータ

dbm_identifier

dbmopen() が返す DBM リンク識別子。

key

追加するキー。

value

追加するキーの値。

返り値

データベースがリードオンリーでオープンされていた場合は -1 を返し、 挿入が成功すれば 0 を返します。指定されたキーが存在する場合は 1 を 返します（値を置き換えるには dbmreplace() を 使ってください）。

参考

- dbmreplace()
- dbmdelete()

---

dbmnextkey

(PHP 4)

dbmnextkey —

説明

string **dbmnextkey** (resource `$dbm_identifier` , string `$key`)

key の次のキーを返します。

パラメータ

dbm_identifier

dbmopen() が返す DBM リンク識別子。

key

前のキー。

返り値

次のキーが存在すれば、それを返します。

例

dbmfirstkey() をコールし、引き続き **dbmnextkey()** をコールすれば、 dbm データベース上のすべてのキー/値 の組み合わせを取り出すことができます。たとえば、以下のようにします。

Example#1 DBM データベース上のすべての キー/値 ペアの取得

```
<?php

$key = dbmfirstkey($dbm_id);
while ($key) {
    echo "$key = " . dbmfetch($dbm_id, $key) . "¥n";
    $key = dbmnextkey($dbm_id, $key);
}

?>
```

参考

- dbmfirstkey()

---

dbmopen

(PHP 4)

dbmopen — DBM データベースをオープンする

説明

resource **dbmopen** (string `$filename` , string `$flags`)

指定した DBM データベースを、指定したモードでオープンします。

パラメータ

filename

オープンする DBM ファイルのフルパスでのファイル名。

flags

ファイルオープンモードで、*r*、 *n*、*c* あるいは *w* のいずれかを指定します。これらはそれぞれ 読み込み専用、新規 (読み書き可能。同名の既存のファイルを上書きする可能性があります)、 作成 (読み書き可能。同名の既存のファイルは上書きしません)、 そして読み書き可能を表します。

返り値

成功した場合は他の DBM 関数に渡す ID、失敗した場合は **FALSE** を返します。

注意

注意: NDBM サポートが有効な場合、実際に NDBM が *filename.dir* と *filename.pag* ファイルを作成します。GDBM は、 PHP 組み込みのフラット・ファイル機能と同様に 1 つのファイルしか 使いません。Berkeley DB は、*filename.db* ファイルを作成します。
PHP では、DBM ライブラリ自体が行うファイルロックに加えて 自分自身でもファイルロックを行うということに注意してください。
PHP では自分で生成した *.lck* ファイルを 削除しません。PHP では単純にこれらの ファイルをファイルロックのための固定 inode として使用します。 DBM ファイルに関する詳細情報は、Unix の man ページを参照するか、 » GNU の GDBM を取得してください。

注意: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- dbmclose()

---

dbmreplace

(PHP 4)

dbmreplace — DBM データベース中の値を置き換える

説明

int **dbmreplace** (resource `$dbm_identifier` , string `$key` , string `$value`)

データベース中の、指定されたキーに対応する値を置き換えます。

データベース中にキーが存在しない場合は、この関数はキーも追加します。

パラメータ

`dbm_identifier`

dbmopen() が返す DBM リンク識別子。

`key`

置き換えるキー。

`value`

新しい値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- dbminsert()

目次

- dbmnextkey — 説明
- dbmopen — DBM データベースをオープンする
- dbmreplace — DBM データベース中の値を置き換える

---

dbx 関数

導入

dbx モジュールは、データベース抽象化レイヤ（db 'X' の 'X' は、 サポートされるデータベースの一つを意味します）のことです。dbx 関数 により、サポートされる全てのデータベースを単一の呼出表記により アクセスすることが可能になります。dbx 関数自体は、データベースへの 直接のインターフェイスを有しませんが、それらのデータベースをサポート するために使用されるモジュールへのインターフェイスを有します。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.0.

要件

dbx モジュールでデータベースを使用可能とするには、そのモジュールが PHP にリンクされるかロードされる必要があり、そのデータベースを dbx モジュールがサポートされている必要があります。現在 以下のデータベースがサポートされていますが、他のデータベースも 追加される予定です。

- FrontBase (PHP 4.1.0 以降で利用可能)
- Microsoft SQL Server
- MySQL
- ODBC
- PostgreSQL
- Sybase-CT (PHP 4.2.0 以降で利用可能)
- Oracle (oci8) (PHP 4.3.0 以降で利用可能)
- SQLite (PHP 5)

dbx にデータベースのサポートを追加するためのドキュメントは、 » http://www.guidance.nl/php/dbx/doc/ にあります。

インストール手順

これらの関数を利用可能にするには、 --enable-dbx オプションを使用して dbx サポートを有効にし、また、使用するデータベースに関するオプション、 例えば MySQL の場合は --with-mysql=[DIR] も指定して PHP をコンパイルする必要があります。 他のサポートされるデータベースを dbx モジュールで動作させるには、 個別のドキュメントを参照してください。

実行時設定

php.ini の設定により動作が変化します。

DBX 設定オプション

名前	デフォルト	変更の可否	変更履歴
dbx.colnames_case	"unchanged"	PHP_INI_SYSTEM	PHP 4.3.0 以降で有効です。PHP 5.1.0 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

`dbx.colnames_case` string

> カラム名は、変更されず("unchanged"の場合)に返すか、大文字 ("uppercase"の場合)または小文字("lowercase"の場合)に変換すること ができます。このディレクティブは、dbx_query() のフラグで上書きすることが可能です。

リソース型

dbx モジュールでは2種類のリソース型があります。最初のリソースは、 データベース接続用のリンク object で、2 番目はクエリ結果を 結果 object です。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

DBX_MYSQL (integer)
DBX_ODBC (integer)
DBX_PGSQL (integer)
DBX_MSSQL (integer)
DBX_FBSQL (integer)
DBX_OCI8 (integer) (PHP 4.3.0 以降で有効)
DBX_SYBASECT (integer)
DBX_SQLITE (integer) (PHP 5)
DBX_PERSISTENT (integer)
DBX_RESULT_INFO (integer)
DBX_RESULT_INDEX (integer)
DBX_RESULT_ASSOC (integer)
DBX_RESULT_UNBUFFERED (integer) (PHP 5)
DBX_COLNAMES_UNCHANGED (integer) (PHP 4.3.0 以降で有効)
DBX_COLNAMES_UPPERCASE (integer) (PHP 4.3.0 以降で有効)
DBX_COLNAMES_LOWERCASE (integer) (PHP 4.3.0 以降で有効)
DBX_CMP_NATIVE (integer)
DBX_CMP_TEXT (integer)
DBX_CMP_NUMBER (integer)
DBX_CMP_ASC (integer)
DBX_CMP_DESC (integer)

---

dbx_close

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_close — オープンされた接続/データベースを閉じる

説明

int **dbx_close** (object `$link_identifier`)

パラメータ

`link_identifier`

閉じたい DBX リンクオブジェクト。

返り値

成功した場合に 1、エラーの場合に 0 を返します。

例

Example#1 dbx_close() の例

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die("接続できませんでした");

echo "接続に成功しました";
dbx_close($link);
?>
```

注意

注意: モジュール固有のドキュメントも参照ください。

参考

- dbx_connect()

---

dbx_compare

(PHP 4 >= 4.0.7, PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_compare — ソートするために二つのレコードを比較する

説明

int **dbx_compare** (array $row_a , array $row_b , string $column_key [, int $flags])

dbx_compare() は dbx_sort() のヘルパ関数で、 独自のソート関数を作成しやすくします。

パラメータ

row_a

最初の行。

row_b

二番目の行。

column_key

比較するカラム。

flags

flags によって比較の方向を指定します。

- **DBX_CMP_ASC** - 昇順
- **DBX_CMP_DESC** - 降順

また、比較の型についても指定します。

- **DBX_CMP_NATIVE** - 型の変換を行いません
- **DBX_CMP_TEXT** - 文字列として比較します
- **DBX_CMP_NUMBER** - 数値として比較します

方向からひとつ、型の定数からひとつを選び、ビット OR 演算子 (¦) で 組み合わせます。*flags* パラメータのデフォルト値は **DBX_CMP_ASC** ¦ **DBX_CMP_NATIVE** です。

返り値

row_a[$column_key] が *row_b[$column_key]* に等しい場合に *0*、 前者が後者より大きいかあるいは小さい場合にそれぞれ *1* あるいは *-1*、 もし **DBX_CMP_DESC** が設定されていればその逆を返します。

例

Example#1 dbx_compare() の例

```
<?php
function user_re_order($a, $b)
{
    $rv = dbx_compare($a, $b, "parentid", DBX_CMP_DESC);
    if (!$rv) {
        $rv = dbx_compare($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link    = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("接続できませんでした");

$result = dbx_query($link, "SELECT id, parentid, description FROM table ORDER BY id");
    // $result のデータは id で並べ替えられます

dbx_sort($result, "user_re_order");
    // $result のデータは、まず parentid の降順で並べ替えられ、次に id で並べ替えられます

dbx_close($link);
?>
```

参考

- dbx_sort()

---

dbx_connect

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_connect — 接続/データベースをオープンする

説明

object **dbx_connect** (mixed $module , string $host , string $database , string $username , string $password [, int $persistent])

データベースへの接続をオープンします。

パラメータ

module

module パラメータには文字列または定数を設定します。 定数の使用が推奨されます。指定可能な値を以下に示しますが、これは 該当するモジュールが実際にロードされている場合のみ動作することに注意してください。

- **DBX_MYSQL** あるいは "mysql"
- **DBX_ODBC** あるいは "odbc"
- **DBX_PGSQL** あるいは "pgsql"
- **DBX_MSSQL** あるいは "mssql"
- **DBX_FBSQL** あるいは "fbsql" (PHP 4.1.0 以降で使用可能)
- **DBX_SYBASECT** あるいは "sybase_ct" (PHP 4.2.0 以降で使用可能)
- **DBX_OCI8** あるいは "oci8" (PHP 4.3.0 以降で使用可能)
- **DBX_SQLITE** あるいは "sqlite" (PHP 5)

host

SQL サーバのホスト。

database

データベース名。

username

ユーザ名。

password

パスワード。

persistent

persistent パラメータに **DBX_PERSISTENT** を設定すると、持続的接続が作成されます。

host 、*database* 、 *username* および *password* のパラメータを受け付けますが、抽象化されたモジュールの接続関数によっては これらがすべて使われるわけではないこともあります。

返り値

dbx_connect() は成功時にオブジェクト、エラー時に **FALSE** を返します。接続は確立したもののデータベースが選択できなかった場合には、 接続はクローズされて **FALSE** を返します。

返される *object* は 3 つのプロパティを有します。

database
: 現在選択されているデータベースの名前。

handle
: 接続されたデータベースの有効なハンドルで、モジュール固有の関数に （必要に応じて）使用されます。

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password");
mysql_close($link->handle); //  dbx_close($link)  の方が良いかもしれません
?>
```

module
: dbx の内部でのみ使用され、上で述べたモジュール番号を表します。

例

Example#1 dbx_connect() example

```
<?php
$link = dbx_connect(DBX_ODBC, "", "db", "username", "password", DBX_PERSISTENT)
    or die("接続できませんでした");

echo "接続に成功しました";
dbx_close($link);
?>
```

注意

注意: モジュール固有のドキュメントも参照ください。

参考

- dbx_close()

---

dbx_error

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_error — 使用するモジュールの最新の関数コールにおけるエラーメッセージを出力する

説明

string **dbx_error** (object `$link_identifier`)

dbx_error() は、直近のエラーメッセージを返します。

パラメータ

`link_identifier`

dbx_connect() が返す DBX リンクオブジェクト。

返り値

抽象化されたモジュール（例えば mysql モジュール）の直近の関数コールからエラーメッセージを有する文字列を返します。 同じモジュールに複数の接続がある場合、 最後のエラーのみが取得されます。別のモジュールに接続がある場合、 （*`link_identifier`* パラメータで）指定した モジュールに関する直近のエラーのみが返されます。

例

Example#1 dbx_error() の例

```
<?php
$link    = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die("接続に失敗しました");

$result  = dbx_query($link, "select  id  from  non_existing_table");
if ($result  == 0) {
    echo dbx_error($link);
}
dbx_close($link);
?>
```

注意

注意: モジュール固有のドキュメントも参照ください。
Microsoft SQL Server に関するエラーメッセージは、実際には mssql_get_last_message() 関数の結果となります。
Oracle (oci8) に関するエラーメッセージは、（まだ）実装されていません。

---

dbx_escape_string

(PHP 4 >= 4.3.0, PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_escape_string — SQL ステートメントで安全に使用できるように文字列をエスケープする

説明

string **dbx_escape_string** (object `$link_identifier` , string `$text`)

指定した文字列をエスケープし、SQL ステートメントで安全に使用できるようにします。

パラメータ

`link_identifier`

dbx_connect() が返す DBX リンクオブジェクト。

text

エスケープする文字列。

返り値

テキストを返します。 必要なら（クォートやバックスラッシュなど）エスケープ処理を行います。 エラー時には **NULL** を返します。

例

Example#1 dbx_escape_string() の例

```
<?php
$link    = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die("接続できませんでした");

$text  = dbx_escape_string($link, "It¥'s  quoted  and  backslashed  (¥¥).");
$result  = dbx_query($link, "insert  into  tbl  (txt)  values  ('"  . $text  . "')");
if  ($result  ==  0)  {
     echo dbx_error($link);
}
dbx_close($link);
?>
```

参考

- dbx_query()

---

dbx_fetch_row

(PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_fetch_row — **DBX_RESULT_UNBUFFERED** フラグを指定した クエリ結果から、行を取得する

説明

mixed **dbx_fetch_row** (object `$result_identifier`)

dbx_fetch_row() は、 **DBX_RESULT_UNBUFFERED** フラグを指定したクエリ結果から、行を取得します。

クエリで **DBX_RESULT_UNBUFFERED** が指定されていない場合、 **dbx_fetch_row()** は失敗します。なぜなら、すでにすべての 行は取得されて data プロパティに格納されているからです。

副作用として、**dbx_fetch_row()** のコールのたびに クエリ結果オブジェクトの rows プロパティの 値が加算されます。

パラメータ

result_identifier

dbx_query() が返す結果セット。

返り値

成功した場合にオブジェクトを返します。このオブジェクトには、 dbx_query() の結果における data プロパティの内容と同じ情報が含まれ、dbx_query() で設定した内容に応じてインデックスあるいはフィールド名でアクセスが可能です。

失敗した場合 (例: 行がもうない場合) には *0* を返します。

例

Example#1 返り値を処理する方法

```
<?php
$result  =  dbx_query($link,  'SELECT  id,  parentid,  description  FROM  table',  DBX_RESULT_UNBUFFERED);

echo  "<table>¥n";
while  ($row  =  dbx_fetch_row($result))  {
     echo  "<tr>¥n";
     foreach  ($row  as  $field)  {
          echo  "<td>$field</td>";
     }
     echo  "</tr>¥n";
}
echo  "</table>¥n";
?>
```

参考

- dbx_query()

---

dbx_query

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_query — クエリを送信し、(ある場合には)結果を全て取得する

説明

mixed **dbx_query** (object `$link_identifier` , string `$sql_statement` [, int `$flags`])

クエリを送信し、すべての結果を取得します。

パラメータ

`link_identifier`

dbx_connect() が返す DBX リンクオブジェクト。

`sql_statement`

SQL 文。

`flags`

`flags` パラメータは、返される情報の量を制御する ために使用されます。以下の定数の組み合わせを、論理 OR 演算子 (¦) で連結したものとなります。DBX_COLNAMES_* フラグは *php.ini* の dbx.colnames_case 設定を上書きします。

DBX_RESULT_INDEX
これは*常に*設定され、結果に二次元配列の data プロパティを含めます。たとえば *data[2][3]* では *2* が行番号 （レコード番号）を表し、*3* がカラム番号 （フィールド番号）を表します。最初の行やカラムの番号は 0 です。 **DBX_RESULT_ASSOC** が指定された場合、返されるオブジェクトには **DBX_RESULT_INFO** に関連する 情報が（指定していなくても）含まれます。

DBX_RESULT_INFO
カラムに関する情報、つまりフィールド名とフィールドの型を含めます。

DBX_RESULT_ASSOC
返されるオブジェクトの data プロパティの キーとして、関連するカラム名の値がアクセスされます。 関連付けられた結果は数値添字となっているので、 *data[0][0]* を変更すると *data[0]['最初のカラムのフィールド名']* も同様に 変更されます。

DBX_RESULT_UNBUFFERED (PHP 5)
このフラグは data プロパティを作成せず、 rows プロパティの初期値を 0 に設定します。 結果セットが大きくなる場合にこのフラグを使用し、 dbx_fetch_row() を使用して結果を 1 行ごとに 取得してください。 dbx_fetch_row() 関数は、このクエリで設定した フラグの内容を反映じた行を返します。また、この関数がコールされる たびに rows が更新されます。

DBX_COLNAMES_UNCHANGED (PHP 4.3.0 以降で使用可能)
返されるカラム名の大文字小文字を変更しません。

DBX_COLNAMES_UPPERCASE (PHP 4.3.0 以降で使用可能)
返されるカラム名を大文字に変更します。

DBX_COLNAMES_LOWERCASE (PHP 4.3.0 以降で使用可能)
返されるカラム名を小文字に変更します。

`flags` パラメータの実際の値にかかわらず、常に **DBX_RESULT_INDEX** が有効となることに注意しましょう。 つまり、結果的に使用可能なのは以下の組み合わせだけであるということです。

- **DBX_RESULT_INDEX**
- **DBX_RESULT_INDEX** ¦ **DBX_RESULT_INFO**
- **DBX_RESULT_INDEX** ¦ **DBX_RESULT_INFO** ¦ **DBX_RESULT_ASSOC** - *`flags`* を指定しなかった場合、これがデフォルトです。

返り値

dbx_query() は、成功した場合に オブジェクトあるいは *1*、失敗した場合に *0* を返します。 結果オブジェクトが返されるのは、*`sql_statement`* で指定されたクエリが結果セットを生成する場合（例: SELECT クエリ。結果が 0 件の場合も含む）のみです。

返される*object* は、*`flags`* の設定により 4 つまたは 5 つのプロパティを保持します。

handle

接続したデータベースの有効なハンドルで、モジュール専用関数で （必要ならば）使用されます。

```
<?php
```

```
$result  = dbx_query($link, "SELECT  id  FROM  table");
mysql_field_len($result->handle, 0);
?>
```

cols および rows

これらは、それぞれカラム数（フィールド数）および行数（レコード数） を表します。

```
<?php
$result  = dbx_query($link, 'SELECT  id  FROM  table');
echo $result->rows; //  レコード数
echo $result->cols; //  フィールド数
?>
```

info (オプション)

flags パラメータに **DBX_RESULT_INFO** あるいは **DBX_RESULT_ASSOC** が指定されている場合のみ 返されます。2 次元の配列で、2 つのキー（*name* および *type*）を持ち、カラムの情報を取得するために 使用します。

Example#1 各フィールドの名前と型を一覧表示する

```
<?php
$result  = dbx_query($link, 'SELECT  id  FROM  table',
                     DBX_RESULT_INDEX  | DBX_RESULT_INFO);

for ($i  = 0; $i  < $result->cols; $i++ ) {
     echo $result->info['name'][$i] . "¥n";
     echo $result->info['type'][$i] . "¥n";
}
?>
```

data

このプロパティには結果のデータが含まれます。*flags* の設定によってはこのデータはカラム名と関連付けられているでしょう。**DBX_RESULT_ASSOC** が設定されていた場合、 *$result->data[2]["field_name"]* のように使用可能です。

Example#2 data プロパティの内容を HTML テーブルで表示する

```
<?php
$result  = dbx_query($link, 'SELECT  id,  parentid,  description  FROM  table');

echo  "<table>¥n";
foreach  ($result->data  as  $row) {
     echo "<tr>¥n";
     foreach ($row  as  $field) {
          echo "<td>$field</td>";
     }
     echo "</tr>¥n";
}
echo  "</table>¥n";
?>
```

Example#3 UNBUFFERED クエリの処理法

```
<?php

$result  = dbx_query  ($link, 'SELECT  id,  parentid,  description  FROM  table', DBX_RESULT_UNBUFFERED);

echo  "<table>¥n";
while  ($row  = dbx_fetch_row($result)) {
     echo "<tr>¥n";
     foreach ($row  as  $field) {
          echo "<td>$field</td>";
     }
     echo "</tr>¥n";
}
echo  "</table>¥n";

?>
```

例

Example#4 返り値を処理する方法

```
<?php
$link    = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("接続できませんでした");

$result  = dbx_query($link, 'SELECT  id,  parentid,  description  FROM  table');

if (is_object($result) ) {
    // ...  ここで何かを行います。詳細は以下の例で示します。  ...
    // まず、フィールド名とデータ型を表示します。
    // 次に、返されたフィールド値をもとに表を作成します。
} else {
    exit("クエリに失敗しました");
}

dbx_close($link);
?>
```

注意

注意: モジュール固有のドキュメントも参照ください。

Oracle データベースでは、クエリ結果のカラム名は小文字で返されます。

参考

- dbx_escape_string()
- dbx_fetch_row()
- dbx_connect()

---

dbx_sort

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5, PECL dbx:1.1.0)

dbx_sort — カスタマイズされたソート関数により、dbx_query から結果をソートする

説明

bool **dbx_sort** (object `$result` , string `$user_compare_function`)

dbx_query() のコール結果を、 独自のソート関数で並べ替えます。

パラメータ

result

dbx_query() が返す結果セット。

user_compare_function

ユーザ定義の比較関数。二つの引数を受け取り、 第一引数が第二引数より小さい場合に負の数、等しい場合にゼロ、 大きい場合に正の数を返すものでなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 dbx_sort() の例

```
<?php
function user_re_order($a, $b)
{
    $rv = dbx_compare($a, $b, "parentid", DBX_CMP_DESC);
    if (!$rv) {
        $rv = dbx_compare($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link    = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("接続できませんでした");

$result = dbx_query($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
    // $result のデータは id で並べ替えられます

dbx_sort($result, "user_re_order");
    // $result のデータは、まず parentid の降順で並べ替えられ、次に id で並べ替えられます

dbx_close($link);
?>
```

注意

注意: 可能ならば、**dbx_sort()** を使用するよりも *SQL* の *ORDER BY* 句を使用することを推奨します。

参考

- dbx_compare()

目次

- dbx_connect — 接続/データベースをオープンする
- dbx_error — 使用するモジュールの最新の関数コールにおけるエラーメッセージを出力する
- dbx_escape_string — SQL ステートメントで安全に使用できるように文字列をエスケープする
- dbx_fetch_row — DBX_RESULT_UNBUFFERED フラグを指定した クエリ結果から、行を取得する
- dbx_query — クエリを送信し、(ある場合には)結果を全て取得する
- dbx_sort — カスタマイズされたソート関数により、dbx_query から結果をソートする

ダイレクト IO (DIO) 関数

導入

PHP は、POSIX 標準(第 6 章)に記述されたダイレクト IO 関数を サポートします。これらは、C 言語のストリーム I/O 関数 (fopen(), fread(),...)よりも低レベルの I/O 関数の実装です。 DIO 関数の使用は、あるデバイスを直接制御することが必要な場合のみ 考えるべきです。その他の場合は、標準のファイルシステム関数の方が 適当です。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.0.

この拡張モジュールは、Windows プラットフォーム上では PHP 5.0.0 以降でのみ使用可能です。

要件

外部ライブラリを必要としません。

インストール手順

これらの関数を動作させるには、 --enable-dio を指定して PHP の configure を実行する必要があります。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

c (integer)

F_DUPFD (integer)

F_GETFD (integer)

F_GETFL (integer)

F_GETLK (integer)

F_GETOWN (integer)

F_RDLCK (integer)

F_SETFL (integer)

F_SETLK (integer)

F_SETLKW (integer)

F_SETOWN (integer)

F_UNLCK (integer)

F_WRLCK (integer)

O_APPEND (integer)

O_ASYNC (integer)

O_CREAT (integer)

O_EXCL (integer)

O_NDELAY (integer)

O_NOCTTY (integer)

O_NONBLOCK (integer)

O_RDONLY (integer)

O_RDWR (integer)

O_SYNC (integer)

O_TRUNC (integer)

O_WRONLY (integer)

S_IRGRP (integer)

S_IROTH (integer)

S_IRUSR (integer)

S_IRWXG (integer)

S_IRWXO (integer)

S_IRWXU (integer)

S_IWGRP (integer)

S_IWOTH (integer)

S_IWUSR (integer)

S_IXGRP (integer)

S_IXOTH (integer)

S_IXUSR (integer)

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールでは、1 種類のリソース型、すなわち dio_open() により返されるファイル記述子 が定義されています。

---

dio_close

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_close — fd で指定したファイル記述子を閉じる

説明

void **dio_close** (resource $fd)

関数 **dio_close()** は、ファイル記述子 *fd* を閉じます。

パラメータ

fd

dio_open() が返すファイル記述子。

返り値

値を返しません。

例

Example#1 開いているファイル記述子を閉じる

```php
<?php
$fd = dio_open('/dev/ttyS0', O_RDWR);

dio_close($fd);
?>
```

参考

- dio_open()

---

dio_fcntl

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_fcntl — fd について C ライブラリの fcntl を実行する

説明

mixed **dio_fcntl** (resource $fd , int $cmd [, mixed $args])

関数 **dio_fcntl()** は、ファイル記述子 *fd* において *cmd* で指定された処理を行います。いくつかのコマンドでは、オプションの引数 *args* の指定が必要となります。

> **注意**: この関数は Windows 環境にはまだ実装されていません。

パラメータ

fd

dio_open() が返すファイル記述子。

cmd

以下の処理のいずれか。

- **F_SETLK** - ロックが設定あるいはクリアされます。 ロックが他の誰かに設定されている場合、**dio_fcntl()** は -1 を返します。
- **F_SETLKW** - **F_SETLK** と似ていますが、ロックが他の誰かに設定されている場合は **dio_fcntl()** はそのロックが開放されるまで待ちます。
- **F_GETLK** - **dio_fcntl()** は、他の誰かがロックを妨げる場合、（上で述べた）連想配列を返します。 妨げるものがない場合、キー "type" は **F_UNLCK** に設定されます。
- **F_DUPFD** - *args* 以上で最小のファイル記述子を探し、それを返します。
- **F_SETFL** - ファイル記述子のフラグを *args* で指定した値に設定します。指定できる値は **O_APPEND**、**O_NONBLOCK** あるいは **O_ASYNC** のいずれかです。 **O_ASYNC** を使用するには、 PCNTL 拡張モジュールが必要です。

args

args は連想配列で、 *cmd* が **F_SETLK** あるいは **F_SETLLW** の際に以下のキーを保持します。

- "start" - ロックを開始するオフセット。
- "length" - ロックする領域の大きさ。ゼロはファイルの終端までを意味します。
- "wenth" - l_start の相対位置指定。 **SEEK_SET**、 **SEEK_END** および **SEEK_CUR** のいずれか。
- "type" - ロックの種類。**F_RDLCK**（読み込み ロック）、**F_WRLCK**（書き込みロック）あるいは **F_UNLCK**（ロック解除）のいずれか。

返り値

C ライブラリをコールした結果を返します。

例

Example#1 ロックの設定と解除

```php
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR);

if (dio_fcntl($fd, F_SETLK, Array("type"=>F_WRLCK)) == -1) {
   // ファイル記述子がロックされている
   echo "ロックを解除できません。別のプロセスによってロックされています。";
} else {
```

```
    echo "ロックが正常に設定/解除できました。";
}

dio_close($fd);
?>
```

---

dio_open

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_open — 指定したパーミッション flags と作成許可 mode を指定して 新しいファイルをオープンする

説明

resource **dio_open** (string $filename , int $flags [, int $mode])

dio_open() は、ファイルをオープンして そのファイル記述子を返します。

パラメータ

filename

オープンするファイル名。

flags

flags パラメータには、 以下のフラグの組み合わせを指定します。

- **O_CREAT** - ファイルが存在しなければ 新しいファイルを作成します。
- **O_EXCL** - **O_CREAT** および **O_EXCL** をともに指定すると、 すでにファイルが存在する場合に **dio_open()** が失敗します。
- **O_TRUNC** - すでにファイルが存在し、書き込み アクセス用にオープンされている場合、ファイルの長さをゼロに切り詰めます。
- **O_APPEND** - 書き込み時は、ファイルの終端に 追記します。
- **O_NONBLOCK** - 非ブロッキングモードを指定します。

mode

flags が **O_CREAT** の場合に、*mode* でファイルのモード （作成許可）を指定します。

- **O_RDONLY** - 読み込み専用でファイルをオープンします。
- **O_WRONLY** - 書き込み専用でファイルをオープンします。
- **O_RDWR** - 読み書き両用でファイルをオープンします。

返り値

ファイル記述子を返します。エラー時には **FALSE** を返します。

例

Example#1 ファイル記述子をオープンする

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR | O_NOCTTY | O_NONBLOCK);

dio_close($fd);
?>
```

参考

- dio_close()

---

dio_read

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_read — ファイル記述子からバイトデータを読み込む

説明

string **dio_read** (resource $fd [, int $len])

関数 **dio_read()** は 記述子 *fd* で示されるファイルから *len* バイトを読み込み、それを返します。

パラメータ

fd

dio_open() が返すファイル記述子。

len

読み込むバイト数。指定されなかった場合は **dio_read()** は 1K サイズのブロックを読み込みます。

返り値

fd から読み込んだバイトデータを返します。

参考

- dio_write()

---

dio_seek

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_seek — fd 上で whence から pos に移動する

説明

int **dio_seek** (resource $fd , int $pos [, int $whence])

関数 **dio_seek()** は、指定されたファイル記述子の ファイル内の位置を変更する際に使用されます。

パラメータ

fd

dio_open() が返すファイル記述子。

pos

新しい位置。

whence

位置 *pos* をどのように解釈するかを指示します。

- **SEEK_SET**（デフォルト）- *pos* がファイル先頭からの位置であることを 指定します。
- **SEEK_CUR** - *pos* が現在のファイル位置からの文字数である ことを指定します。このカウントは正にも負にもなりえます。
- **SEEK_END** - *pos* がファイル終端からの文字数であることを 指定します。負の値は、現在のファイルの範囲内の位置を指定します。 正の値は、現在のファイル終端を越えた位置を指定します。ファイル終端を 超える位置を指定して実際にデータを書き込んだ場合、ファイルはその位置までゼロバイトで埋めて拡張されます。

返り値

例

Example#1 ファイル内の位置の指定

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR);

dio_seek($fd, 10, SEEK_SET);
// ファイル先頭から 10 文字目

dio_seek($fd, -2, SEEK_CUR);
// ファイル先頭から 8 文字目
```

```
dio_seek($fd, -5, SEEK_END);
// ファイル終端から 5 文字戻ったところ

dio_seek($fd, 10, SEEK_END);
// ファイル終端からさらに 10 文字進んだところ。
// ファイル終端と現在の位置の間は ゼロ で埋められる。

dio_close($fd);
?>
```

---

dio_stat

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_stat — ファイル記述子 fd に関する stat 情報を取得する

説明

array **dio_stat** (resource $fd)

dio_stat() は、与えられたファイル記述子に関する 情報を返します。

パラメータ

fd

dio_open() が返すファイル記述子。

返り値

以下のキーを含む連想配列を返します。

- "device" - デバイス
- "inode" - i ノード
- "mode" - モード
- "nlink" - ハードリンク数
- "uid" - ユーザ ID
- "gid" - グループ ID
- "device_type" - デバイス型（inode デバイスの場合）
- "size" - サイズ（バイト数）
- "blocksize" - ブロック長
- "blocks" - 割り当てられたブロック数
- "atime" - 最終アクセス時刻
- "mtime" - 最終更新時刻
- "ctime" - 最終変更時刻

エラー時には **dio_stat()** は **NULL** を返します。

---

dio_tcsetattr

(PHP 4 >= 4.3.0, PHP 5 <= 5.0.5)

dio_tcsetattr — シリアルポートの端末属性とボーレートを設定する

説明

bool **dio_tcsetattr** (resource $fd , array $options)

dio_tcsetattr() は、オープンした *fd* の端末属性とボーレートを指定します。

注意: この関数は Windows 環境にはまだ実装されていません。

パラメータ

fd

dio_open() が返すファイル記述子。

options

現在使用可能なオプションは以下のとおりです。

- 'baud' - ポートのボーレート - とりうる値は 38400,19200,9600,4800,2400,1800, 1200,600,300,200,150,134,110,75 あるいは 50 で、デフォルト値は 9600 。
- 'bits' - データビット - とりうる値は 8,7,6 あるいは 5 で、デフォルト値は 8 。
- 'stop' - ストップビット - とりうる値は 1 あるいは 2 で、デフォルト値は 1 。
- 'parity' - とりうる値は 0,1 あるいは 2 で、デフォルト値は 0 。

返り値

値を返しません。

例

Example#1 シリアルポートのボーレートを設定する

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR | O_NOCTTY | O_NONBLOCK);

dio_fcntl($fd, F_SETFL, O_SYNC);

dio_tcsetattr($fd, array(
  'baud'  => 9600,
  'bits'  => 8,
  'stop'  => 1,
  'parity'  => 0
));

while (1) {

  $data = dio_read($fd, 256);

  if ($data) {
      echo $data;
  }
}

?>
```

---

dio_truncate

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_truncate — ファイル記述子 fd をオフセットバイトへ丸める

説明

bool **dio_truncate** (resource `$fd` , int `$offset`)

dio_truncate() は、ファイルの大きさを 最大 *offset* バイトまでに丸めます。

ファイルがこのサイズより大きかった場合は、残りのデータは失われます。 ファイルがこのサイズより小さかった場合は、ファイルがそのままになるか拡張されるかは未定義です。後者の場合、拡張された部分のデータはゼロと なります。

注意: この関数は Windows 環境にはまだ実装されていません。

パラメータ

fd

dio_open() が返すファイル記述子。

offset

オフセットのバイト数。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

dio_write

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

dio_write — オプションで丸め長さを指定してデータを書き込む

説明

int **dio_write** (resource $fd , string $data [, int $len])

dio_write() は、*data* から最大 *len* バイトを ファイル *fd* に書き込みます。

パラメータ

fd

dio_open() が返すファイル記述子。

data

書き込むデータ。

len

書き込むデータのバイト長。指定しなかった場合は、データ全体を 指定したファイルに書き込みます。

返り値

fd に書き込んだバイト数を返します。

参考

- dio_read()

目次

ディレクトリ関数

導入

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

DIRECTORY_SEPARATOR (string)
PATH_SEPARATOR (string)

> **注意**: **PATH_SEPARATOR** は PHP 4.3.0-RC2 で導入されました。

参考

dirname(), is_dir(), mkdir(), rmdir()等の関数 については、ファイルシステム の節を参照ください。

---

chdir

(PHP 4, PHP 5)

chdir — ディレクトリを変更する

説明

bool **chdir** (string $directory)

PHP のカレントディレクトリを *directory* に変更します。

パラメータ

directory

> 新しいカレントディレクトリ

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 chdir() の例

```
<?php

// カレントディレクトリ
echo getcwd() . "¥n";

chdir('public_html');

// カレントディレクトリ
echo getcwd() . "¥n";

?>
```

上の例の出力は、たとえば以下のようになります。

```
/home/vincent
/home/vincent/public_html
```

注意

> **注意**: セーフモード が有効の場合、PHP は、操作を行うディレクトリが、実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- getcwd()

---

chroot

(PHP 4 >= 4.0.5, PHP 5)

chroot — ルートディレクトリを変更する

説明

bool **chroot** (string $directory)

カレントのプロセスのルートディレクトリを *directory* に変更します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

> **注意**: 使用しているシステムがサポートし、 かつCLI,CGI,Embed SAPIを使用している場合でのみ この関数は有効です。
>
> **注意**: この関数は Windows 環境にはまだ実装されていません。

---

dir

(PHP 4, PHP 5)

dir — ディレクトリクラスのインスタンスを返す

説明

Directory
Directory (string $directory)
string*$path* ;
resource*$handle* ;
string **read** (void)
void **rewind** (void)
void **close** (void)

ディレクトリを読むための疑似オブジェクト指向の機構です。 指定した *directory* がオープンされます。 いったんディレクトリがオープンされると、2 つのプロパティを使用 できるようになります。handle プロパティは、**readdir ()**, rewinddir(), **closedir()** のような他のディレクトリ関数と組み合わせて 使用可能です。path プロパティには、オープンするディレクトリのパス がセットされます。read, rewind, close の 3 種類のメソッドが 使用できます。

例

Example#1 dir() の例

以下の例で、**dir::read()** の返り値をどのように調べているかに 注目してください。返り値が **FALSE** と一致することを、明示的に （値が等しく、かつ型も等しい - 詳細は 比較演算子 を参照ください）調べています。なぜなら、そうしないと **FALSE** と評価されてしまうディレクトリエントリがあった場合にループが とまってしまうからです。

```
<?php
$d = dir("/etc/php5");
echo "Handle: " . $d->handle . "¥n";
echo "Path: " . $d->path . "¥n";
while (false !== ($entry = $d->read())) {
    echo $entry."¥n";
}
$d->close();
?>
```

上の例の出力は、たとえば以下のようになります。

```
Handle: Resource id #2
Path: /etc/php5
.
..
apache
cgi
cli
```

注意

注意: 読み込みにより返されるディレクトリエントリの順番は、システムに依存します。

---

closedir

(PHP 4, PHP 5)

closedir — ディレクトリハンドルをクローズする

説明

void **closedir** (resource `$dir_handle`)

dir_handle で指定したディレクトリのストリームをクローズします。このストリームは、 opendir()により事前にオープンされていなければ なりません。

パラメータ

dir_handle

opendir()により事前にオープンされた ディレクトリハンドルリソース

例

Example#1 closedir() の例

```
<?php
$dir = "/etc/php5/";

// 既知のディレクトリをオープンし、変数にディレクトリを読み込んで閉じる
if (is_dir($dir)) {
    if ($dh = opendir($dir)) {
        $directory = readdir($dh);
        closedir($dh);
    }
}
?>
```

---

getcwd

(PHP 4, PHP 5)

getcwd — カレントのワーキングディレクトリを取得する

説明

string **getcwd** (void)

カレントのワーキングディレクトリを返します。

返り値

成功時はカレントのワーキングディレクトリを返します。 失敗時は **FALSE** を返します。

いくつかの UNIX の派生系では、親ディレクトリが読み込めない、 もしくは検索モードが設定されている場合、カレントディレクトリが可能だとしても **getcwd()** は **FALSE** を返します。 モードと権限についての詳細は、chmod() を参照ください。

例

Example#1 getcwd() の例

```
<?php

// カレントディレクトリ
echo getcwd() . "¥n";

chdir('cvs');

// カレントディレクトリ
echo getcwd() . "¥n";

?>
```

上の例の出力は、たとえば以下のようになります。

```
/home/didou
/home/didou/cvs
```

参考

- chdir()
- chmod()

---

opendir

(PHP 4, PHP 5)

opendir — ディレクトリハンドルをオープンする

説明

resource **opendir** (string $path [, resource $context])

ディレクトリハンドルをオープンします。このハンドルは、この後 closedir(), readdir(), rewinddir() 関数コールで使用されます。

パラメータ

path

オープンするディレクトリのパス。

context

context パラメータの詳細については マニュアルのストリーム を参照ください。

返り値

成功した場合にディレクトリハンドルの resource 、 失敗した場合に **FALSE** を返します。

path が有効なディレクトリでないかまたは権限が 制限されているかファイルシステムのエラーによりディレクトリが オープンできない場合、**opendir()** は **FALSE** を返し、 E_WARNING エラーが発行されます。 **opendir()** のこのエラー出力は、 関数名の前に '@' を付けることにより抑制できます。

変更履歴

バージョン	説明
5.0.0	*path* が *ftp://* URL ラッパをサポートします。
4.3.0	*path* に、ディレクトリの一覧表示をサポートする URL を指定することが可能です。しかし、PHP 4 では *file://* URL ラッパのみをサポートしています。

例

Example#1 opendir() の例

```
<?php
$dir = "/etc/php5/";

// 既知のディレクトリをオープンし、その内容を読み込みます。
if (is_dir($dir)) {
    if ($dh = opendir($dir)) {
        while (($file = readdir($dh)) !== false) {
            echo "filename: $file : filetype: " . filetype($dir . $file) . "¥n";
        }
        closedir($dh);
    }
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
filename: . : filetype: dir
filename: .. : filetype: dir
filename: apache : filetype: dir
filename: cgi : filetype: dir
```

```
filename: cli : filetype: dir
```

参考

- is_dir()
- readdir()
- Dir

---

readdir

(PHP 4, PHP 5)

readdir — ディレクトリハンドルからエントリを読み込む

説明

string **readdir** (resource $dir_handle)

ディレクトリから次のファイルのファイル名を返します。 ファイル名はファイルシステム上に格納されている順番で返されます。

パラメータ

dir_handle

opendir() で事前にオープンした ディレクトリハンドルの resource 。

返り値

成功した場合にファイル名、失敗した場合に **FALSE** を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される *0* や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

例

Example#1 ディレクトリ内の全てのファイルのリストを得る

以下の例で、**readdir()** の返り値をどのように調べているかに 注目してください。返り値が **FALSE** と一致することを、明示的に （値が等しく、かつ型も等しい - 詳細は 比較演算子 を参照ください）調べています。なぜなら、そうしないと **FALSE** と評価されてしまうディレクトリエントリ （例: "0" という名前のディレクトリ）があった場合にループが とまってしまうからです。

```php
<?php
// 4.0.0-RC2 より前のバージョンでは、!== は存在しないことに注意しましょう

if ($handle = opendir('/path/to/files')) {
    echo "Directory handle: $handle¥n";
    echo "Files:¥n";

    /* ディレクトリをループする際の正しい方法です */
    while (false !== ($file = readdir($handle))) {
        echo "$file¥n";
    }

    /* ディレクトリをループする際の「間違った」方法です */
    while ($file = readdir($handle)) {
        echo "$file¥n";
    }

    closedir($handle);
}
?>
```

Example#2 カレントディレクトリの全てのファイルを一覧する。ただし . および .. は取り除く

```php
<?php
if ($handle = opendir('.')) {
    while (false !== ($file = readdir($handle))) {
        if ($file != "." && $file != "..") {
            echo "$file¥n";
        }
    }
    closedir($handle);
}
?>
```

参考

- is_dir()
- glob()

---

rewinddir

(PHP 4, PHP 5)

rewinddir — ディレクトリハンドルを元に戻す

説明

void **rewinddir** (resource $dir_handle)

dir_handle で指定されたディレクトリの ストリームをディレクトリの先頭にリセットします。

パラメータ

dir_handle

opendir()により事前にオープンされた ディレクトリハンドルリソース

---

scandir

(PHP 5)

scandir — 指定されたパスのファイルとディレクトリのリストを取得する

説明

array **scandir** (string $directory [, int $sorting_order [, resource $context]])

directory 内のファイルおよびディレクトリを 配列で返します。

パラメータ

directory

調べるディレクトリ。

sorting_order

デフォルトでは、ソート順はアルファベット昇順です。 オプションの *sorting_order* が使用（1にセット）された 場合、ソート順はアルファベット降順になります。

context

context パラメータの説明については、 マニュアルのストリーム を参照ください。

返り値

成功した場合にファイル名の配列、失敗した場合に **FALSE** を返します。 *directory* がディレクトリではない場合は、 **FALSE** を返し、**E_WARNING** レベルのエラーを 発行します。

例

Example#1 scandir() の簡単な例

```
<?php
$dir    = '/tmp';
$files1 = scandir($dir);
$files2 = scandir($dir, 1);

print_r($files1);
print_r($files2);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => .
     [1] => ..
     [2] => bar.php
     [3] => foo.txt
     [4] => somedir
)
Array
(
     [0] => somedir
     [1] => foo.txt
     [2] => bar.php
     [3] => ..
     [4] => .
)
```

Example#2 PHP 4 での scandir() の代用方法

```
<?php
$dir = "/tmp";
$dh  = opendir($dir);
while (false !== ($filename = readdir($dh))) {
    $files[] = $filename;
}

sort($files);

print_r($files);

rsort($files);

print_r($files);

?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => .
     [1] => ..
     [2] => bar.php
     [3] => foo.txt
     [4] => somedir
)
Array
(
     [0] => somedir
     [1] => foo.txt
     [2] => bar.php
     [3] => ..
     [4] => .
)
```

注意

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

参考

- opendir()
- readdir()
- glob()
- is_dir()
- sort()

目次

- rewinddir — ディレクトリハンドルを元に戻す
- scandir — 指定されたパスのファイルとディレクトリのリストを取得する

DOM 関数

導入

DOM 拡張モジュールを使用すると、DOM API を使用した XML ドキュメントの操作を PHP 5 で行えます。

PHP 4 では DOM XML を使用します。

> **注意**: DOM 拡張モジュールは UTF-8 エンコーディングを使用します。ISO-8859-1 エンコーディングのテキストを扱うには utf8_encode() と utf8_decode() を使用します。またその他のエンコーディングを扱うには Iconv を使用します。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

定義済みクラス

このモジュールの API は、» DOM Level 3 の標準規格と可能な限り 一致させています。そのため、API は完全にオブジェクト指向となっています。このモジュールを使用する際は、DOM 標準規格を意識することが大切です。

このモジュールで定義されている多くのクラスについて以下の表で説明します。 DOM 標準規格に相当するクラスについては DOMxxx という名前がつけられています。

DOMAttr

DOMNode を継承します。DOMAttr インターフェースは、DOMElement オブジェクトの 属性を表します。

コンストラクタ

- DOMAttr->__construct() - 新しい DOMAttr オブジェクトを作成する

メソッド

- DOMAttr->isId() - 属性が定義済みの ID であるかどうかを調べる

プロパティ

名前	型	読み込み専用	説明
name	string	yes	属性の名前
ownerElement	DOMElement	yes	属性を保持する要素
schemaTypeInfo	bool	yes	未実装です。常に **NULL** を返します。
specified	bool	yes	未実装です。常に **NULL** を返します。
value	string	no	属性の値

DOMCharacterData

DOMNode を継承します。

メソッド

- DOMCharacterData->appendData() - ノードの文字データの最後に文字列を追加する
- DOMCharacterData->deleteData() - 指定した範囲の文字列をノードから削除する
- DOMCharacterData->insertData() - 指定した 16 ビット単位のオフセットに、文字列を挿入する
- DOMCharacterData->replaceData() - DOMCharacterData ノードの文字列の一部を置換する

- DOMCharacterData->substringData() - ノードから指定した範囲のデータを抽出する

プロパティ

名前	型	読み込み専用	説明
data	string	no	ノードの内容
length	int	yes	内容の長さ

DOMComment

DOMCharacterData を継承します。

コンストラクタ

- DOMComment->__construct() - 新しい DOMComment オブジェクトを作成する

DOMDocument

DOMNode を継承します。

コンストラクタ

- DOMDocument->__construct() - 新しい DOMDocument オブジェクトを作成する

メソッド

- DOMDocument->createAttribute() - 新しい属性を作成する
- DOMDocument->createAttributeNS() - 関連付けられた名前空間に新しい属性を作成する
- DOMDocument->createCDATASection() - 新しい cdata ノードを作成する
- DOMDocument->createComment() - 新しい comment ノードを作成する
- DOMDocument->createDocumentFragment() - 新しい文書片を作成する
- DOMDocument->createElement() - 新しい要素ノードを作成する
- DOMDocument->createElementNS() - 関連付けられた名前空間に新しい要素ノードを作成する
- DOMDocument->createEntityReference() - 新しいエンティティ参照ノードを作成する
- DOMDocument->createProcessingInstruction() - 新しい PI ノードを作成する
- DOMDocument->createTextNode() - 新しいテキストノードを作成する
- DOMDocument->getElementById() - id に対応する要素を検索する
- DOMDocument->getElementsByTagName() - 指定したタグ名に対応するすべての要素を検索する
- DOMDocument->getElementsByTagNameNS() - 指定した名前空間で、タグ名に対応するすべての要素を検索する
- DOMDocument->importNode() - 現在のドキュメントにノードをインポートする
- DOMDocument->load() - ファイルから XML を読み込む
- DOMDocument->loadHTML() - 文字列から HTML を読み込む
- DOMDocument->loadHTMLFile() - ファイルから HTML を読み込む
- DOMDocument->loadXML() - 文字列から XML を読み込む
- DOMDocument->normalizeDocument() - ドキュメントを正規化する
- DOMDocument->relaxNGValidate() - ドキュメントを relaxNG で検証する
- DOMDocument->relaxNGValidateSource() - ドキュメントを relaxNG で検証する

- DOMDocument->registerNodeClass() - 基底ノード型の作成に使用する拡張クラスを登録する (DOM 標準ではありません)
- DOMDocument->save() - 内部の XML ツリーをファイルに出力する
- DOMDocument->saveHTML() - 内部のドキュメントを HTML 形式の文字列として出力する
- DOMDocument->saveHTMLFile() - 内部のドキュメントを HTML 形式でファイルに出力する
- DOMDocument->saveXML() - 内部の XML ツリーを文字列として出力する
- DOMDocument->schemaValidate() - スキーマに基づいてドキュメントを検証する
- DOMDocument->schemaValidateSource() - スキーマに基づいてドキュメントを検証する
- DOMDocument->validate() - DTD に基づいてドキュメントを検証する
- DOMDocument->xinclude() - DOMDocument オブジェクト内の XIncludes を置換する

プロパティ

名前	型	読み込み専用	説明
actualEncoding	string	yes	
config	DOMConfiguration	yes	
doctype	DOMDocumentType	yes	このドキュメントに関連付けられた文書型宣言
documentElement	DOMElement	yes	ドキュメントの子ノードであるドキュメント要素に対し、 直接アクセスするために便利な属性
documentURI	string	no	ドキュメントの位置。未定義の場合は **NULL**
encoding	string	no	
formatOutput	bool	no	
implementation	DOMImplementation	yes	このドキュメントを処理する DOMImplementation オブジェクト
preserveWhiteSpace	bool	no	余分な空白を取り除かない。デフォルトは **TRUE**
recover	bool	no	
resolveExternals	bool	no	文書型宣言で外部エンティティを読み込む際に **TRUE** を設定する。 XML ドキュメントに文字エンティティを含める際に便利です。
standalone	bool	no	
strictErrorChecking	bool	no	エラー時に DOMException をスローする。デフォルトは **TRUE**
substituteEntities	bool	no	
validateOnParse	bool	no	DTD を読み込んで検証する。デフォルトは **FALSE**
version	string	no	
xmlEncoding	string	yes	XML 宣言の一部として、このドキュメントのエンコーディングを 指定する属性。指定されていない場合や不明な場合 (たとえば ドキュメントがメモリ上に存在する場合など) は **NULL**
xmlStandalone	bool	no	XML 宣言の一部として、このドキュメントがスタンドアローンか どうかを指定する。指定されていない場合は **FALSE**
xmlVersion	string	no	XML 宣言の一部として、このドキュメントのバージョン番号を指定する。 バージョン番号が定義されておらず、ドキュメントが "XML" の機能を サポートしている場合は、値は "1.0"

DOMDocumentFragment

DOMNode を継承します。

メソッド

- DOMDocumentFragment->appendXML() - 生の XML データを追加する (DOM 標準ではありません)

DOMDocumentType

DOMNode を継承します。

各 DOMDocument は *doctype* 属性を保持しており、その値は **NULL** あるいは DOMDocumentType オブジェクトです。

プロパティ

名前	型	読み込み専用	説明
publicId	string	yes	外部サブセットの公開 ID
systemId	string	yes	外部サブセットのシステム ID。完全 URI である場合とそうでない場合がある。
name	string	yes	DTD の名前。すなわち、*DOCTYPE* キーワードに続く値
entities	DOMNamedNodeMap	yes	DTD で宣言されている一般エンティティ (外部・内部とも) を含む DOMNamedNodeMap
notations	DOMNamedNodeMap	yes	DTD で宣言されている記法を含む DOMNamedNodeMap
internalSubset	string	yes	内部サブセットを文字列として取得する。存在しない場合は null となる。区切りの角括弧は含まない。

DOMElement

DOMNode を継承します。

コンストラクタ

- DOMElement->__construct() - 新しい DOMElement オブジェクトを作成する

メソッド

- DOMElement->getAttribute() - 属性の値を返す
- DOMElement->getAttributeNode() - 属性ノードを返す
- DOMElement->getAttributeNodeNS() - 属性ノードを返す
- DOMElement->getAttributeNS() - 属性の値を返す
- DOMElement->getElementsByTagName() - タグ名から要素を取得する
- DOMElement->getElementsByTagNameNS() - 名前空間 URI とローカル名から要素を取得する
- DOMElement->hasAttribute() - 属性が存在するかどうかを調べる
- DOMElement->hasAttributeNS() - 属性が存在するかどうかを調べる
- DOMElement->removeAttribute() - 属性を削除する
- DOMElement->removeAttributeNode() - 属性を削除する
- DOMElement->removeAttributeNS() - 属性を削除する
- DOMElement->setAttribute() - 新しい属性を追加する
- DOMElement->setAttributeNode() - 新しい属性ノードを要素に追加する
- DOMElement->setAttributeNodeNS() - 新しい属性ノードを要素に追加する
- DOMElement->setAttributeNS() - 新しい属性を追加する
- DOMElement->setIdAttribute() - ID 属性を宣言する
- DOMElement->setIdAttributeNode() - ID 属性を宣言する
- DOMElement->setIdAttributeNS() - ID 属性を宣言する

プロパティ

名前	型	読み込み専用	説明
schemaTypeInfo	bool	yes	未実装。常に **NULL** を返す。
tagName	string	yes	要素名

DOMEntity

DOMNode を継承します。

このインターフェースは、XML ドキュメント内の既知のエンティティを表します。 パース済みかどうかは関係ありません。

プロパティ

名前	型	読み込み専用	説明
publicId	string	yes	エンティティに関連付けられているパブリック ID が存在すればその値、 それ以外の場合は **NULL**
systemId	string	yes	エンティティに関連付けられているシステム ID が存在すればその値、 それ以外の場合は **NULL**。これは完全な URI かもしれないし、 そうでないかもしれない。
notationName	string	yes	パースされていないエンティティの場合はそのエンティティの名前、 パース済みのエンティティの場合は **NULL**
actualEncoding	string	no	外部でパースされたエンティティの場合は、このエンティティの パース時に使用されたエンコーディングを指定する属性。内部 サブセットからのエンティティであったり未知のエンティティで あった場合は **NULL**
encoding	string	yes	外部でパースされたエンティティの場合は、テキスト宣言の一部として このエンティティのエンコーディングを指定する属性。それ以外の 場合は **NULL**
version	string	yes	外部でパースされたエンティティの場合は、テキスト宣言の一部として このエンティティのバージョン番号を指定する属性。それ以外の場合は **NULL**

DOMEntityReference

DOMNode を継承します。

コンストラクタ

- DOMEntityReference->__construct() - 新しい DOMEntityReference オブジェクトを作成する

DOMException

しかるべき状況、すなわち論理的に不可能な操作を行った際などの場合に DOM 操作は例外を発生させます。

例外(exceptions) も参照ください。

プロパティ

名前	型	読み込み専用	説明
code	int	yes	発生したエラーの型を示す整数値

DOMImplementation

DOMImplementation インターフェースは、個々の ドキュメントオブジェクトモデルのインスタンス独自の操作を行うための メソッド群を提供します。

コンストラクタ

- DOMImplementation->__construct() - 新しい DOMImplementation オブジェクトを作成する

メソッド

- DOMImplementation->createDocument() - 指定した型とドキュメント要素の DOMDocument オブジェクトを作成する
- DOMImplementation->createDocumentType() - 空の DOMDocumentType オブジェクトを作成する
- DOMImplementation->hasFeature() - DOM 実装が、特定の機能を実装しているかどうかを調べる

DOMNamedNodeMap

メソッド

- DOMNamedNodeMap->getNamedItem() - 名前からノードを取得する
- DOMNamedNodeMap->getNamedItemNS() - ローカル名と名前空間 URI からノードを取得する
- DOMNamedNodeMap->item() - インデックスからノードを取得する

DOMNode

メソッド

- DOMNode->appendChild() - 子要素群の最後に新しい子要素を追加する
- DOMNode->cloneNode() - ノードを複製する
- DOMNode->hasAttributes() - ノードが属性を保持しているかどうかを確かめる
- DOMNode->hasChildNodes() - ノードが子を保持しているかどうかを調べる
- DOMNode->insertBefore() - 参照しているノードの前に 新しい子を追加する
- DOMNode->isDefaultNamespace() - 指定した名前空間 URI がデフォルトの名前空間かどうかを調べる
- DOMNode->isSameNode() - 2 つのノードが等しいかどうかを調べる
- DOMNode->isSupported() - 指定したバージョンで機能がサポートされているかどうかを調べる
- DOMNode->lookupNamespaceURI() - プレフィックスに基づいて、ノードの名前空間 URI を返す
- DOMNode->lookupPrefix() - 名前空間 URI に基づいて、ノードの名前空間プレフィックスを返す
- DOMNode->normalize() - ノードを正規化する
- DOMNode->removeChild() - 子要素群から子を削除する
- DOMNode->replaceChild() - 子を置き換える

プロパティ

名前	型	読み込み専用	説明
nodeName	string	yes	現在のノード型の正確な名前を返す
nodeValue	string	no	その型に応じてノードの値を返す
nodeType	int	yes	ノードの型を、定義済みの定数 XML_xxx_NODE のいずれかで返す
parentNode	DOMNode	yes	このノードの親を返す
childNodes	DOMNodeList	yes	このノードのすべての子を含む DOMNodeList。 子が存在しない場合は、空の DOMNodeList
firstChild	DOMNode	yes	このノードの最初の子。存在しない場合は **NULL** を返す
lastChild	DOMNode	yes	このノードの最後の子。存在しない場合は **NULL** を返す
previousSibling	DOMNode	yes	このノードの直前のノード。存在しない場合は **NULL** を返す
nextSibling	DOMNode	yes	このノードの直後のノード。存在しない場合は **NULL** を返す
attributes	DOMNamedNodeMap	yes	このノードが DOMElement の場合は ノードの属性を含む DOMNamedNodeMap、 それ以外の場合は **NULL**
ownerDocument	DOMDocument	yes	このノードに関連付けられている DOMDocument オブジェクト
namespaceURI	string	yes	このノードの名前空間 URI。指定されていない場合は **NULL**
prefix	string	no	このノードの名前空間プレフィックス。指定されていない場合は **NULL**
localName	string	yes	このノードの名前のローカル部分を返す
baseURI	string	yes	このノードの完全なベース URI。もし実装が完全な URL を できなかった場合は **NULL**
textContent	string	no	このノードとその子孫ノードのテキストを返す

DOMNodeList

メソッド

- DOMNodelist->item() - インデックスで指定したノードを取得する

プロパティ

名前	型	読み込み専用	説明
length	int	yes	リスト内のノードの数。有効な子ノードのインデックスの範囲は 0 以上 *length - 1* 以下

DOMNotation

DOMNode を継承します。

プロパティ

名前	型	読み込み専用	説明
publicId	string	yes	
systemId	string	yes	

DOMProcessingInstruction

DOMNode を継承します。

コンストラクタ

- DOMProcessingInstruction->__construct() - 新しい DOMProcessingInstruction オブジェクトを作成する

プロパティ

名前	型	読み込み専用	説明
target	string	yes	
data	string	no	

DOMText

DOMCharacterData を継承します。

コンストラクタ

- DOMText->__construct() - 新しい DOMText オブジェクトを作成する

メソッド

- DOMText->isWhitespaceInElementContent() - このテキストノードが空白を含むかどうかを示す
- DOMText->splitText() - 指定したオフセットでノードを 2 つに分割する

プロパティ

名前	型	読み込み専用	説明
wholeText	string	yes	

DOMXPath

コンストラクタ

- DOMXPath->__construct() - 新しい DOMXPath オブジェクトを作成する

メソッド

- DOMXPath->registerNamespace() - DOMXpath オブジェクトの名前空間を登録する
- DOMXPath->evaluate() - XPath 式を評価し、結果を返す
- DOMXPath->query() - XPath 式を評価する

プロパティ

名前	型	読み込み専用	説明
document	DOMDocument		

例

このリファレンスの多くの例では XML ファイルを使用します。その際には、 以下のような形式の *book.xml* を使用します。

Example#1 book.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd" [
]>
<book id="listing">
  <title>My lists</title>
  <chapter id="books">
    <title>My books</title>
   <para>
     <informaltable>
      <tgroup cols="4">
       <thead>
        <row>
           <entry>Title</entry>
           <entry>Author</entry>
           <entry>Language</entry>
           <entry>ISBN</entry>
        </row>
       </thead>
       <tbody>
        <row>
           <entry>The Grapes of Wrath</entry>
           <entry>John Steinbeck</entry>
          <entry>en</entry>
           <entry>0140186409</entry>
       </row>
       <row>
           <entry>The Pearl</entry>
           <entry>John Steinbeck</entry>
          <entry>en</entry>
           <entry>014017737X</entry>
       </row>
       <row>
           <entry>Samarcande</entry>
           <entry>Amine Maalouf</entry>
          <entry>fr</entry>
           <entry>2253051209</entry>
       </row>
        <!-- TODO: I have a lot of remaining books to add.. -->
       </tbody>
      </tgroup>
     </informaltable>
   </para>
  </chapter>
</book>
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

XML 定数

定数	値	説明
XML_ELEMENT_NODE (integer)	1	ノードは DOMElement です。
XML_ATTRIBUTE_NODE (integer)	2	ノードは DOMAttr です。
XML_TEXT_NODE (integer)	3	ノードは DOMText です。
XML_CDATA_SECTION_NODE (integer)	4	ノードは DOMCharacterData です。
XML_ENTITY_REF_NODE (integer)	5	ノードは DOMEntityReference です。
XML_ENTITY_NODE (integer)	6	ノードは DOMEntity です。
XML_PI_NODE (integer)	7	ノードは DOMProcessingInstruction です。
XML_COMMENT_NODE (integer)	8	ノードは DOMComment です。
XML_DOCUMENT_NODE (integer)	9	ノードは DOMDocument です。
XML_DOCUMENT_TYPE_NODE (integer)	10	ノードは DOMDocumentType です。
XML_DOCUMENT_FRAG_NODE (integer)	11	ノードは DOMDocumentFragment です。
XML_NOTATION_NODE (integer)	12	ノードは DOMNotation です。
XML_HTML_DOCUMENT_NODE (integer)	13	
XML_DTD_NODE (integer)	14	
XML_ELEMENT_DECL_NODE (integer)	15	
XML_ATTRIBUTE_DECL_NODE (integer)	16	
XML_ENTITY_DECL_NODE (integer)	17	
XML_NAMESPACE_DECL_NODE (integer)	18	
XML_ATTRIBUTE_CDATA (integer)	1	
XML_ATTRIBUTE_ID (integer)	2	
XML_ATTRIBUTE_IDREF (integer)	3	
XML_ATTRIBUTE_IDREFS (integer)	4	
XML_ATTRIBUTE_ENTITY (integer)	5	
XML_ATTRIBUTE_NMTOKEN (integer)	7	
XML_ATTRIBUTE_NMTOKENS (integer)	8	
XML_ATTRIBUTE_ENUMERATION (integer)	9	
XML_ATTRIBUTE_NOTATION (integer)	10	

DOMException 定数

定数	値	説明
DOM_INDEX_SIZE_ERR (integer)	1	インデックスあるいはサイズが負です。または上限を超えています。
DOMSTRING_SIZE_ERR (integer)	2	指定したテキストは DOMString 内に収まりません。
DOM_HIERARCHY_REQUEST_ERR (integer)	3	そのノードが所属できない場所に挿入されました。
DOM_WRONG_DOCUMENT_ERR (integer)	4	ノードが、もともと作成されたのと別のドキュメントで使用されました。
DOM_INVALID_CHARACTER_ERR (integer)	5	名前などで、不正な文字が指定されました。
DOM_NO_DATA_ALLOWED_ERR (integer)	6	データをサポートしていないノードでデータが指定されました。
DOM_NO_MODIFICATION_ALLOWED_ERR (integer)	7	変更が許可されていないオブジェクトを変更しようとしました。
DOM_NOT_FOUND_ERR (integer)	8	存在しないノードを参照しようとしました。
DOM_NOT_SUPPORTED_ERR (integer)	9	指定した型のオブジェクトや操作は、この実装ではサポートしていません。
DOM_INUSE_ATTRIBUTE_ERR (integer)	10	別の場所で使用中の属性を追加しようとしました。
DOM_INVALID_STATE_ERR (integer)	11	現在使用できない、あるいは使用できなくなったオブジェクトを 使用しようとしました。
DOM_SYNTAX_ERR (integer)	12	不正な文字列が指定されました。
DOM_INVALID_MODIFICATION_ERR (integer)	13	基底オブジェクトの型を変更しようとしました。
DOM_NAMESPACE_ERR (integer)	14	名前空間に存在しないオブジェクトを作成または変更しようとしました。
DOM_INVALID_ACCESS_ERR (integer)	15	パラメータや操作は基底オブジェクトではサポートされていません。
DOM_VALIDATION_ERR (integer)	16	insertBefore や removeChild のようなメソッドのコールによってノードの 「部分的な妥当性」が満たされなくなった際にこの例外が発生し、操作は 行われません。

DOMAttr->__construct()

(PHP 5)

DOMAttr->__construct() — 新しい DOMAttr オブジェクトを作成する

説明

DOMAttr
__construct (string $name [, string $value])

新しい DOMAttr オブジェクトを作成します。 このオブジェクトは読み込み専用です。このオブジェクトはドキュメントに 追加することができますが、さらに追加のノードを付け加えるためには ノードをドキュメントに関連付ける必要があります。書き込み可能な ノードを作成するには、DOMDocument->createAttribute() を使用します。

パラメータ

name

属性のタグ名。

value

属性の値。

例

Example#1 新しい DOMAttr を作成する

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');
$element = $dom->appendChild(new DOMElement('root'));
$attr = $element->setAttributeNode(new DOMAttr('attr', 'attrvalue'));
echo $dom->saveXML(); /* <?xml version="1.0" encoding="iso-8859-1"?><root attr="attrvalue" /> */

?>
```

参考

- DOMDocument->createAttribute()

DOMAttr->isId()

(No version information available, might be only in CVS)

DOMAttr->isId() — 属性が定義済みの ID かどうかを調べる

説明

DOMAttr
bool **isId** (void)

この関数は、属性が定義済みの ID かどうかを調べます。

DOM 標準規格によると、DTD では ID 型の属性 ID が定義されていなければ なりません。この関数を使用する前には、 DOMDocument->validate() あるいは *DOMDocument::validateOnParse* を使用して ドキュメントを検証する必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 DOMAttr->isId() の例

```
<?php

$doc = new DomDocument;

// id を参照する前に、ドキュメントを検証しなければなりません。
$doc->validateOnParse = true;
$doc->Load('book.xml');
```

```
// chapter 要素から、id という名前の属性を取得します。
$attr = $doc->getElementsByTagName('chapter')->item(0)->getAttributeNode('id');

var_dump($attr->isId()); // bool(true)

?>
```

DOMCharacterData->appendData()

(No version information available, might be only in CVS)

DOMCharacterData->appendData() — ノードの文字データの最後に文字列を追加する

説明

DOMCharacterData
void **appendData** (string $data)

ノードの文字データの最後に文字列 *data* を追加します。

パラメータ

data

追加する文字列。

返り値

値を返しません。

参考

- DOMCharacterData->deleteData()
- DOMCharacterData->insertData()
- DOMCharacterData->replaceData()
- DOMCharacterData->substringData()

DOMCharacterData->deleteData()

(No version information available, might be only in CVS)

DOMCharacterData->deleteData() — 指定した範囲の文字列をノードから削除する

説明

DOMCharacterData
void **deleteData** (int $offset , int $count)

offset で指定した位置から *count* 文字ぶんの文字を削除します。

パラメータ

offset

削除開始位置のオフセット。

count

削除する文字の数。 *offset* と *count* の和が文字列の長さをこえている場合、データ末尾までのすべての データが削除されます。

返り値

値を返しません。

エラー / 例外

DOM_INDEX_SIZE_ERR

offset が負、あるいは 16 ビット単位の データ長より大きい場合、または *count* が 負の場合に発生します。

参考

- DOMCharacterData->appendData()
- DOMCharacterData->insertData()
- DOMCharacterData->replaceData()
- DOMCharacterData->substringData()

DOMCharacterData->insertData()

(No version information available, might be only in CVS)

DOMCharacterData->insertData() — 指定した 16 ビット単位のオフセットに、文字列を挿入する

説明

DOMCharacterData
void **insertData** (int `$offset` , string `$data`)

offset の位置に、文字列 *data* を挿入します。

パラメータ

offset

挿入する場所の文字オフセット。

data

挿入する文字列。

返り値

値を返しません。

エラー / 例外

DOM_INDEX_SIZE_ERR

offset が負であるか、あるいは 16 ビット単位の データ長より大きい場合に発生します。

参考

- DOMCharacterData->appendData()
- DOMCharacterData->deleteData()
- DOMCharacterData->replaceData()
- DOMCharacterData->substringData()

DOMCharacterData->replaceData()

(No version information available, might be only in CVS)

DOMCharacterData->replaceData() — DOMCharacterData ノードの文字列の一部を置換する

説明

DOMCharacterData
void **replaceData** (int `$offset` , int `$count` , string `$data`)

位置 *offset* から *count* 文字ぶんのデータを、*data* に置換します。

パラメータ

offset

置換開始位置のオフセット。

count

置換する文字の数。 *offset* と *count* の和が文字列の長さをこえている場合、データ末尾までのすべての データが置換されます。

data

置換する文字列。

返り値

値を返しません。

エラー / 例外

DOM_INDEX_SIZE_ERR

offset が負、あるいは 16 ビット単位の データ長より大きい場合、または *count* が 負の場合に発生します。

参考

- DOMCharacterData->appendData()
- DOMCharacterData->deleteData()
- DOMCharacterData->insertData()
- DOMCharacterData->substringData()

DOMCharacterData->substringData()

(No version information available, might be only in CVS)

DOMCharacterData->substringData() — ノードから指定した範囲のデータを抽出する

説明

DOMCharacterData
string **substringData** (int $offset , int $count)

指定した部分文字列を返します。

パラメータ

offset

部分文字列の抽出開始位置のオフセット。

count

抽出する文字数。

返り値

指定された部分文字列を返します。 *offset* と *count* の和が文字列の長さをこえている場合、データの末尾までのすべての 16 ビット単位が返されます。

エラー / 例外

DOM_INDEX_SIZE_ERR

offset が負、あるいは 16 ビット単位の データ長より大きい場合、または *count* が 負の場合に発生します。

参考

- DOMCharacterData->appendData()
- DOMCharacterData->deleteData()
- DOMCharacterData->insertData()
- DOMCharacterData->replaceData()

DOMComment->__construct()

(PHP 5)

DOMComment->__construct() — 新しい DOMComment オブジェクトを作成する

説明

DOMComment
__construct ([string $value])

新しい DOMComment オブジェクトを作成します。 このオブジェクトは読み込み専用です。このオブジェクトをドキュメントに 追加することが可能ですが、ノードがドキュメントと関連付けられるまでは ノードを追加することはできません。書き込み可能なノードを作成するには、DOMDocument->createComment() を使用します。

パラメータ

value

コメントの値。

例

Example#1 新しい DOMComment を作成する

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');
$element = $dom->appendChild(new DOMElement('root'));
$comment = $element->appendChild(new DOMComment('root comment'));
echo $dom->saveXML(); /* <?xml version="1.0" encoding="iso-8859-1"?><root><!--root comment--></root> */

?>
```

参考

- DOMDocument->createComment()

DOMDocument->__construct()

(PHP 5)

DOMDocument->__construct() — 新しい DOMDocument オブジェクトを作成する

説明

DOMDocument
__construct ([string $version [, string $encoding]])

新しい DOMDocument オブジェクトを作成します。

パラメータ

version

XML 宣言の一部である、ドキュメントのバージョン番号。

encoding

XML 宣言の一部である、ドキュメントのエンコーディング。

例

Example#1 新しい DOMDocument を作成する

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');

echo $dom->saveXML(); /* <?xml version="1.0" encoding="iso-8859-1"?> */

?>
```

参考

- DOMImplementation->createDocument()

DOMDocument->createAttribute()

(No version information available, might be only in CVS)

DOMDocument->createAttribute() — 新しい属性を作成する

説明

DOMDocument
DOMAttr **createAttribute** (string `$name`)

この関数は、DOMAttr クラスの新しいインスタンスを 作成します。 このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

name

属性の名前。

返り値

新しい DOMAttr、あるいはエラーが発生した場合は **FALSE** を返します。

エラー / 例外

DOM_INVALID_CHARACTER_ERR

name が無効な文字を含んでいる場合に発生します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttributeNS()
- DOMDocument->createCDATASection()
- DOMDocument->createComment()
- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()
- DOMDocument->createEntityReference()
- DOMDocument->createProcessingInstruction()
- DOMDocument->createTextNode()

DOMDocument->createAttributeNS()

(No version information available, might be only in CVS)

DOMDocument->createAttributeNS() — 関連付けられた名前空間に新しい属性を作成する

説明

DOMDocument
DOMAttr **createAttributeNS** (string `$namespaceURI` , string `$qualifiedName`)

この関数は、DOMAttr クラスの新しいインスタンスを 作成します。このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

namespaceURI

名前空間の URI。

qualifiedName

属性のタグ名とプレフィックスを、*prefix:tagname* のような形式で指定する。

返り値

新しい DOMAttr、あるいはエラーが発生した場合は **FALSE** を返します。

エラー / 例外

DOM_INVALID_CHARACTER_ERR

qualifiedName が無効な文字を含んでいる場合に発生します。

DOM_NAMESPACE_ERR

qualifiedName が不正な形式である場合、あるいは *qualifiedName* がプレフィックスを含んでいる にもかかわらず *namespaceURI* が **NULL** である場合に発生します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createCDATASection()
- DOMDocument->createComment()
- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()
- DOMDocument->createEntityReference()
- DOMDocument->createProcessingInstruction()
- DOMDocument->createTextNode()

DOMDocument->createCDATASection()

(No version information available, might be only in CVS)

DOMDocument->createCDATASection() — 新しい cdata ノードを作成する

説明

DOMDocument
DOMCDATASection **createCDATASection** (string `$data`)

この関数は、DOMCDATASection クラスの 新しいインスタンスを作成します。このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

data

cdata の内容。

返り値

新しい DOMCDATASection、あるいはエラーが発生した場合は **FALSE** を返します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()
- DOMDocument->createComment()
- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()

- DOMDocument->createEntityReference()
- DOMDocument->createProcessingInstruction()
- DOMDocument->createTextNode()

DOMDocument->createComment()

(No version information available, might be only in CVS)

DOMDocument->createComment() — 新しい comment ノードを作成する

説明

DOMDocument
DOMComment **createComment** (string `$data`)

この関数は、DOMComment クラスの 新しいインスタンスを作成します。このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

`data`

コメントの内容。

返り値

新しい DOMComment、あるいはエラーが発生した場合は **`FALSE`** を返します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()
- DOMDocument->createCDATASection()
- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()
- DOMDocument->createEntityReference()
- DOMDocument->createProcessingInstruction()
- DOMDocument->createTextNode()

DOMDocument->createDocumentFragment()

(No version information available, might be only in CVS)

DOMDocument->createDocumentFragment() — 新しい文書片を作成する

説明

DOMDocument
DOMDocumentFragment **createDocumentFragment** (void)

この関数は、DOMDocumentFragment クラスの新しいインスタンスを作成します。このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

返り値

新しい DOMDocumentFragment、あるいはエラーが発生した場合は **`FALSE`** を返します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()

- DOMDocument->createCDATASection()
- DOMDocument->createComment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()
- DOMDocument->createEntityReference()
- DOMDocument->createProcessingInstruction()
- DOMDocument->createTextNode()

DOMDocument->createElement()

(No version information available, might be only in CVS)

DOMDocument->createElement() — 新しい要素ノードを作成する

説明

DOMDocument
DOMElement **createElement** (string `$name` [, string `$value`])

この関数は、DOMElement クラスの新しいインスタンスを作成します。 このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

name

要素のタグ名。

value

要素の値。デフォルトでは、空の要素が作成されます。 その後に *DOMElement->nodeValue* で 値を設定することも可能です。

返り値

新しい DOMElement クラスの新しいインスタンス、 あるいはエラーが発生した場合は **FALSE** を返します。

エラー / 例外

DOM_INVALID_CHARACTER_ERR

name が無効な文字を含んでいる場合に発生します。

例

Example#1 新しい要素を作成し、ルートとして挿入する

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');

$element = $dom->createElement('test', 'This is the root element!');

// 新しい要素をルート（ドキュメントの子要素）として挿入する
$dom->appendChild($element);

echo $dom->saveXML();
?>
```

上の例の出力は以下となります。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<test>This is the root element!</test>
```

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()
- DOMDocument->createCDATASection()
- DOMDocument->createComment()

DOMDocument->createElementNS()

(No version information available, might be only in CVS)

DOMDocument->createElementNS() — 関連付けられた名前空間に新しい要素を作成する

説明

DOMDocument
DOMElement **createElementNS** (string `$namespaceURI` , string `$qualifiedName` [, string `$value`])

この関数は、関連付けられた名前空間に新しい要素を作成します。 このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

`namespaceURI`

名前空間の URI。

`qualifiedName`

要素名を、*prefix:tagname* のような形式で指定する。

`value`

要素の値。デフォルトでは、空の要素が作成されます。 その後に *DOMElement->nodeValue* で 値を設定することも可能です。

返り値

新しい DOMElement、 あるいはエラーが発生した場合には **`FALSE`** を返します。

エラー / 例外

`DOM_INVALID_CHARACTER_ERR`

`qualifiedName` が無効な文字を含んでいる場合に発生します。

`DOM_NAMESPACE_ERR`

`qualifiedName` が無効な名前である場合に発生します。

例

Example#1 新しい要素を作成し、ルートとして挿入する

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');

$element = $dom->createElementNS('http://www.example.com/XFoo', 'xfoo:test', 'This is the root element!');

// 新しい要素をルート（ドキュメントの子要素）として挿入する
$dom->appendChild($element);

echo $dom->saveXML();
?>
```

上の例の出力は以下となります。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xfoo:test xmlns:xfoo="http://www.example.com/XFoo">This is the root element!</xfoo:test>
```

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()
- DOMDocument->createCDATASection()
- DOMDocument->createComment()
- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createEntityReference()
- DOMDocument->createProcessingInstruction()
- DOMDocument->createTextNode()

DOMDocument->createEntityReference()

(No version information available, might be only in CVS)

DOMDocument->createEntityReference() — 新しいエンティティ参照ノードを作成する

説明

DOMDocument
DOMEntityReference **createEntityReference** (string `$name`)

この関数は、DOMEntityReference クラスの新しいインスタンスを作成します。このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

name

エンティティ参照の内容、つまり、エンティティ参照から 先頭の & および末尾の ; を取り除いたもの。

返り値

新しい DOMEntityReference、 あるいはエラーが発生した場合には **FALSE** を返します。

エラー / 例外

DOM_INVALID_CHARACTER_ERR

name が無効な文字を含んでいる場合に発生します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()
- DOMDocument->createCDATASection()
- DOMDocument->createComment()
- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()
- DOMDocument->createProcessingInstruction()
- DOMDocument->createTextNode()

DOMDocument->createProcessingInstruction()

(No version information available, might be only in CVS)

DOMDocument->createProcessingInstruction() — 新しい PI ノードを作成する

説明

DOMDocument
DOMProcessingInstruction **createProcessingInstruction** (string `$target` [, string `$data`])

この関数は、DOMProcessingInstruction クラスの新しいインスタンスを作成します。このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

target

処理命令の対象。

data

処理命令の内容。

返り値

新しい DOMProcessingInstruction、 あるいはエラーが発生した場合には **FALSE** を返します。

エラー / 例外

DOM_INVALID_CHARACTER_ERR

target が無効な文字を含んでいる場合に発生します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()
- DOMDocument->createCDATASection()
- DOMDocument->createComment()
- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()
- DOMDocument->createEntityReference()
- DOMDocument->createTextNode()

DOMDocument->createTextNode()

(No version information available, might be only in CVS)

DOMDocument->createTextNode() — 新しいテキストノードを作成する

説明

DOMDocument
DOMText **createTextNode** (string `$content`)

この関数は、DOMText クラスの新しいインスタンスを作成します。このノードは、DOMNode->appendChild() などで挿入されない限り、ドキュメント内に現われません。

パラメータ

content

テキストの内容。

返り値

新しい DOMText、 あるいがエラーが発生した場合には **FALSE** を返します。

参考

- DOMNode->appendChild()
- DOMDocument->createAttribute()
- DOMDocument->createAttributeNS()
- DOMDocument->createCDATASection()
- DOMDocument->createComment()

- DOMDocument->createDocumentFragment()
- DOMDocument->createElement()
- DOMDocument->createElementNS()
- DOMDocument->createEntityReference()
- DOMDocument->createProcessingInstruction()

DOMDocument->getElementById()

(No version information available, might be only in CVS)

DOMDocument->getElementById() — id に対応する要素を検索する

説明

DOMDocument
DOMElement **getElementById** (string `$elementId`)

この関数は、 DOMDocument->getElementsByTagName() と同じですが、指定した ID から要素を検索する点が違います。

この関数を動作させるには、何らかの ID 属性を DOMElement->setIdAttribute() で設定するか、あるいは DTD で ID 型の属性を定義する必要があります。 後者の場合は、 DOMDocument->validate() あるいは *DOMDocument->validateOnParse* を使用してドキュメントを検証する必要があります。

パラメータ

`elementId`

要素の ID。

返り値

DOMElement、あるいは要素が 見つからなかった場合は **`NULL`** を返します。

例

Example#1 DOMDocument->getElementById() の例

```php
<?php

$doc = new DomDocument;

// ID を参照する前に、ドキュメントを検証する必要があります。
$doc->validateOnParse = true;
$doc->Load('book.xml');

echo "ID が books の要素は " . $doc->getElementById('books')->tagName . " です。\n";

?>
```

上の例の出力は以下となります。

```
ID が books の要素は chapter です。
```

参考

- DOMDocument->getElementsByTagName()

DOMDocument->getElementsByTagName()

(No version information available, might be only in CVS)

DOMDocument->getElementsByTagName() — 指定したタグ名に対応するすべての要素を検索する

説明

DOMDocument
DOMNodeList **getElementsByTagName** (string `$name`)

この関数は、指定したタグ名の要素を含む DOMNodeList クラスの新しいインスタンスを返します。

パラメータ

name

タグ名に一致する名前。* はすべてのタグに 一致します。

返り値

一致するすべての要素を含む、新しい DOMNodeList オブジェクトを返します。

参考

- DOMDocument->getElementsByTagNameNS()

DOMDocument->getElementsByTagNameNS()

(No version information available, might be only in CVS)

DOMDocument->getElementsByTagNameNS() — 指定した名前空間で、タグ名に対応するすべての要素を検索する

説明

DOMDocument
DOMNodeList **getElementsByTagNameNS** (string `$namespaceURI` , string `$localName`)

指定したローカル名および名前空間 URI に一致するすべての要素の DOMNodeList を返します。

パラメータ

namespaceURI

条件に一致する要素の名前空間 URI。 * はすべての名前空間に一致します。

localName

条件に一致する要素のローカル名。 * はすべてのローカル名に一致します。

返り値

一致するすべての要素を含む、新しい DOMNodeList オブジェクトを返します。

例

Example#1 すべての XInclude 要素を取得する

```
<?php

$xml = <<<EOD
<?xml version="1.0" ?>
<chapter xmlns:xi="http://www.w3.org/2001/XInclude">
<title>Books of the other guy..</title>
<para>
  <xi:include href="book.xml">
   <xi:fallback>
     <error>xinclude: book.xml not found</error>
   </xi:fallback>
  </xi:include>
  <include>
   This is another namespace
  </include>
</para>
</chapter>
EOD;
$dom = new DOMDocument;

// 上で定義した XML 文字列を読み込みます
$dom->loadXML($xml);

foreach ($dom->getElementsByTagNameNS('http://www.w3.org/2001/XInclude', '*') as $element) {
    echo 'local name: ', $element->localName, ', prefix: ', $element->prefix, "¥n";
}
?>
```

上の例の出力は以下となります。

```
local name: include, prefix: xi
```

```
local name: fallback, prefix: xi
```

参考

- DOMDocument->getElementsByTagName()

DOMDocument->importNode()

(No version information available, might be only in CVS)

DOMDocument->importNode() — 現在のドキュメントにノードをインポートする

説明

DOMDocument
DOMNode **importNode** (DOMNode `$importedNode` [, bool `$deep`])

この関数は、インポートするノードのコピーを返し、それを現在の ドキュメントに関連付けます。

パラメータ

`importedNode`

インポートするノード。

`deep`

TRUE の場合、このメソッドは *importedNode* 以下のサブツリーも再帰的にインポートします。

返り値

コピーされたノードを返します。

エラー / 例外

ノードがインポートできなかった場合には DOMException がスローされます。

DOMDocument->load()

(No version information available, might be only in CVS)

DOMDocument->load() — ファイルから XML を読み込む

説明

DOMDocument
mixed **load** (string `$filename` [, int `$options`])

XML ドキュメントをファイルから読み込みます。

パラメータ

`filename`

XML ドキュメントへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 静的にコールされた場合には DOMDocument を返しますが、同時に E_STRICT 警告も発生します。

エラー / 例外

空の文字列を *filename* に渡したり中身が空のファイルを指定したりすると、警告が発生します。 この警告は libxml が発するものではないので、libxml のエラー処理関数では処理できません。

例

Example#1 ドキュメントの作成

```
<?php
$doc = new DOMDocument();
$doc->load('book.xml');
echo $doc->saveXML();
?>
```

参考

- DOMDocument->loadXML()
- DOMDocument->save()
- DOMDocument->saveXML()

DOMDocument->loadHTML()

(No version information available, might be only in CVS)

DOMDocument->loadHTML() — 文字列から HTML を読み込む

説明

DOMDocument
bool **loadHTML** (string $source)

この関数は、文字列 *source* に含まれる HTML を パースします。XML を読み込む場合とは異なり、妥当な HTML でなくても 読み込むことができます。この関数をスタティックにコールすると、 読み込んだ内容をもとに DOMDocument オブジェクトを作成します。 読み込み前に DOMDocument のプロパティを 設定する必要がない場合に、スタティックに実行することがあるでしょう。

パラメータ

source

HTML 文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

空の文字列を *source* に渡すと、警告が発生します。 この警告は libxml が発するものではないので、libxml のエラー処理関数では処理できません。

例

Example#1 ドキュメントを作成する

```
<?php
$doc = new DOMDocument();
$doc->loadHTML("<html><body>Test<br></body></html>");
echo $doc->saveHTML();
?>
```

参考

- DOMDocument->loadHTMLFile()
- DOMDocument->saveHTML()
- DOMDocument->saveHTMLFile()

DOMDocument->loadHTMLFile()

(No version information available, might be only in CVS)

DOMDocument->loadHTMLFile() — ファイルから HTML を読み込む

説明

DOMDocument
bool **loadHTMLFile** (string `$filename`)

この関数は、*filename* という名前のファイルから 読み込んだ HTML ドキュメントをパースします。 XML を読み込む場合とは異なり、妥当な HTML でなくても 読み込むことができます。

この関数をスタティックにコールすると、 読み込んだ内容をもとに DOMDocument オブジェクトを作成します。 読み込み前に DOMDocument のプロパティを 設定する必要がない場合に、スタティックに実行することがあるでしょう。

パラメータ

filename

HTML ファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

空の文字列を *filename* に渡したり中身が空のファイルを指定したりすると、警告が発生します。 この警告は libxml が発するものではないので、libxml のエラー処理関数では処理できません。

例

Example#1 ドキュメントを作成する

```
<?php
$doc = new DOMDocument();
$doc->loadHTMLFile("filename.html");
echo $doc->saveHTML();
?>
```

参考

- DOMDocument->loadHTML()
- DOMDocument->saveHTML()
- DOMDocument->saveHTMLFile()

DOMDocument->loadXML()

(No version information available, might be only in CVS)

DOMDocument->loadXML() — 文字列から XML を読み込む

説明

DOMDocument
mixed **loadXML** (string `$source` [, int `$options`])

XML ドキュメントを文字列から読み込みます。

このメソッドをスタティックにコールすると、読み込んだ内容をもとに DOMDocument オブジェクトを作成します。 読み込み前に DOMDocument のプロパティを 設定する必要がない場合に、スタティックに実行することがあるでしょう。

パラメータ

source

XML を含む文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 静的にコールされた場合には、DOMDocument を返します。

エラー / 例外

空の文字列を *source* に渡すと、警告が発生します。 この警告は libxml が発するものではないので、libxml のエラー処理関数では処理できません。

例

Example#1 ドキュメントを作成する

```
<?php
$doc = new DOMDocument();
$doc->loadXML('<root><node/></root>');
echo $doc->saveXML();
?>
```

Example#2 *loadXML* の静的な起動

```
<?php
$doc = DOMDocument::loadXML('<root><node/></root>');
echo $doc->saveXML();
?>
```

参考

- DOMDocument->load()
- DOMDocument->save()
- DOMDocument->saveXML()

DOMDocument->normalizeDocument()

(No version information available, might be only in CVS)

DOMDocument->normalizeDocument() — ドキュメントを正規化する

説明

DOMDocument
void **normalizeDocument** (void)

このメソッドは、まるでドキュメントを一度保存してから読み込みなおしたかのように動作し、 ドキュメントを "正規化された" 形式にします。

返り値

値を返しません。

参考

- » DOM 仕様
- DOMNode->normalize()

DOMDocument->registerNodeClass()

(PHP 5 >= 5.2.0)

DOMDocument->registerNodeClass() — 基底ノード型を作成する際に使用する拡張クラスを登録する

説明

DOMDocument
bool **registerNodeClass** (string `$baseclass` , string `$extendedclass`)

このメソッドにより、独自に拡張した DOM クラスを登録することができます。 これを、後で PHP DOM 拡張モジュールで使用します。

このメソッドは、DOM の標準にはないものです。

パラメータ

`baseclass`

拡張したい DOM クラス。クラス名の一覧は、この章の導入部にあります。

もちろん、DOMDocument を拡張したクラスを登録することはできません。 しかし、拡張したクラスのインスタンスを作成することで、常にドキュメントを開始できます。

extendedclass

拡張したクラスの名前。**NULL** を渡した場合は、 それまでに *baseclass* を拡張して作成したすべてのクラスが削除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
PHP 5.2.2	5.2.2 より前のバージョンでは、同一の *baseclass* を継承した新しいクラスを登録する際には、以前に登録されていた *extendedclass* の登録を解除する必要がありました。

例

Example#1 新しいメソッドを DOMElement に追加し、コードを書きやすくする

```
<?php

class myElement extends DOMElement {
    function appendElement($name) {
       return $this->appendChild(new myElement($name));
    }
}

class myDocument extends DOMDocument {
    function setRoot($name) {
       return $this->appendChild(new myElement($name));
    }
}

$doc = new myDocument();
$doc->registerNodeClass('DOMElement', 'myElement');

// これ以降、他の要素への要素の追加が一回のメソッドコールでできるようになります!
$root = $doc->setRoot('root');
$child = $root->appendElement('child');
$child->setAttribute('foo', 'bar');

echo $doc->saveXML();

?>
```

上の例の出力は以下となります。

```
<?xml version="1.0"?>
<root><child foo="bar"/></root>
```

DOMDocument->relaxNGValidate()

(No version information available, might be only in CVS)

DOMDocument->relaxNGValidate() — ドキュメントを relaxNG で検証する

説明

DOMDocument
bool **relaxNGValidate** (string `$filename`)

指定した RNG スキーマに基づいて、ドキュメントを » relaxNG で検証します。

パラメータ

filename

RNG ファイル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOMDocument->relaxNGValidateSource()
- DOMDocument->schemaValidate()
- DOMDocument->schemaValidateSource()
- DOMDocument->validate()

DOMDocument->relaxNGValidateSource()

(No version information available, might be only in CVS)

DOMDocument->relaxNGValidateSource() — ドキュメントを relaxNG で検証する

説明

DOMDocument
bool **relaxNGValidateSource** (string `$source`)

指定した RNG ソースに基づいて、ドキュメントを » relaxNG で検証します。

パラメータ

`source`

RNG スキーマを含む文字列。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- DOMDocument->relaxNGValidate()
- DOMDocument->schemaValidate()
- DOMDocument->schemaValidateSource()
- DOMDocument->validate()

DOMDocument->save()

(No version information available, might be only in CVS)

DOMDocument->save() — 内部の XML ツリーをファイルに出力する

説明

DOMDocument
mixed **save** (string `$filename` [, int `$options`])

DOM 表現から XML ドキュメントを作成します。この関数は、通常は以下の例のように DOM ドキュメントを新しく作成した後にコールされます。

パラメータ

`filename`

保存された XML ドキュメントへのパス。

`options`

追加のオプション。現在は LIBXML_NOEMPTYTAG のみが サポートされています。

返り値

書き込んだバイト数、あるいはエラーが発生した場合は **`FALSE`** を返します。

変更履歴

バージョン	説明

バージョン	説明
5.1.0	*options* パラメータが追加されました。

例

Example#1 DOM ツリーをファイルに保存する

```
<?php

$doc = new DOMDocument('1.0');
// 出力はきれいに整形したいですね。
$doc->formatOutput = true;

$root = $doc->createElement('book');
$root = $doc->appendChild($root);

$title = $doc->createElement('title');
$title = $root->appendChild($title);

$text = $doc->createTextNode('This is the title');
$text = $title->appendChild($text);

echo 'Wrote: ' . $doc->save("/tmp/test.xml") . ' bytes'; // Wrote: 72 bytes

?>
```

参考

- DOMDocument->saveXML()
- DOMDocument->load()
- DOMDocument->loadXML()

DOMDocument->saveHTML()

(No version information available, might be only in CVS)

DOMDocument->saveHTML() — 内部のドキュメントを HTML 形式の文字列として出力する

説明

DOMDocument
string **saveHTML** (void)

DOM 表現から HTML ドキュメントを作成します。この関数は、通常は以下の例のように DOM ドキュメントを新しく作成した後にコールされます。

返り値

HTML、あるいはエラーが発生した場合は **FALSE** を返します。

例

Example#1 HTML ツリーを文字列に保存する

```
<?php

$doc = new DOMDocument('1.0');

$root = $doc->createElement('html');
$root = $doc->appendChild($root);

$head = $doc->createElement('head');
$head = $root->appendChild($head);

$title = $doc->createElement('title');
$title = $head->appendChild($title);

$text = $doc->createTextNode('This is the title');
$text = $title->appendChild($text);

echo $doc->saveHTML();

?>
```

参考

- DOMDocument->saveHTMLFile()
- DOMDocument->loadHTML()
- DOMDocument->loadHTMLFile()

DOMDocument->saveHTMLFile()

(No version information available, might be only in CVS)

DOMDocument->saveHTMLFile() — 内部のドキュメントを HTML 形式でファイルに出力する

説明

DOMDocument
int **saveHTMLFile** (string `$filename`)

DOM 表現から HTML ドキュメントを作成します。この関数は、通常は以下の例のように DOM ドキュメントを新しく作成した後にコールされます。

パラメータ

`filename`

保存された HTML ドキュメントへのパス。

返り値

書き込んだバイト数、あるいはエラーが発生した場合は **FALSE** を返します。

例

Example#1 HTML ツリーをファイルに保存する

```php
<?php

$doc = new DOMDocument('1.0');
// 出力はきれいに整形したいですね。
$doc->formatOutput = true;

$root = $doc->createElement('html');
$root = $doc->appendChild($root);

$head = $doc->createElement('head');
$head = $root->appendChild($head);

$title = $doc->createElement('title');
$title = $head->appendChild($title);

$text = $doc->createTextNode('This is the title');
$text = $title->appendChild($text);

echo 'Wrote: ' . $doc->saveHTMLFile("/tmp/test.html") . ' bytes'; // Wrote: 129 bytes

?>
```

参考

- DOMDocument->saveHTML()
- DOMDocument->loadHTML()
- DOMDocument->loadHTMLFile()

DOMDocument->saveXML()

(No version information available, might be only in CVS)

DOMDocument->saveXML() — 内部の XML ツリーを文字列として出力する

説明

DOMDocument
string **saveXML** ([DOMNode `$node` [, int `$options`]])

DOM 表現から XML ドキュメントを作成します。この関数は、通常は以下の例のように DOM ドキュメントを新しく作成した後にコールされます。

パラメータ

`node`

ドキュメント全体ではなく、XML 宣言以外の特定のノードだけを 出力したい場合にこのパラメータを使用します。

options

追加のオプション。現在は LIBXML_NOEMPTYTAG のみが サポートされています。

返り値

XML、あるいはエラーが発生した場合は **FALSE** を返します。

エラー / 例外

DOM_WRONG_DOCUMENT_ERR

node が別のドキュメントのものである場合に 発生します。

変更履歴

バージョン	説明
5.1.0	*options* パラメータが追加されました。

例

Example#1 DOM ツリーを文字列に保存する

```
<?php

$doc  = new DOMDocument('1.0');
// 出力はきれいに整形したいですね。
$doc->formatOutput  = true;

$root  = $doc->createElement('book');
$root  = $doc->appendChild($root);

$title  = $doc->createElement('title');
$title  = $root->appendChild($title);

$text = $doc->createTextNode('これはタイトルです');
$text  = $title->appendChild($text);

echo "ドキュメント全体を保存します\n";
echo  $doc->saveXML() . "¥n";

echo "タイトルの部分のみを保存します\n";
echo  $doc->saveXML($title);

?>
```

上の例の出力は以下となります。

```
ドキュメント全体を保存します
<?xml  version="1.0"?>
<book>
   <title>これはタイトルです</title>
</book>

タイトルの部分のみを保存します
<title>これはタイトルです</title>
```

参考

- DOMDocument->save()
- DOMDocument->load()
- DOMDocument->loadXML()

DOMDocument->schemaValidate()

(No version information available, might be only in CVS)

DOMDocument->schemaValidate() — スキーマに基づいてドキュメントを検証する

説明

DOMDocument
bool **schemaValidate** (string `$filename`)

指定されたスキーマファイルに基づいてドキュメントを検証します。

パラメータ

filename

スキーマへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOMDocument->schemaValidateSource()
- DOMDocument->relaxNGValidate()
- DOMDocument->relaxNGValidateSource()
- DOMDocument->validate()

DOMDocument->schemaValidateSource()

(No version information available, might be only in CVS)

DOMDocument->schemaValidateSource() — スキーマに基づいてドキュメントを検証する

説明

DOMDocument
bool **schemaValidateSource** (string $source)

指定した文字列に定義されているスキーマに基づいてドキュメントを検証します。

パラメータ

source

スキーマを含む文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOMDocument->schemaValidate()
- DOMDocument->relaxNGValidate()
- DOMDocument->relaxNGValidateSource()
- DOMDocument->validate()

DOMDocument->validate()

(No version information available, might be only in CVS)

DOMDocument->validate() — DTD に基づいてドキュメントを検証する

説明

DOMDocument
bool **validate** (void)

DTD に基づいてドキュメントを検証します。

DTD による検証を行うには、DOMDocument の *validateOnParse* プロパティを使用することも可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 ドキュメントに DTD が添付されていない場合は、このメソッドは **FALSE** を返します。

例

Example#1 DTD による検証の例

```
<?php
$dom = new DOMDocument;
$dom->Load('book.xml');
if ($dom->validate()) {
    echo "このドキュメントは正常です!\n";
}
?>
```

XML ファイルの読み込み時に検証することも可能です。

```
<?php
$dom = new DOMDocument;
$dom->validateOnParse = true;
$dom->Load('book.xml');
?>
```

参考

- DOMDocument->schemaValidate()
- DOMDocument->schemaValidateSource()
- DOMDocument->relaxNGValidate()
- DOMDocument->relaxNGValidateSource()

DOMDocument->xinclude()

(No version information available, might be only in CVS)

DOMDocument->xinclude() — DOMDocument オブジェクト内の XIncludes を置換する

説明

DOMDocument
int **xinclude** ([int `$options`])

このメソッドは、DOMDocument オブジェクト内の » XIncludes を置換します。

> **注意**: include される XML ファイルに DTD が添付されている場合、libxml2 が 自動的にエンティティを解決するため、このメソッドは予期せぬ結果を 引き起こすことがあります。

パラメータ

options

> libxml のパラメータ。 PHP 5.1.0 および Libxml 2.6.7 以降で使用可能です。

返り値

ドキュメント内の XIncludes の数を返します。

例

Example#1 DOMDocument->xinclude() の例

```
<?php

$xml = <<<EOD
<?xml version="1.0" ?>
<chapter xmlns:xi="http://www.w3.org/2001/XInclude">
 <title>Books of the other guy..</title>
 <para>
  <xi:include href="book.xml">
   <xi:fallback>
    <error>xinclude: book.xml not found</error>
   </xi:fallback>
  </xi:include>
 </para>
</chapter>
EOD;

$dom = new DOMDocument;

// 見た目をきれいにします
$dom->preserveWhiteSpace = false;
$dom->formatOutput = true;
```

```
// 上で定義した XML 文字列を読み込みます
$dom->loadXML($xml);

// xincludes を置換します
$dom->xinclude();

echo $dom->saveXML();

?>
```

上の例の出力は、たとえば以下のようになります。

```
<?xml  version="1.0"?>
<chapter  xmlns:xi="http://www.w3.org/2001/XInclude">
   <title>Books of the other guy..</title>
   <para>
       <row  xml:base="/home/didou/book.xml">
          <entry>The Grapes of Wrath</entry>
           <entry>John Steinbeck</entry>
          <entry>en</entry>
           <entry>0140186409</entry>
       </row>
       <row  xml:base="/home/didou/book.xml">
          <entry>The Pearl</entry>
           <entry>John Steinbeck</entry>
          <entry>en</entry>
           <entry>014017737X</entry>
       </row>
       <row  xml:base="/home/didou/book.xml">
           <entry>Samarcande</entry>
           <entry>Amine Maalouf</entry>
          <entry>fr</entry>
           <entry>2253051209</entry>
       </row>
   </para>
</chapter>
```

DOMDocumentFragment->appendXML()

(PHP 5 >= 5.1.0)

DOMDocumentFragment->appendXML() — 生の XML データを追加する

説明

DOMDocumentFragment
bool **appendXML** (string `$data`)

生の XML データを DOMDocumentFragment に追加します。

このメソッドは、DOM の標準にはないものです。 これは、XML DocumentFragment を DOMDocument に簡単に追加できるように作成されました。

標準に従いたい場合は、まずテンポラリの DOMDocument をダミーのルートで作成し、 追加したい XML データのルートの子ノードを順にループする必要があります。

パラメータ

data

追加する XML。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 XML データのドキュメントへの追加

```
<?php
$doc = new DOMDocument();
$doc->loadXML("<root/>");
$f = $doc->createDocumentFragment();
$f->appendXML("<foo>text</foo><bar>text2</bar>");
$doc->documentElement->appendChild($f);
echo $doc->saveXML();
?>
```

上の例の出力は以下となります。

```
<?xml  version="1.0"?>
<root><foo>text</foo><bar>text2</bar></root>
```

DOMElement->__construct()

(PHP 5)

DOMElement->__construct() — 新しい DOMElement オブジェクトを作成する

説明

DOMElement
__construct (string $name [, string $value [, string $namespaceURI]])

新しい DOMElement オブジェクトを作成します。 このオブジェクトは読み込み専用です。このオブジェクトをドキュメントに 追加することが可能ですが、ノードがドキュメントに関連付けられるまでは ノードを追加することはできません。書き込み可能なノードを作成するには、DOMDocument->createElement() あるいは DOMDocument->createElementNS() を使用します。

パラメータ

name

要素のタグ名。namespaceURI をともに指定した場合、 この URI に関連付けられたプレフィックスが与えられます。

value

要素の値。

namespaceURI

指定した名前空間の中で要素を作成するための 名前空間 URI。

例

Example#1 新しい DOMElement を作成する

```php
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');
$element  = $dom->appendChild(new DOMElement('root'));
$element_ns  = new DOMElement('pr:node1', 'thisvalue', 'http://xyz');
$element->appendChild($element_ns);
echo $dom->saveXML(); /*  <?xml  version="1.0"  encoding="iso-8859-1"?>
<root><pr:node1  xmlns:pr="http://xyz">thisvalue</pr:node1></root>  */

?>
```

参考

- DOMDocument->createElement()
- DOMDocument->createElementNS()

DOMElement->getAttribute()

(No version information available, might be only in CVS)

DOMElement->getAttribute() — 属性の値を返す

説明

DOMElement
string **getAttribute** (string $name)

現在のノードから、名前が *name* である 属性の値を取得します。

パラメータ

name

属性の名前。

返り値

属性の値、あるいは *name* に対応する属性が 見つからなかった場合には空の文字列を返します。

参考

- DOMElement->hasAttribute()
- DOMElement->setAttribute()
- DOMElement->removeAttribute()

DOMElement->getAttributeNode()

(No version information available, might be only in CVS)

DOMElement->getAttributeNode() — 属性ノードを返す

説明

DOMElement
DOMAttr **getAttributeNode** (string $name)

現在の要素の、*name* という名前の 属性ノードを返します。

パラメータ

name

属性の名前。

返り値

属性ノードを返します。

参考

- DOMElement->hasAttribute()
- DOMElement->setAttributeNode()
- DOMElement->removeAttributeNode()

DOMElement->getAttributeNodeNS()

(No version information available, might be only in CVS)

DOMElement->getAttributeNodeNS() — 属性ノードを返す

説明

DOMElement
DOMAttr **getAttributeNodeNS** (string $namespaceURI , string $localName)

現在のノードの、名前空間 *namespaceURI* における *localName* というローカル名の 属性ノードを返します。

パラメータ

namespaceURI

名前空間 URI。

localName

ローカル名。

返り値

属性ノードを返します。

参考

- DOMElement->hasAttributeNS()
- DOMElement->setAttributeNodeNS()
- DOMElement->removeAttributeNode()

DOMElement->getAttributeNS()

(No version information available, might be only in CVS)

DOMElement->getAttributeNS() — 属性の値を返す

説明

DOMElement
string **getAttributeNS** (string `$namespaceURI` , string `$localName`)

現在のノードから、名前空間 *namespaceURI* における *localName* というローカル名の 属性の値を取得します。

パラメータ

namespaceURI

名前空間 URI。

localName

ローカル名。

返り値

属性の値、あるいは *localName* および *namespaceURI* に対応する属性が 見つからなかった場合には空の文字列を返します。

参考

- DOMElement->hasAttributeNS()
- DOMElement->setAttributeNS()
- DOMElement->removeAttributeNS()

DOMElement->getElementsByTagName()

(No version information available, might be only in CVS)

DOMElement->getElementsByTagName() — タグ名から要素を取得する

説明

DOMElement
DOMNodeList **getElementsByTagName** (string `$name`)

この関数は、指定した名前 *name* のタグを持つ すべての子孫要素からなるクラス DOMNodeList の新しいインスタンスを返します。要素は、ツリーをたどっていく際に 見つかった順に並べられます。

パラメータ

name

タグ名。ツリー内のすべての要素を返すには * を使用します。

返り値

この関数は、一致したすべての要素からなる DOMNodeList クラスのインスタンスを返します。

参考

- DOMElement->getElementsByTagNameNS()

DOMElement->getElementsByTagNameNS()

(No version information available, might be only in CVS)

DOMElement->getElementsByTagNameNS() — 名前空間 URI とローカル名から要素を取得する

説明

DOMElement
DOMNodeList **getElementsByTagNameNS** (string `$namespaceURI` , string `$localName`)

この関数は、指定した名前 *localName* および *namespaceURI* のタグを持つ すべての子孫要素を取得します。

パラメータ

namespaceURI

名前空間 URI。

localName

ローカル名。ツリー内のすべての要素を返すには * を使用します。

返り値

この関数は、一致したすべての要素からなる DOMNodeList クラスのインスタンスを返します。 各要素は、ツリー内の探索時に見つかった順で並べられます。

参考

- DOMElement->getElementsByTagName()

DOMElement->hasAttribute()

(No version information available, might be only in CVS)

DOMElement->hasAttribute() — 属性が存在するかどうかを調べる

説明

DOMElement
bool **hasAttribute** (string `$name`)

要素のメンバとして *name* という名前の属性が 存在するかどうかを示します。

パラメータ

name

要素名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOMElement->hasAttributeNS()
- DOMElement->getAttribute()
- DOMElement->setAttribute()
- DOMElement->removeAttribute()

DOMElement->hasAttributeNS()

(No version information available, might be only in CVS)

DOMElement->hasAttributeNS() — 属性が存在するかどうかを調べる

説明

DOMElement
bool **hasAttributeNS** (string `$namespaceURI` , string `$localName`)

要素のメンバとして *localName* という名前の属性が 名前空間 *namespaceURI* に存在するかどうかを示します。

パラメータ

namespaceURI

名前空間 URI。

localName

ローカル名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOMElement->hasAttribute()
- DOMElement->getAttributeNS()
- DOMElement->setAttributeNS()
- DOMElement->removeAttributeNS()

DOMElement->removeAttribute()

(No version information available, might be only in CVS)

DOMElement->removeAttribute() — 属性を削除する

説明

DOMElement
bool **removeAttribute** (string `$name`)

name という名前の属性を要素から削除します。

パラメータ

name

属性の名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

参考

- DOMElement->hasAttribute()
- DOMElement->getAttribute()
- DOMElement->setAttribute()

DOMElement->removeAttributeNode()

(No version information available, might be only in CVS)

DOMElement->removeAttributeNode() — 属性を削除する

説明

DOMElement
bool **removeAttributeNode** (DOMAttr `$oldnode`)

属性 *oldnode* を要素から削除します。

パラメータ

oldnode

属性ノード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

DOM_NOT_FOUND_ERROR

oldnode が要素の属性でない場合に発生します。

参考

- DOMElement->hasAttribute()
- DOMElement->getAttributeNode()
- DOMElement->setAttributeNode()

DOMElement->removeAttributeNS()

(No version information available, might be only in CVS)

DOMElement->removeAttributeNS() — 属性を削除する

説明

DOMElement
bool **removeAttributeNS** (string `$namespaceURI` , string `$localName`)

名前空間 *namespaceURI* にある *localName* という名前の属性を 要素から削除します。

パラメータ

namespaceURI

名前空間 URI。

localName

ローカル名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

参考

- DOMElement->hasAttributeNS()
- DOMElement->getAttributeNS()
- DOMElement->setAttributeNS()

DOMElement->setAttribute()

(No version information available, might be only in CVS)

DOMElement->setAttribute() — 新しい属性を追加する

説明

DOMElement
bool **setAttribute** (string $name , string $value)

name という名前の属性を、指定した値に設定します。 属性が存在しない場合は、作成されます。

パラメータ

name

属性の名前。

value

属性の値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

例

Example#1 属性を設定する

```
<?php
$doc  = new DOMDocument("1.0");
$node  = $doc->createElement("para");
$newnode  = $doc->appendChild($node);
$newnode->setAttribute("align", "left");
?>
```

参考

- DOMElement->hasAttribute()
- DOMElement->getAttribute()
- DOMElement->removeAttribute()

DOMElement->setAttributeNode()

(No version information available, might be only in CVS)

DOMElement->setAttributeNode() — 新しい属性ノードを要素に追加する

説明

DOMElement
DOMAttr **setAttributeNode** (DOMAttr `$attr`)

新しい属性ノード *attr* を要素に追加します。

パラメータ

attr

属性ノード。

返り値

属性が置換された場合は置換前のノード、そうでない場合は **NULL** を返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

参考

- DOMElement->hasAttribute()
- DOMElement->getAttributeNode()
- DOMElement->removeAttributeNode()

DOMElement->setAttributeNodeNS()

(No version information available, might be only in CVS)

DOMElement->setAttributeNodeNS() — 新しい属性ノードを要素に追加する

説明

DOMElement
DOMAttr **setAttributeNodeNS** (DOMAttr `$attr`)

新しい属性ノード *attr* を要素に追加します。

パラメータ

name

属性ノード。

返り値

属性が置換された場合は置換前のノードを返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

参考

- DOMElement->hasAttributeNS()
- DOMElement->getAttributeNodeNS()
- DOMElement->removeAttributeNode()

DOMElement->setAttributeNS()

(No version information available, might be only in CVS)

DOMElement->setAttributeNS() — 新しい属性を追加する

説明

DOMElement
void **setAttributeNS** (string `$namespaceURI` , string `$qualifiedName` , string `$value`)

名前空間 *namespaceURI* にある *localName* という名前の属性を、指定した値に設定します。 属性が存在しない場合は、作成されます。

パラメータ

namespaceURI

名前空間 URI。

qualifiedName

prefix:tagname 形式で表した属性名。

value

属性の値。

返り値

値を返しません。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

DOM_NAMESPACE_ERR

qualifiedName が不正な形式であった場合や、 *qualifiedName* がプレフィックスを含むにもかかわらず *namespaceURI* が **NULL** の場合に発生します。

参考

- DOMElement->hasAttributeNS()
- DOMElement->getAttributeNS()
- DOMElement->removeAttributeNS()

DOMElement->setIdAttribute()

(No version information available, might be only in CVS)

DOMElement->setIdAttribute() — ID 型の属性を名前で宣言する

説明

DOMElement
void **setIdAttribute** (string `$name` , bool `$isId`)

ID 型となる属性を *name* で宣言します。

パラメータ

name

属性の名前。

isId

name を ID 型にしたい場合に **TRUE**、 それ以外の場合に **FALSE** を設定します。

返り値

値を返しません。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用である場合に発生します。

DOM_NOT_FOUND

name がこの要素の属性でない場合に発生します。

参考

- DOMDocument->getElementById()
- DOMElement->setIdAttributeNode()
- DOMElement->setIdAttributeNS()

DOMElement->setIdAttributeNode()

(No version information available, might be only in CVS)

DOMElement->setIdAttributeNode() — ID 型の属性をノードで宣言する

説明

DOMElement
void **setIdAttributeNode** (DOMAttr `$attr` , bool `$isId`)

ID 型となる属性 *attr* を宣言します。

パラメータ

attr

属性ノード。

isId

name を ID 型にしたい場合に **TRUE**、 それ以外の場合に **FALSE** を設定します。

返り値

値を返しません。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用である場合に発生します。

DOM_NOT_FOUND

name がこの要素の属性でない場合に発生します。

参考

- DOMDocument->getElementById()
- DOMElement->setIdAttribute()
- DOMElement->setIdAttributeNS()

DOMElement->setIdAttributeNS()

(No version information available, might be only in CVS)

DOMElement->setIdAttributeNS() — ID 型の属性をローカル名および名前空間 URI で宣言する

説明

DOMElement
void **setIdAttributeNS** (string `$namespaceURI` , string `$localName` , bool `$isId`)

ID 型となる属性を *localName* および *namespaceURI* で宣言します。

パラメータ

namespaceURI

属性の名前空間 URI。

localName

属性のローカル名。*prefix:tagname* 形式。

isId

name を ID 型にしたい場合に **TRUE**、 それ以外の場合に **FALSE** を設定します。

返り値

値を返しません。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用である場合に発生します。

DOM_NOT_FOUND

name がこの要素の属性でない場合に発生します。

参考

- DOMDocument->getElementById()
- DOMElement->setIdAttribute()
- DOMElement->setIdAttributeNode()

DOMEntityReference->__construct()

(PHP 5)

DOMEntityReference->__construct() — 新しい DOMEntityReference オブジェクトを作成する

説明

DOMEntityReference
__construct (string `$name`)

新しい DOMEntityReference オブジェクトを作成します。

パラメータ

name

エンティティ参照の名前。

例

Example#1 新しい DOMEntityReference を作成する

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');
$element = $dom->appendChild(new DOMElement('root'));
$entity = $element->appendChild(new DOMEntityReference('nbsp'));
echo $dom->saveXML(); /* <?xml version="1.0" encoding="iso-8859-1"?><root> </root> */

?>
```

参考

- DOMDocument->createEntityReference()

DOMImplementation->__construct()

(No version information available, might be only in CVS)

DOMImplementation->__construct() — 新しい DOMImplementation オブジェクトを作成する

説明

DOMImplementation
__construct (void)

新しい DOMImplementation オブジェクトを作成します。

DOMImplementation->createDocument()

(No version information available, might be only in CVS)

DOMImplementation->createDocument() — 指定した型とドキュメント要素の DOMDocument オブジェクトを作成する

説明

DOMImplementation
DOMDocument **createDocument** ([string `$namespaceURI` [, string `$qualifiedName` [, DOMDocumentType `$doctype`]]])

指定した型とドキュメント要素の DOMDocument オブジェクトを作成します。

パラメータ

`namespaceURI`

作成するドキュメント要素の名前空間 URI。

`qualifiedName`

作成するドキュメント要素の修飾名。

`doctype`

作成するドキュメントの型、あるいは **`NULL`**。

返り値

新しい DOMDocument オブジェクトを返します。 *`namespaceURI`* 、*`qualifiedName`* および *`doctype`* が null の場合は、 ドキュメント要素を含まない空の DOMDocument を返します。

エラー / 例外

`DOM_WRONG_DOCUMENT_ERR`

`doctype` が既に別のドキュメントで使用されていたり、 別の実装で作成されている場合に発生します。

`DOM_NAMESPACE_ERR`

`namespaceURI` および *`qualifiedName`* で指定した名前空間に間違いがある場合に発生します。

参考

- DOMDocument->__construct()
- DOMImplementation->createDocumentType()

DOMImplementation->createDocumentType()

(No version information available, might be only in CVS)

DOMImplementation->createDocumentType() — 空の DOMDocumentType オブジェクトを作成する

説明

DOMImplementation
DOMDocumentType **createDocumentType** ([string `$qualifiedName` [, string `$publicId` [, string `$systemId`]]])

空の DOMDocumentType オブジェクトを作成します。 エンティティ宣言および記法は使用可能にはなりません。エンティティ参照の 展開やデフォルト属性の追加は発生しません。

パラメータ

`qualifiedName`

作成されるドキュメント型の修飾名。

`publicId`

外部サブセットの公開 ID。

`systemId`

外部サブセットのシステム ID。

返り値

ownerDocument を **`NULL`** に設定した 新しい DOMDocumentType ノードを返します。

例

Example#1 DTD を添付してドキュメントを作成する

```php
<?php

// DOMImplementation クラスのインスタンスを作成します。
$imp = new DOMImplementation;

// DOMDocumentType のインスタンスを作成します。
$dtd = $imp->createDocumentType('graph', '', 'graph.dtd');

// DOMDocument のインスタンスを作成します。
$dom = $imp->createDocument("", "", $dtd);

// その他のプロパティを設定します。
$dom->encoding = 'UTF-8';
$dom->standalone = false;

// 空の要素を作成します。
$element = $dom->createElement('graph');

// 要素を追加します。
$dom->appendChild($element);

// ドキュメントの内容を出力します。
echo $dom->saveXML();

?>
```

上の例の出力は以下となります。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE graph SYSTEM "graph.dtd">
<graph/>
```

エラー / 例外

`DOM_NAMESPACE_ERR`

`qualifiedName` で指定した名前空間に 間違いがある場合に発生します。

参考

- DOMImplementation->createDocument()

DOMImplementation->hasFeature()

(No version information available, might be only in CVS)

DOMImplementation->hasFeature() — DOM implementation が、指定した機能を実装しているかどうかを調べる

説明

DOMImplementation
bool **hasFeature** (string `$feature` , string `$version`)

DOM implementation が、指定した機能 *feature* を実装しているかどうかを調べます。

DOM 仕様の » Conformance の節に、すべての機能の一覧があります。

パラメータ

feature

調べる機能。

version

調べる *feature* のバージョン番号。 DOM level 2 においては、これは *2.0* あるいは *1.0* のいずれかです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 DOM Implementation を調べる

```
<?php

$features = array(
  'Core'             => 'Core module',
  'XML'              => 'XML module',
  'HTML'             => 'HTML module',
  'Views'            => 'Views module',
  'Stylesheets'      => 'Style Sheets module',
  'CSS'              => 'CSS module',
  'CSS2'             => 'CSS2 module',
  'Events'           => 'Events module',
  'UIEvents'         => 'User interface Events module',
  'MouseEvents'      => 'Mouse Events module',
  'MutationEvents'   => 'Mutation Events module',
  'HTMLEvents'       => 'HTML Events module',
  'Range'            => 'Range module',
  'Traversal'        => 'Traversal module'
);

foreach ($features as $key => $name) {
  if (DOMImplementation::hasFeature($key, '2.0')) {
    echo "$name の機能を保持しています\n";
  } else {
    echo "$name の機能は保持していません\n";
  }
}

?>
```

参考

- DOMNode->isSupported()

DOMNamedNodeMap->getNamedItem()

(No version information available, might be only in CVS)

DOMNamedNodeMap->getNamedItem() — 名前で指定されたノードを取得する

説明

DOMNamedNodeMap
DOMNode **getNamedItem** (string `$name`)

nodeName で指定されたノードを取得します。

パラメータ

name

取得するノードの名前。

返り値

指定した (任意の型の) *nodeName* のノード、 あるいはノードが見つからなかった場合には **NULL** を返します。

参考

- DOMNamedNodeMap->getNamedItemNS()

DOMNamedNodeMap->getNamedItemNS()

(No version information available, might be only in CVS)

DOMNamedNodeMap->getNamedItemNS() — ローカル名および名前空間 URI で指定したノードを取得する

説明

DOMNamedNodeMap
DOMNode **getNamedItemNS** (string `$namespaceURI` , string `$localName`)

localName および *namespaceURI* で指定したノードを取得します。

パラメータ

namespaceURI

取得するノードの名前空間 URI。

localName

取得するノードのローカル名。

返り値

指定した (任意の型の) ローカル名および名前空間 URI のノード、 あるいはノードが見つからなかった場合には **NULL** を返します。

参考

- DOMNamedNodeMap->getNamedItem()

DOMNamedNodeMap->item()

(No version information available, might be only in CVS)

DOMNamedNodeMap->item() — インデックスで指定したノードを取得する

説明

DOMNamedNodeMap
DOMNode **item** (int `$index`)

DOMNamedNodeMap オブジェクトから、 *index* で指定したノードを取得します。

パラメータ

index

マップ内のインデックス。

返り値

マップ内の *index* 番目の位置にあるノード、 あるいはインデックスが不正な形式 (マップ内の要素数以上) の場合は **NULL** を返します。

DOMNode->appendChild()

(No version information available, might be only in CVS)

DOMNode->appendChild() — 子要素群の最後に新しい子要素を追加する

説明

DOMNode
DOMNode **appendChild** (DOMNode $newnode)

この関数は、既存の子要素のリストに新しい子要素を追加するか、 あるいは新しい子要素リストを作成します。子要素の作成には DOMDocument->createElement()、 DOMDocument->createTextNode() などを使用するか、単に別のノードを使用します。

パラメータ

newnode

追加する子要素。

返り値

The node added.

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用であったり、挿入するノードの以前の親が 読み込み専用であったりした場合に発生します。

DOM_HIERARCHY_REQUEST_ERR

newnode で指定した型の子ノードを 保持することが許可されていない場合、あるいは追加しようとしている ノードが自分自身やその祖先であった場合に発生します。

DOM_WRONG_DOCUMENT_ERR

newnode が、このノードとは別の ドキュメントで作成されたものである場合に発生します。

例

以下の例は、新しいドキュメントに新しい要素を追加します。

Example#1 子要素の追加

```
<?php

$doc = new DOMDocument;

$node = $doc->createElement("para");
$newnode = $doc->appendChild($node);

echo $doc->saveXML();
?>
```

参考

- DOMNode->removeChild()
- DOMNode->replaceChild()

DOMNode->cloneNode()

(No version information available, might be only in CVS)

DOMNode->cloneNode() — ノードを複製する

説明

DOMNode
DOMNode **cloneNode** ([bool $deep])

ノードのコピーを作成します。

パラメータ

deep

子孫要素を含めてコピーするかどうかを指定します。 このパラメータのデフォルト値は **FALSE** です。

返り値

複製されたノードを返します。

DOMNode->hasAttributes()

(No version information available, might be only in CVS)

DOMNode->hasAttributes() — ノードが属性を保持しているかどうかを調べる

説明

DOMNode
bool **hasAttributes** (void)

このメソッドは、ノードが属性を保持しているかどうかを調べます。 調べるノードは **XML_ELEMENT_NODE** である必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOMNode->hasChildNodes()

DOMNode->hasChildNodes()

(No version information available, might be only in CVS)

DOMNode->hasChildNodes() — ノードが子を保持しているかどうかを調べる

説明

DOMNode
bool **hasChildNodes** (void)

この関数は、ノードが子を保持しているかどうかを調べます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOMNode->hasAttributes()

DOMNode->insertBefore()

(No version information available, might be only in CVS)

DOMNode->insertBefore() — 参照しているノードの前に新しい子を追加する

説明

DOMNode
DOMNode **insertBefore** (DOMNode `$newnode` [, DOMNode `$refnode`])

この関数は、参照しているノードの直前に新しいノードを挿入します。 追加するノードに対して変更を加えたい場合は、この関数から返される ノードを使用します。

パラメータ

newnode

新しいノード。

refnode

参照ノード。指定されなかった場合は、*newnode* が子要素として追加されます。

返り値

挿入されたノードを返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

このノードが読み込み専用であったり、挿入されるノードの以前の親が 読み込み専用であった場合に発生します。

DOM_HIERARCHY_REQUEST_ERR

newnode で指定した型の子ノードを 保持することが許可されていない場合、あるいは追加しようとしている ノードが自分自身やその祖先であった場合に発生します。

DOM_WRONG_DOCUMENT_ERR

newnode が、このノードとは別の ドキュメントで作成されたものである場合に発生します。

DOM_NOT_FOUND

refnode がこのノードの子ではない場合に 発生します。

DOMNode->isDefaultNamespace()

(No version information available, might be only in CVS)

DOMNode->isDefaultNamespace() — 指定した namespaceURI がデフォルトの名前空間かどうかを調べる

説明

DOMNode
bool **isDefaultNamespace** (string `$namespaceURI`)

namespaceURI がデフォルトの名前空間であるかどうかを調べます。

パラメータ

namespaceURI

調べる名前空間 URI。

返り値

namespaceURI がデフォルトの名前空間である場合に **TRUE**、それ以外の場合に **FALSE** を返します。

DOMNode->isSameNode()

(No version information available, might be only in CVS)

DOMNode->isSameNode() — 2 つのノードが等しいかどうかを調べる

説明

DOMNode
bool **isSameNode** (DOMNode `$node`)

この関数は、2 つのノードが等しいかどうかを調べます。この比較は、 その内容に基づくものでは*ありません*。

パラメータ

`node`

比較対象となるノード。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

DOMNode->isSupported()

(No version information available, might be only in CVS)

DOMNode->isSupported() — 指定したバージョンで機能がサポートされているかどうかを調べる

説明

DOMNode
bool **isSupported** (string `$feature` , string `$version`)

指定したバージョン *`version`* で、機能 *`feature`* がサポートされているかどうかを調べます。

パラメータ

`feature`

調べる機能。機能の一覧については DOMImplementation->hasFeature() の例を参照ください。

`version`

調べる機能 *`feature`* のバージョン番号。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- DOMImplementation->hasFeature()

DOMNode->lookupNamespaceURI()

(No version information available, might be only in CVS)

DOMNode->lookupNamespaceURI() — プレフィックスに基づいて、ノードの名前空間 URI を取得する

説明

DOMNode
string **lookupNamespaceURI** (string `$prefix`)

`prefix` に基づいて、ノードの 名前空間 URI を取得します。

パラメータ

`prefix`

名前空間のプレフィックス。

返り値

ノードの名前空間 URI を返します。

参考

- DOMNode->lookupPrefix()

DOMNode->lookupPrefix()

(No version information available, might be only in CVS)

DOMNode->lookupPrefix() — 名前空間 URI に基づいて、ノードの名前空間プレフィックスを取得する

説明

DOMNode
string **lookupPrefix** (string `$namespaceURI`)

名前空間 URI に基づいて、ノードの名前空間プレフィックスを取得します。

パラメータ

`namespaceURI`

名前空間 URI。

返り値

名前空間のプレフィックスを返します。

参考

- DOMNode->lookupNamespaceURI()

DOMNode->normalize()

(No version information available, might be only in CVS)

DOMNode->normalize() — ノードを正規化する

説明

DOMNode
void **normalize** (void)

ノードを正規化します。

返り値

値を返しません。

参考

- » DOM 仕様
- DOMDocument->normalizeDocument()

DOMNode->removeChild()

(No version information available, might be only in CVS)

DOMNode->removeChild() — 子要素群から子要素を削除する

説明

DOMNode
DOMNode **removeChild** (DOMNode `$oldnode`)

この関数は、子要素群から子要素を削除します。

パラメータ

oldnode

削除する子要素。

返り値

子要素の削除に成功した場合に、削除した要素を返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

ノードが読み込み専用の場合に発生します。

DOM_NOT_FOUND

oldnode がこのノードの子要素でない場合に 発生します。

例

以下の例は、XML ドキュメントから chapter 要素を削除します。

Example#1 子要素の削除

```
<?php

$doc = new DOMDocument;
$doc->load('book.xml');

$book = $doc->documentElement;

// chapter を取得し、それを book から削除します
$chapter = $book->getElementsByTagName('chapter')->item(0);
$oldchapter = $book->removeChild($chapter);

echo $doc->saveXML();
?>
```

上の例の出力は以下となります。

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
                "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">
<book id="listing">
  <title>My lists</title>

</book>
```

参考

- DOMNode->appendChild()
- DOMNode->replaceChild()

DOMNode->replaceChild()

(No version information available, might be only in CVS)

DOMNode->replaceChild() — 子を置き換える

説明

DOMNode
DOMNode **replaceChild** (DOMNode `$newnode` , DOMNode `$oldnode`)

この関数は、子要素 *oldnode* を新しいノードに 置き換えます。もし新しいノードがすでに子要素であった場合は、 それがふたたび追加されることはありません。置き換えに成功すると、 置き換え前のノードが返されます。

パラメータ

newnode

新しいノード。対象ドキュメントのメンバ、 すなわち、DOMDocument->createXXX() メソッドのひとつで作成されたものか DOMDocument->importNode() で ドキュメントにインポートされたものである必要があります。

oldnode

古いノード。

返り値

古いノード、あるいはエラーが発生した場合は **FALSE** を返します。

エラー / 例外

DOM_NO_MODIFICATION_ALLOWED_ERR

このノードが読み込み専用であったり、挿入されるノードの以前の親が 読み込み専用であった場合に発生します。

DOM_HIERARCHY_REQUEST_ERR

newnode で指定した型の子ノードを 保持することが許可されていない場合、あるいは追加しようとしている ノードが自分自身やその祖先であった場合に発生します。

DOM_WRONG_DOCUMENT_ERR

newnode が、このノードとは別の ドキュメントで作成されたものである場合に発生します。

DOM_NOT_FOUND

oldnode がこのノードの子でない場合に 発生します。

参考

- DOMNode->appendChild()
- DOMNode->removeChild()

DOMNodelist->item()

(No version information available, might be only in CVS)

DOMNodelist->item() — インデックスで指定したノードを取得する

説明

DOMNodeList
DOMNode **item** (int $index)

DOMNodeList オブジェクトから、 *index* で指定したノードを取得します。

ヒント

コレクション内のノードの数を知るには、 DOMNodeList オブジェクトの *length* プロパティを使用します。

パラメータ

index

コレクション内のノードのインデックス。

返り値

DOMNodeList 内の *index* 番目の位置にあるノード、あるいはインデックスが不正な形式の場合は **NULL** を返します。

例

Example#1 テーブル内のすべての要素を取得する

```
<?php

$doc = new DOMDocument;
$doc->load('book.xml');

$items = $doc->getElementsByTagName('entry');

for ($i = 0; $i < $items->length; $i++) {
    echo $items->item($i)->nodeValue . "¥n";
}

?>
```

別の方法として foreach を使用することも可能で、こちらのほうが ずっと使いやすいでしょう。

```
<?php

foreach ($items as $item) {
    echo $item->nodeValue . "¥n";
}

?>
```

上の例の出力は以下となります。

```
Title
Author
Language
ISBN
The Grapes of Wrath
John Steinbeck
en
0140186409
The Pearl
John Steinbeck
en
014017737X
Samarcande
Amine Maalouf
fr
2253051209
```

DOMProcessingInstruction->__construct()

(PHP 5)

DOMProcessingInstruction->__construct() — 新しい DOMProcessingInstruction オブジェクトを作成する

説明

DOMProcessingInstruction
__construct (string $name [, string $value])

新しい DOMProcessingInstruction オブジェクトを 作成します。このオブジェクトは読み込み専用です。このオブジェクトを ドキュメントに追加することは可能ですが、ノードをドキュメントに 関連付けるまではこのノードに別のノードを追加することはできません。 書き込み可能なノードを作成するには、 DOMDocument->createProcessingInstruction() を使用してください。

パラメータ

name

処理命令のタグ名。

value

処理命令の値。

例

Example#1 新しい DOMProcessingInstruction を作成する

```
<?php

$dom = new DOMDocument('1.0', 'UTF-8');
$html = $dom->appendChild(new DOMElement('html'));
$body = $html->appendChild(new DOMElement('body'));
$pinode = new DOMProcessingInstruction('php', 'echo "Hello World"; ');
$body->appendChild($pinode);
echo $dom->saveXML();
```

```
?>
```

上の例の出力は以下となります。

```
<?xml  version="1.0"  encoding="UTF-8"?>
<html><body><?php  echo  "Hello  World";  ?></body></html>
```

参考

- DOMDocument->createProcessingInstruction()

DOMText->__construct()

(PHP 5)

DOMText->__construct() — 新しい DOMText オブジェクトを作成する

説明

DOMText
__construct ([string $value])

新しい DOMText オブジェクトを作成します。

パラメータ

value

text ノードの値。指定しなかった場合は空の text ノードが作成されます。

例

Example#1 新しい DOMText を作成する

```
<?php

$dom  =  new  DOMDocument('1.0',  'iso-8859-1');
$element  =  $dom->appendChild(new  DOMElement('root'));
$text  =  $element->appendChild(new  DOMText('root  value'));
echo  $dom->saveXML();  /*  <?xml  version="1.0"  encoding="iso-8859-1"?><root>root  value</root>  */

?>
```

参考

- DOMDocument->createTextNode()

DOMText->isWhitespaceInElementContent()

(No version information available, might be only in CVS)

DOMText->isWhitespaceInElementContent() — このテキストノードが空白を含むかどうかを示す

説明

DOMText
bool **isWhitespaceInElementContent** (void)

このテキストノードが空白を含むかどうかを示します。このテキストノードが 要素の内容に空白を含むかどうかは、ドキュメントの読み込み時に 決定されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

DOMText->splitText()

(No version information available, might be only in CVS)

DOMText->splitText() — 指定したオフセットでノードを 2 つに分割する

説明

DOMText
DOMText **splitText** (int `$offset`)

指定したオフセット *`offset`* でノードを 2 つに分割します。分割したノードのツリー内での位置関係は、 兄弟となります。

分割した後は、このノードは *`offset`* 位置までの 内容を保持するようになります。元のノードが親ノードを保持している場合、 新しいノードは元のノードの兄弟として元のノードの次の位置に挿入されます。 *`offset`* がこのノードの長さに等しい場合は、 新しいノードにはデータが含まれません。

パラメータ

`offset`

分割する位置を示すオフセット。0 から始まります。

返り値

同じ型の新しいノードを返します。*`offset`* 以降の内容をデータとして保持します。

DOMXPath->__construct()

(PHP 5)

DOMXPath->__construct() — 新しい DOMXPath オブジェクトを作成する

説明

DOMXPath
__construct (DOMDocument `$doc`)

新しい DOMXPath オブジェクトを作成します。

パラメータ

`doc`

DOMXPath に関連付けられた DOMDocument。

DOMXPath->evaluate()

(No version information available, might be only in CVS)

DOMXPath->evaluate() — 与えられた XPath 式を評価し、可能であれば結果を返す

説明

DOMXPath
mixed **evaluate** (string `$expression` [, DOMNode `$contextnode`])

与えられた XPath 式 *`expression`* を実行し、 可能であれば型付けされた結果を返します。

パラメータ

`expression`

実行する XPath 式。

`contextnode`

相対 XPath クエリを実行する場合に、オプションで基準ノードを 指定することが可能です。デフォルトでは、クエリは root 要素に 対する相対パスとなります。

返り値

可能であれば型付けされた結果、あるいは指定された XPath 式 *expression* にマッチするすべてのノードを含む DOMNodeList を返します。

例

Example#1 英語の書籍の数を取得する

```
<?php

$doc = new DOMDocument;

$doc->load('book.xml');

$xpath = new DOMXPath($doc);

$tbody = $doc->getElementsByTagName('tbody')->item(0);

// tbody ノードを基準とした相対クエリとなります
$query = 'count(row/entry[. = "en"])';

$entries = $xpath->evaluate($query, $tbody);
echo "There are $entries english books¥n";

?>
```

上の例の出力は以下となります。

```
There are 2 english books
```

参考

- DOMXPath->query()

DOMXPath->query()

(No version information available, might be only in CVS)

DOMXPath->query() — 与えられた XPath 式を評価する

説明

DOMXPath
DOMNodeList **query** (string `$expression` [, DOMNode `$contextnode`])

与えられた XPath 式 *expression* を評価します。

パラメータ

expression

実行する XPath 式。

contextnode

相対 XPath クエリを実行する場合に、オプションで基準ノードを 指定することが可能です。デフォルトでは、クエリは root 要素に 対する相対パスとなります。

返り値

与えられた XPath 式 *expression* にマッチする すべてのノードを含む DOMNodeList を返します。 ノードを返さない式の場合は、空の DOMNodeList を返します。

例

Example#1 すべての英語の書籍を取得する

```
<?php

$doc = new DOMDocument;
```

```php
// 空白に悩まされたくはありません
$doc->preserveWhiteSpace  = false;

$doc->Load('book.xml');

$xpath  = new DOMXPath($doc);

// root 要素から開始します
$query  = '//book/chapter/para/informaltable/tgroup/tbody/row/entry[.  =  "en"]';

$entries  = $xpath->query($query);

foreach  ($entries  as $entry) {
     echo "Found  {$entry->previousSibling->previousSibling->nodeValue},"  .
          "  by  {$entry->previousSibling->nodeValue}¥n";
}
?>
```

上の例の出力は以下となります。

```
Found  The  Grapes  of  Wrath,  by  John  Steinbeck
Found  The  Pearl,  by  John  Steinbeck
```

式を短くするため、*contextnode* パラメータを 使用することも可能です。

```php
<?php

$doc  = new DOMDocument;
$doc->preserveWhiteSpace  = false;

$doc->load('book.xml');

$xpath  = new DOMXPath($doc);

$tbody  = $doc->getElementsByTagName('tbody')->item(0);

// tbody ノードからの相対クエリです
$query  = 'row/entry[.  =  "en"]';

$entries  = $xpath->query($query, $tbody);

foreach  ($entries  as $entry) {
     echo "Found  {$entry->previousSibling->previousSibling->nodeValue},"  .
          "  by  {$entry->previousSibling->nodeValue}¥n";
}
?>
```

参考

- DOMXPath->evaluate()

DOMXPath->registerNamespace()

(No version information available, might be only in CVS)

DOMXPath->registerNamespace() — DOMXPath オブジェクトの名前空間を登録する

説明

DOMXPath
bool **registerNamespace** (string `$prefix` , string `$namespaceURI`)

DOMXPath オブジェクトに、 *namespaceURI* および *prefix* を登録します。

パラメータ

prefix

プレフィックス。

namespaceURI

名前空間の URI。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

dom_import_simplexml

(PHP 5)

dom_import_simplexml — SimpleXMLElement オブジェクトから DOMElement オブジェクトを取得する

説明

DOMElement **dom_import_simplexml** (SimpleXMLElement `$node`)

この関数は、SimpleXML クラスの ノード *node* を受け取り、それを DOMElement ノードに変換します。新しいオブジェクトは、 DOMElement ノードとして使用可能です。

パラメータ

node

SimpleXMLElement ノード。

返り値

追加された DOMElement ノード、あるいは 何らかのエラーが発生した場合は **FALSE** を返します。

例

Example#1 dom_import_simplexml() を用いて SimpleXML を DOM にインポートする

```
<?php

$sxe = simplexml_load_string('<books><book><title>blah</title></book></books>');

if ($sxe === false) {
    echo 'ドキュメントのパース中にエラー';
     exit;
}

$dom_sxe = dom_import_simplexml($sxe);
if (!$dom_sxe) {
    echo 'XML の変換中にエラー';
     exit;
}

$dom = new DOMDocument('1.0');
$dom_sxe = $dom->importNode($dom_sxe, true);
$dom_sxe = $dom->appendChild($dom_sxe);

echo $dom->saveXML();

?>
```

参考

- simplexml_import_dom()

目次

- DOMDocument->createProcessingInstruction() — 新しい PI ノードを作成する
- DOMDocument->createTextNode() — 新しいテキストノードを作成する
- DOMDocument->getElementById() — id に対応する要素を検索する
- DOMDocument->getElementsByTagName() — 指定したタグ名に対応するすべての要素を検索する
- DOMDocument->getElementsByTagNameNS() — 指定した名前空間で、タグ名に対応するすべての要素を検索する
- DOMDocument->importNode() — 現在のドキュメントにノードをインポートする
- DOMDocument->load() — ファイルから XML を読み込む
- DOMDocument->loadHTML() — 文字列から HTML を読み込む
- DOMDocument->loadHTMLFile() — ファイルから HTML を読み込む
- DOMDocument->loadXML() — 文字列から XML を読み込む
- DOMDocument->normalizeDocument() — ドキュメントを正規化する
- DOMDocument->registerNodeClass() — 基底ノード型を作成する際に使用する拡張クラスを登録する
- DOMDocument->relaxNGValidate() — ドキュメントを relaxNG で検証する
- DOMDocument->relaxNGValidateSource() — ドキュメントを relaxNG で検証する
- DOMDocument->save() — 内部の XML ツリーをファイルに出力する
- DOMDocument->saveHTML() — 内部のドキュメントを HTML 形式の文字列として出力する
- DOMDocument->saveHTMLFile() — 内部のドキュメントを HTML 形式でファイルに出力する
- DOMDocument->saveXML() — 内部の XML ツリーを文字列として出力する
- DOMDocument->schemaValidate() — スキーマに基づいてドキュメントを検証する
- DOMDocument->schemaValidateSource() — スキーマに基づいてドキュメントを検証する
- DOMDocument->validate() — DTD に基づいてドキュメントを検証する
- DOMDocument->xinclude() — DOMDocument オブジェクト内の XIncludes を置換する
- DOMDocumentFragment->appendXML() — 生の XML データを追加する
- DOMElement->__construct() — 新しい DOMElement オブジェクトを作成する
- DOMElement->getAttribute() — 属性の値を返す
- DOMElement->getAttributeNode() — 属性ノードを返す
- DOMElement->getAttributeNodeNS() — 属性ノードを返す
- DOMElement->getAttributeNS() — 属性の値を返す
- DOMElement->getElementsByTagName() — タグ名から要素を取得する
- DOMElement->getElementsByTagNameNS() — 名前空間 URI とローカル名から要素を取得する
- DOMElement->hasAttribute() — 属性が存在するかどうかを調べる
- DOMElement->hasAttributeNS() — 属性が存在するかどうかを調べる
- DOMElement->removeAttribute() — 属性を削除する
- DOMElement->removeAttributeNode() — 属性を削除する
- DOMElement->removeAttributeNS() — 属性を削除する
- DOMElement->setAttribute() — 新しい属性を追加する
- DOMElement->setAttributeNode() — 新しい属性ノードを要素に追加する
- DOMElement->setAttributeNodeNS() — 新しい属性ノードを要素に追加する
- DOMElement->setAttributeNS() — 新しい属性を追加する
- DOMElement->setIdAttribute() — ID 型の属性を名前で宣言する
- DOMElement->setIdAttributeNode() — ID 型の属性をノードで宣言する
- DOMElement->setIdAttributeNS() — ID 型の属性をローカル名および名前空間 URI で宣言する
- DOMEntityReference->__construct() — 新しい DOMEntityReference オブジェクトを作成する
- DOMImplementation->__construct() — 新しい DOMImplementation オブジェクトを作成する
- DOMImplementation->createDocument() — 指定した型とドキュメント要素の DOMDocument オブジェクトを作成する
- DOMImplementation->createDocumentType() — 空の DOMDocumentType オブジェクトを作成する
- DOMImplementation->hasFeature() — DOM implementation が、指定した機能を実装しているかどうかを調べる
- DOMNamedNodeMap->getNamedItem() — 名前で指定されたノードを取得する
- DOMNamedNodeMap->getNamedItemNS() — ローカル名および名前空間 URI で指定したノードを取得する
- DOMNamedNodeMap->item() — インデックスで指定したノードを取得する
- DOMNode->appendChild() — 子要素群の最後に新しい子要素を追加する
- DOMNode->cloneNode() — ノードを複製する
- DOMNode->hasAttributes() — ノードが属性を保持しているかどうかを調べる
- DOMNode->hasChildNodes() — ノードが子を保持しているかどうかを調べる
- DOMNode->insertBefore() — 参照しているノードの前に新しい子を追加する
- DOMNode->isDefaultNamespace() — 指定した namespaceURI がデフォルトの名前空間かどうかを調べる
- DOMNode->isSameNode() — 2 つのノードが等しいかどうかを調べる
- DOMNode->isSupported() — 指定したバージョンで機能がサポートされているかどうかを調べる
- DOMNode->lookupNamespaceURI() — プレフィックスに基づいて、ノードの名前空間 URI を取得する
- DOMNode->lookupPrefix() — 名前空間 URI に基づいて、ノードの名前空間プレフィックスを取得する
- DOMNode->normalize() — ノードを正規化する
- DOMNode->removeChild() — 子要素群から子要素を削除する
- DOMNode->replaceChild() — 子を置き換える
- DOMNodelist->item() — インデックスで指定したノードを取得する

- DOMProcessingInstruction->__construct() — 新しい DOMProcessingInstruction オブジェクトを作成する
- DOMText->__construct() — 新しい DOMText オブジェクトを作成する
- DOMText->isWhitespaceInElementContent() — このテキストノードが空白を含むかどうかを示す
- DOMText->splitText() — 指定したオフセットでノードを 2 つに分割する
- DOMXPath->__construct() — 新しい DOMXPath オブジェクトを作成する
- DOMXPath->evaluate() — 与えられた XPath 式を評価し、可能であれば結果を返す
- DOMXPath->query() — 与えられた XPath 式を評価する
- DOMXPath->registerNamespace() — DOMXPath オブジェクトの名前空間を登録する
- dom_import_simplexml — SimpleXMLElement オブジェクトから DOMElement オブジェクトを取得する

DOM XML 関数

導入

domxml 拡張モジュールは、DOM 標準に対する互換性を改善するため PHP バージョン 4.3.0 で書き直されました。拡張モジュールには まだ多くの古い関数が含まれていますが、使用は推奨されません。 特にオブジェクト指向でない関数の使用は避けるべきです。

拡張モジュールにより、DOM API で XML ドキュメントを処理することが 可能となります。また、完全な XML ドキュメントを PHP オブジェクトツリーに 変換する関数 domxml_xmltree() も提供されています。 現在、このツリーは読み込み専用とされています。 このツリーを修正することは可能ですが、DomDocument_dump_mem() にこれを適用することはできないため、意味はないでしょう。 XML ファイルを読み込んで修正した版を書き込みたい場合は DomDocument_create_element()、 **DomDocument_create_text()**、 **set_attribute()** 等を使用し、最後に DomDocument_dump_mem() 関数を使用してください。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.0.0.
>
> **注意**: この拡張は実験的なものではありません。しかしながら、PHP 5 版は決してリリースされないでしょう。PHP 4 でのみ配布されます。 もし PHP 5 でDOM XML をサポートする必要がある場合、 DOM 拡張を使用することができます。 この *domxml* 拡張は DOM 拡張と互換性はありません。

要件

この拡張モジュールは、» GNOME XML ライブラリを使用します。このライブラリをダウンロードし、 インストールしてください。少なくとも libxml-2.4.14 が必要です。 DOM XSLT 機能を使用するために » libxslt ライブラリと » http://www.exslt.org/ による EXSLT 拡張を使用することができます。 (拡張) XSLT 機能を使用するには、これらのライブラリをダウンロード、 インストールしてください。少なくとも libxslt-1.0.18 が必要です。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/domxml.

この拡張モジュールは、 --with-dom=[DIR]を指定してPHPを設 定した場合のみ利用可能です。DOM XSLTサポートを組み込むには、 --with-dom-xslt[=DIR] を追加して下 さい。DIRは、libxsltをインストールしたディレクトリです。DOM EXSLTサ ポートを有効にするには、 --with-dom-exslt[=DIR] を指定しま す。ただし、DIRはlibxsltをインストールしたディレクトリです。

Windows ユーザの場合、これらの関数を使用するには *php.ini* の *php_domxml.dll* を有効にしてください。 PHP 4 の場合、この DLL は PHP の Windows ダウンロードバイナリの *extensions/* ディレクトリにあります。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。 また、この拡張を有効にするためには、追加の DLL をシステムパスにコピーする必要があります。 PHP 4 では、この DLL は *dlls/* にあります。 DLL 名は、PHP <= 4.2.0 で *libxml2.dll*、 PHP >= 4.3.0 で *iconv.dll* です。 また、PHP 5.0.0 以降では iconv が Windows 用 PHP バイナリにデフォルトでコンパイルされていますので、 外部 DLL は必要ありません。

古い関数

DOM 標準に沿っておらず、もう使うべきではない関数がごく少数あります。 これらの関数を以下の表に示します。 関数 DomNode_append_child() はその動作が変更されました。 この関数は、子を追加しますが、兄弟は追加しません。 これにより、アプリケーションの動作に影響が生じる場合、DOM ではない関数 DomNode_append_sibling() を使用してください。

古い関数とその代替関数

古い関数	新しい関数
xmldoc	domxml_open_mem()
xmldocfile	domxml_open_file()
domxml_new_xmldoc	domxml_new_doc()
domxml_dump_mem	DomDocument_dump_mem()
domxml_dump_mem_file	DomDocument_dump_file()
DomDocument_dump_mem_file	DomDocument_dump_file()
DomDocument_add_root	DomDocument_create_element()の後に DomNode_append_child()
DomDocument_dtd	DomDocument_doctype()
DomDocument_root	DomDocument_document_element()
DomDocument_children	DomNode_child_nodes()
DomDocument_imported_node	代替関数なし。
DomNode_add_child	例えば DomDocument_create_element() により、 新しいノードを作成し、 DomNode_append_child() により追加します。
DomNode_children	DomNode_child_nodes()
DomNode_parent	DomNode_parent_node()
DomNode_new_child	例えば DomDocument_create_element() で 新規ノードを作成し、 DomNode_append_child() で追加します。
DomNode_set_content	例えば DomDocument_create_element() で 新規ノードを作成し、 DomNode_append_child() で追加します。
DomNode_get_content	コンテンツは単なるテキストノードであり、 DomNode_child_nodes() でアクセス可能です。
DomNode_set_content	コンテンツは単なるテキストノードであり、 DomNode_append_child() で追加できます。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

XML定数

定数	値	説明
XML_ELEMENT_NODE (integer)	1	ノードは要素(element)である
XML_ATTRIBUTE_NODE (integer)	2	ノードは属性(attribute)である
XML_TEXT_NODE (integer)	3	ノードはテキストの一部である
XML_CDATA_SECTION_NODE (integer)	4	
XML_ENTITY_REF_NODE (integer)	5	
XML_ENTITY_NODE (integer)	6	ノードは のようなエンティティである
XML_PI_NODE (integer)	7	ノードは処理命令(PI)である
XML_COMMENT_NODE (integer)	8	ノードはコメントである
XML_DOCUMENT_NODE (integer)	9	ノードはドキュメントである
XML_DOCUMENT_TYPE_NODE (integer)	10	
XML_DOCUMENT_FRAG_NODE (integer)	11	
XML_NOTATION_NODE (integer)	12	
XML_GLOBAL_NAMESPACE (integer)	1	
XML_LOCAL_NAMESPACE (integer)	2	
XML_HTML_DOCUMENT_NODE (integer)		
XML_DTD_NODE (integer)		
XML_ELEMENT_DECL_NODE (integer)		
XML_ATTRIBUTE_DECL_NODE (integer)		
XML_ENTITY_DECL_NODE (integer)		
XML_NAMESPACE_DECL_NODE (integer)		
XML_ATTRIBUTE_CDATA (integer)		

定数	値	説明
XML_ATTRIBUTE_ID (integer)		
XML_ATTRIBUTE_IDREF (integer)		
XML_ATTRIBUTE_IDREFS (integer)		
XML_ATTRIBUTE_ENTITY (integer)		
XML_ATTRIBUTE_NMTOKEN (integer)		
XML_ATTRIBUTE_NMTOKENS (integer)		
XML_ATTRIBUTE_ENUMERATION (integer)		
XML_ATTRIBUTE_NOTATION (integer)		
XPATH_UNDEFINED (integer)		
XPATH_NODESET (integer)		
XPATH_BOOLEAN (integer)		
XPATH_NUMBER (integer)		
XPATH_STRING (integer)		
XPATH_POINT (integer)		
XPATH_RANGE (integer)		
XPATH_LOCATIONSET (integer)		
XPATH_USERS (integer)		
XPATH_NUMBER (integer)		

クラス

モジュールの API は、DOM レベル 2 標準に可能な限り基づいています。 結果的に API は完全にオブジェクト指向です。このモジュールを使用する際に DOM 標準が利用できることは望ましいことです。 この API はオブジェクト指向ですが、 最初の引数として処理するオブジェクトを渡すことによってコールできる、 オブジェクト指向でない関数も多くあります。 これらの関数は、主に過去のバージョンとの互換性確保のために維持されていますが、 新規開発での使用はもはや推奨されません。

この API は、公式な DOM API とは二つの点で異なっています。まず、 全てのクラスの属性は、同じ名前を有する関数として実装され、関数名は PHP の命名表記に基づいています。これは、DOM 関数 lastChild() が last_child() と書かれることを意味します。

このモジュールは、一連のクラスを定義します。メソッドも含めたリストを 以下の表に示します。クラスは、DOM 標準で DOMxxx という名前のものと等価です。

クラスの一覧

クラス名	親クラス
DomAttribute	DomNode
DomCData	DomNode
DomComment	DomCData : DomNode
DomDocument	DomNode
DomDocumentType	DomNode
DomElement	DomNode
DomEntity	DomNode
DomEntityReference	DomNode
DomProcessingInstruction	DomNode
DomText	DomCData : DomNode
Parser	現在はまだ DomParser と呼ばれています
XPathContext	

DomDocument クラス (DomDocument : DomNode)

メソッド名	関数名	注意
doctype	DomDocument_doctype()	
document_element	DomDocument_document_element()	
create_element	DomDocument_create_element()	
create_text_node	DomDocument_create_text_node()	
create_comment	DomDocument_create_comment()	
create_cdata_section	DomDocument_create_cdata_section()	
create_processing_instruction	DomDocument_create_processing_instruction()	
create_attribute	DomDocument_create_attribute()	
create_entity_reference	DomDocument_create_entity_reference()	
get_elements_by_tagname	DomDocument_get_elements_by_tagname()	
get_element_by_id	DomDocument_get_element_by_id()	
dump_mem	DomDocument_dump_mem()	DOM 標準ではありません
dump_file	DomDocument_dump_file()	DOM 標準ではありません
html_dump_mem	DomDocument_html_dump_mem()	DOM 標準ではありません
xpath_init	xpath_init	DOM 標準ではありません
xpath_new_context	xpath_new_context	DOM 標準ではありません
xptr_new_context	xptr_new_context	DOM 標準ではありません

DomElement クラス (DomElement : DomNode)

メソッド名	関数名	注意
tagname	DomElement_tagname()	
get_attribute	DomElement_get_attribute()	
set_attribute	DomElement_set_attribute()	
remove_attribute	DomElement_remove_attribute()	
get_attribute_node	DomElement_get_attribute_node()	
set_attribute_node	DomElement_set_attribute_node()	
get_elements_by_tagname	DomElement_get_elements_by_tagname()	
has_attribute	DomElement_has_attribute()	

DomNode クラス

メソッド名	注意
DomNode_node_name()	
DomNode_node_value()	
DomNode_node_type()	
DomNode_last_child()	
DomNode_first_child()	
DomNode_child_nodes()	
DomNode_previous_sibling()	
DomNode_next_sibling()	
DomNode_parent_node()	
DomNode_owner_document()	
DomNode_insert_before()	
DomNode_append_child()	
DomNode_append_sibling()	DOM 標準ではありません。この関数は 先ほどの DomNode_append_child() の動作をエミュレートします。
DomNode_remove_child()	
DomNode_has_child_nodes()	
DomNode_has_attributes()	
DomNode_clone_node()	
DomNode_attributes()	
DomNode_unlink_node()	DOM 標準ではありません
DomNode_replace_node()	DOM 標準ではありません
DomNode_set_content()	DOM 標準にはなく、古いメソッドです
DomNode_get_content()	DOM 標準にはなく、古いメソッドです
DomNode_dump_node()	DOM 標準ではありません
DomNode_is_blank_node()	DOM 標準ではありません

DomAttribute クラス (DomAttribute : DomNode)

メソッド名		注意
name	DomAttribute_name()	
value	DomAttribute_value()	
specified	DomAttribute_specified()	

DomProcessingInstruction クラス (DomProcessingInstruction : DomNode)

メソッド名	関数名	注意
target	DomProcessingInstruction_target()	
data	DomProcessingInstruction_data()	

Parser クラス

メソッド名	関数名	注記
add_chunk	Parser_add_chunk()	
end	Parser_end()	

XPathContext クラス

メソッド名	関数名	注記
eval	XPathContext_eval()	
eval_expression	XPathContext_eval_expression()	
register_ns	XPathContext_register_ns()	

DomDocumentType クラス (DomDocumentType : DomNode)

メソッド名	関数名	注記
name	DomDocumentType_name()	
entities	DomDocumentType_entities()	
notations	DomDocumentType_notations()	
public_id	DomDocumentType_public_id()	
system_id	DomDocumentType_system_id()	
internal_subset	DomDocumentType_internal_subset()	

クラス DomDtd は DomNode から派生したものです。DomComment は DomCData から派生したものです。

例

このリファレンスにおける多くの例は、XML 文字列を必要とします。 この文字列を全ての例で繰りかえし記述するかわりに、ファイルに書き込んで 各例で読み込むことにします。この読み込まれるファイルは、 以下の例に示されています。XML ドキュメントを作成し、 **DomDocument_open_file()** で読み込むことも可能です。

Example#1 XML 文字列を有するファイル example.inc を読み込む

```
<?php
$xmlstr  =  "<?xml  version='1.0'  standalone='yes'?>
<!DOCTYPE  chapter  SYSTEM  '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'
[  <!ENTITY  sp  ¥"spanish¥">
]>
<!--  lsfj   -->
<chapter  language='en'><title  language='en'>Title</title>
  <para  language='ge'>
   &sp;
    <!--  comment  -->
     <informaltable  ID='findme'  language='&sp;'>
      <tgroup  cols='3'>
       <tbody>
          <row><entry>a1</entry><entry
morerows='1'>b1</entry><entry>c1</entry></row>
<row><entry>a2</entry><entry>c2</entry></row>
           <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
       </tbody>
      </tgroup>
     </informaltable>
  </para>
</chapter>";
?>
```

DomAttribute->name

(No version information available, might be only in CVS)

DomAttribute->name — 属性の名前を返す

説明

DomAttribute
string **name** (void)

属性名を取得します。

返り値

属性名を返します。

PHP 5 への移行

DOMAttr の name プロパティを使用してください。

参考

- 使用例は DomAttribute->value を参照ください。

DomAttribute->set_value

(No version information available, might be only in CVS)

DomAttribute->set_value — 属性の値を設定する

説明

DomAttribute
bool **set_value** (string `$content`)

この関数は属性の値を設定します。

パラメータ

`content`

新しい値

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PHP 5 への移行

DOMAttr の value プロパティを設定してください。

参考

- DomAttribute->value

DomAttribute->specified

(No version information available, might be only in CVS)

DomAttribute->specified — 属性が指定されているかどうか調べる

説明

DomAttribute
bool **set_value** (string `$content`)

この関数は属性値を設定します。

パラメータ

`content`

新しい値

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PHP 5 への移行

DOMAttr の value プロパティを設定してください。 .

参考

- DomAttribute->value

DomAttribute->value

(No version information available, might be only in CVS)

DomAttribute->value — 属性の値を返す

説明

DomAttribute
string **value** (void)

この関数は属性の値を返します。

例

Example#1 ノードの全ての属性を取得する

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
      echo "Error while parsing the document¥n";
      exit;
}

$root = $dom->document_element();
$attrs = $root->attributes();

echo 'Attributes of ' . $root->node_name() . "¥n";
foreach ($attrs as $attribute) {
      echo ' - ' . $attribute->name . ' : ' . $attribute->value . "¥n";
}

?>
```

上の例の出力は以下となります。

```
Attributes of chapter
 - language : en
```

返り値

Returns the value of the attribute.

PHP 5 への移行

DOMAttr の value プロパティを使用してください。

参考

- DomAttribute->set_value
- DomAttribute->name

DomDocument->add_root

(No version information available, might be only in CVS)

DomDocument->add_root — ルートノードを追加する [推奨されません]

説明

domelement **DomDocument->add_root** (string $name)

DOM 文章にルート要素ノードを追加し、新しいノードを返します。 この要素名はパラメータで渡します。

Example#1 単純な HTML 文章のヘッダを生成する

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->add_root("html");
$head = $root->new_child("head", "");
$head->new_child("title", "Hier der Titel");
echo htmlentities($doc->dump_mem());
?>
```

DomDocument->create_attribute

(No version information available, might be only in CVS)

DomDocument->create_attribute — 新規属性を作成する

説明

domattribute **DomDocument->create_attribute** (string `$name` , string `$value`)

この関数は DomAttribute クラスの新規インスタンスを返します。 属性名は第 1 パラメータの値です。属性値は第 2 パラメータです。 このノードは、domnode_append_child() などで挿入されない限り、ドキュメント内に現われません。

エラーが発生した場合、戻り値は **FALSE** です。

domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_cdata_section(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), domnode_insert_before() も参照ください。

DomDocument->create_cdata_section

(No version information available, might be only in CVS)

DomDocument->create_cdata_section — 新規 cdata ノードを作成する

説明

domcdata **DomDocument->create_cdata_section** (string `$content`)

この関数は DomCData クラスの新規インスタンスを返します。 cdata の内容は渡されたパラメータの値です。 このノードは、domnode_append_child() などで挿入されない限り、ドキュメント内に現われません。

エラーが発生した場合、戻り値は **FALSE** です。

domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), domnode_insert_before() も参照ください。

DomDocument->create_comment

(No version information available, might be only in CVS)

DomDocument->create_comment — 新規コメントノードを作成する

説明

domcomment **DomDocument->create_comment** (string `$content`)

この関数は DomComment クラスの新規インスタンスを返します。 コメントの内容は渡されたパラメータの値です。このノードは、domnode_append_child() などで挿入されない限り、ドキュメント内に現われません。

エラーが発生した場合、戻り値は **FALSE** です。

domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), domnode_insert_before() も参照ください。

DomDocument->create_element_ns

(No version information available, might be only in CVS)

DomDocument->create_element_ns — 関連する名前空間を持つ新規要素ノードを作成する

説明

domelement **DomDocument->create_element_ns** (string `$uri` , string `$name` [, string `$prefix`])

この関数は DomElement クラスの新規インスタンスを返します。 要素のタグ名は渡されたパラメータ *name* の値です。 名前空間の URI は渡されたパラメータ *uri* の値です。文章のルートノードで同じ名前空間が URI がすでに宣言されている場合、その接頭辞が使用されます。 そうでない場合、オプションパラメータ *prefix* で与えられた接頭辞、もしくはランダムに設定された接頭辞が使用されます。 このノードは、domnode_append_child() などで挿入されない限り、ドキュメント内に現われません。

エラーが発生した場合、戻り値は **FALSE** です。

domdocument_create_element_ns(), domnode_add_namespace(), domnode_set_namespace(), domnode_append_child(), **domdocument_create_text()**, domdocument_create_comment(), domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), domnode_insert_before() も参照ください。

DomDocument->create_element

(No version information available, might be only in CVS)

DomDocument->create_element — 新規要素ノードを作成する

説明

object **DomDocument->create_element** (string `$name`)

この関数は DomElement クラスの新規インスタンスを返します。 要素のタグ名は渡されたパラメータの値です。このノードは、例えば DomNode_append_child() などを用いて挿入されるまで、 文章には現れません。

エラーが発生した場合、戻り値は **FALSE** です。

DomNode_append_child(), **DomDocument_create_text()**, DomDocument_create_comment(), DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before() も参照ください。

DomDocument->create_entity_reference

(No version information available, might be only in CVS)

DomDocument->create_entity_reference — エンティティ参照を作成する

説明

domentityreference **DomDocument->create_entity_reference** (string `$content`)

この関数は DomEntityReference クラスの新規インスタンスを返します。 エンティティ参照の内容は渡されたパラメータの値です。このノードは、domnode_append_child() などで挿入されない限り、ドキュメント内に現われません。

エラーが発生した場合、戻り値は **FALSE** です。

domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_cdata_section(), domdocument_create_processing_instruction(), domdocument_create_attribute(), domnode_insert_before() も参照ください。

DomDocument->create_processing_instruction

(No version information available, might be only in CVS)

DomDocument->create_processing_instruction — 新規 PI ノードを作成する

説明

domprocessinginstruction **DomDocument->create_processing_instruction** (string `$content`)

この関数は DomCData クラスの新規インスタンスを返します。 PI の内容は渡されたパラメータの値です。このノードは、domnode_append_child() などで挿入されない限り、ドキュメント内に現われません。

エラーが発生した場合、戻り値は **FALSE** です。

domnode_append_child(), domdocument_create_element(), **domdocument_create_text()**, domdocument_create_cdata_section(), domdocument_create_attribute(), domdocument_create_entity_reference(), domnode_insert_before() も参照ください。

DomDocument->create_text_node

(No version information available, might be only in CVS)

DomDocument->create_text_node — 新規テキストノードを作成する

説明

domtext **DomDocument->create_text_node** (string `$content`)

この関数は DomText クラスの新規インスタンスを返します。 テキストの内容は渡されたパラメータの値です。このノードは、 domnode_append_child() などで挿入されない限り、ドキュメント内に現われません。

エラーが発生した場合、戻り値は **FALSE** です。

domnode_append_child(), domdocument_create_element(), domdocument_create_comment(), **domdocument_create_text()**, domdocument_create_attribute(), domdocument_create_processing_instruction(), domdocument_create_entity_reference(), domnode_insert_before() も参照ください。

DomDocument->doctype

(No version information available, might be only in CVS)

DomDocument->doctype — 文章型を返す

説明

domdocumenttype **DomDocument->doctype** (void)

この関数は、クラス DomDocumentType のオブジェクトを返します。4.3より前のバージョンのPHPでは、 このオブジェクトは、クラス Dtd でしたが、 DOM 標準にはそのようなクラスはありません。

クラスDomDocumentTypeのメソッドも参照下さい。

DomDocument->document_element

(No version information available, might be only in CVS)

DomDocument->document_element — ルート要素ノードを返す

説明

domelement **DomDocument->document_element** (void)

この関数は文章のルート要素ノードを返します。

以下の例はただ CHAPTER という名前の要素を返し表示します。 他のノード -- コメント -- は返しません。

Example#1 ルート要素を取得する

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$root = $dom->document_element();
print_r($root);
?>
```

上の例の出力は以下となります。

```
domelement Object
(
     [type] => 1
     [tagname] => chapter
     [0] => 6
     [1] => 137960648
)
```

DomDocument->dump_file

(No version information available, might be only in CVS)

DomDocument->dump_file — 内部 XML ツリーをファイルにダンプする

説明

string **DomDocument->dump_file** (string `$filename` [, bool `$compressionmode` [, bool `$format`]])

DOM 表現から XML 文章を生成します。この関数は以下の例のように、 通常スクラッチから新規 DOM 文章を生成した後にコールされます。 *format* により、 きちんと整形するかどうかを指定します。 第 1 パラメータはファイル名、第 2 パラメータは圧縮するかどうかを指定します。

Example#1 簡単な HTML 文章のヘッダを生成する

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
$doc->dump_file("/tmp/test.xml", false, true);
?>
```

domdocument_dump_mem(), domdocument_html_dump_mem() も参照ください。

DomDocument->dump_mem

(No version information available, might be only in CVS)

DomDocument->dump_mem — 内部 XML ツリーを文字列にダンプする

説明

string **DomDocument->dump_mem** ([bool `$format` [, string `$encoding`]])

DOM 表現から XML 文章を生成します。この関数は以下の例のように、 通常スクラッチから新規 DOM 文章を生成した後にコールされます。 *format* により、 きちんと整形するかどうかを指定します。

Example#1 簡単な HTML 文章のヘッダを生成する

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->dump_mem(true));
echo "</PRE>";
?>
```

> **注意**: 第 1 パラメータは PHP 4.3.0 で追加されました。

domdocument_dump_file(), domdocument_html_dump_mem() も参照ください。

DomDocument->get_element_by_id

(No version information available, might be only in CVS)

DomDocument->get_element_by_id — 特定の ID を持つ要素を検索する

説明

domelement **DomDocument->get_element_by_id** (string `$id`)

この関数は domdocument_get_elements_by_tagname() と似ていますが、与えられた ID を持つ要素を検索します。 DOM 標準によれば、これには属性 ID を ID 型として定義する DTD が必要になりますが、現在の実装は単純に "//*[@ID = '%s']" に対する xpath 検索を行っています。これは属性が ID 型でない場合に NULL を返す必要がある DOM 標準に準拠していません。 この動作は固定されているようですので、現在の動作を当てにしないでください。

domdocument_get_elements_by_tagname() も参照ください。

DomDocument->get_elements_by_tagname

(No version information available, might be only in CVS)

DomDocument->get_elements_by_tagname — 文章中に与えられたタグ名を持つノードの配列を返す。 もしくは、見つからない場合は空の配列を返す

説明

array **DomDocument->get_elements_by_tagname** (string $name)

domdocument_add_root() も参照ください。

DomDocument->html_dump_mem

(No version information available, might be only in CVS)

DomDocument->html_dump_mem — HTML として文字列に内部 XML ツリーをダンプする

説明

string **DomDocument->html_dump_mem** (void)

DOM 表現から HTML 文章を生成します。この関数は以下の例のように、 通常スクラッチから新規 DOM 文章を生成した後にコールされます。

Example#1 簡単な HTML 文章のヘッダを生成する

```
<?php

// 文章を生成する
$doc  = domxml_new_doc("1.0");

$root  = $doc->create_element("html");
$root  = $doc->append_child($root);

$head  = $doc->create_element("head");
$head  = $root->append_child($head);

$title  = $doc->create_element("title");
$title  = $head->append_child($title);

$text  = $doc->create_text_node("This  is  the  title");
$text  = $title->append_child($text);

echo  $doc->html_dump_mem();
?>
```

上の例の出力は以下となります。

```
<html><head><title>This  is  the  title</title></head></html>
```

domdocument_dump_file(), **domdocument_html_dump_mem()** も参照ください。

DomDocument->xinclude

(No version information available, might be only in CVS)

DomDocument->xinclude — DomDocument オブジェクトにおける XIncludes の代替

説明

int **DomDocument->xinclude** (void)

この関数は DomDocument オブジェクトにおける » XIncludes の代替です。

Example#1 Xincludes の代替

```
<?php

// include.xml は以下の内容を含む :
//  <child>test</child>

$xml = '<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
   <xi:include href="include.xml">
     <xi:fallback>
        <error>xinclude: include.xml not found</error>
     </xi:fallback>
   </xi:include>
</root>';

$domxml = domxml_open_mem($xml);
$domxml->xinclude();

echo $domxml->dump_mem();

?>
```

上の例の出力は以下となります。

```
<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
   <child>test</child>
</root>
```

もし *include.xml* が存在しない場合、 次の通りとなります:

```
<?xml version="1.0"?>
<root xmlns:xi="http://www.w3.org/2001/XInclude">
   <error>xinclude:dom.xml not found</error>
</root>
```

DomDocumentType->entities()

(No version information available, might be only in CVS)

DomDocumentType->entities() — エンティティの一覧を返す

説明

DomDocumentType
array **entities** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

PHP 5 への移行

DOMDocumentType オブジェクトの entities プロパティを使用してください。

DomDocumentType->internal_subset()

(No version information available, might be only in CVS)

DomDocumentType->internal_subset() — 内部サブセットを返す

説明

DomDocumentType
bool **internal_subset** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

PHP 5 への移行

DOMDocumentType オブジェクトの internalSubset プロパティを使用してください。

DomDocumentType->name()

(No version information available, might be only in CVS)

DomDocumentType->name() — 文章型の名前を返す

説明

DomDocumentType
string **name** (void)

この関数は文章型の名前を返します。

返り値

DomDocumentType の名前を文字列として返します。

例

Example#1 文章型の名前を取得する

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
     echo "Error while parsing the document¥n";
     exit;
}

$doctype = $dom->doctype();
echo $doctype->name(); // chapter

?>
```

PHP 5 への移行

DOMDocumentType オブジェクトの name プロパティを使用してください。

DomDocumentType->notations()

(No version information available, might be only in CVS)

DomDocumentType->notations() — ノーテーションの一覧を返す

説明

DomDocumentType
array **notations** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

PHP 5 への移行

DOMDocumentType オブジェクトの notations プロパティを使用してください。

DomDocumentType->public_id()

(No version information available, might be only in CVS)

DomDocumentType->public_id() — 文章型の PUBLIC ID を返す

説明

DomDocumentType
string **PUBLIC_id** (void)

この関数は、文章型の PUBLIC ID を返します。

返り値

DomDocumentType の PUBLIC ID を文字列として返します。

例

以下の例は何も出力しません。

Example#1 PUBLIC ID を取得する

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$doctype = $dom->doctype();
echo $doctype->public_id();
?>
```

PHP 5 への移行

DOMDocumentType オブジェクトの publicId プロパティを使用してください。

DomDocumentType->system_id()

(No version information available, might be only in CVS)

DomDocumentType->system_id() — 文章型のSYSTEM ID を返す

説明

DomDocumentType
string **system_id** (void)

文章型のSYSTEM ID を返します。

返り値

DomDocumentType のSYSTEM ID を文字列として返します。

例

Example#1 SYSTEM ID を取得する

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$doctype = $dom->doctype();
echo $doctype->system_id();
?>
```

上の例の出力は以下となります。

```
/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd
```

PHP 5 への移行

DOMDocumentType オブジェクトの systemId プロパティを使用してください。

DomElement->get_attribute_node()

(No version information available, might be only in CVS)

DomElement->get_attribute_node() — 与えられた属性のノードを返す

説明

DomElement
DomAttribute **get_attribute_node** (string $name)

現在の要素中の与えられた属性のノードを返す Returns the node of the given attribute in the current element.

パラメータ

name

検索する属性の名前。このパラメータは大文字小文字を区別します。

返り値

属性のノードを DomAttribute として返す、 もしくは与えられた名前 *name* を持つ属性がない場合、**FALSE** が返されます。

例

Example#1 属性ノードを取得する

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
     echo "Error while parsing the document¥n";
     exit;
}

$root = $dom->document_element();
if ($attribute = $root->get_attribute_node('language')) {
     echo 'Language is: ' . $attribute->value() . "¥n";
}

?>
```

PHP 5 への移行

DOMElement->getAttributeNode() を使用してください。

参考

- DomElement->get_attribute()
- DomElement->set_attribute()

DomElement->get_attribute()

(No version information available, might be only in CVS)

DomElement->get_attribute() — 与えられた属性の値を返す

説明

DomElement
string **get_attribute** (string $name)

現在の要素中の与えられた属性の値を返します。

PHP 4.3 以降、与えられた名前 *name* を持つ属性がない場合、空文字列が返されます。

パラメータ

name

検索する属性の名前。このパラメータは大文字小文字を区別します。

返り値

属性の名前を文字列として返す、もしくは与えられた名前 *name* を持つ属性がない場合、空文字列が返されます。

例

Example#1 属性の値を取得する

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
     echo "Error while parsing the document¥n";
     exit;
}

// chapter を取得する
$root = $dom->document_element();
echo $root->get_attribute('language'); // en

?>
```

PHP 5 への移行

DOMElement->getAttribute() を使用してください。

参考

- DomElement->get_attribute_node()
- DomElement->set_attribute()

DomElement->get_elements_by_tagname()

(No version information available, might be only in CVS)

DomElement->get_elements_by_tagname() — タグ名により要素を取得する

説明

DomElement
array **get_elements_by_tagname** (string $name)

指定した名前 *name* を持つ現在の要素以下にある全ての子要素を取得します。

パラメータ

name

検索する属性の名前

返り値

DomElement オブジェクトの配列を返します。

例

Example#1 内容を取得する

```
<?php
if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$root = $dom->document_element();

$node_array = $root->get_elements_by_tagname('element');

foreach ($node_array as $node) {
     echo ' - ' . $node->get_content() . "¥n";
}

?>
```

PHP 5 への移行

DOMElement->getElementsByTagName() を使用してください。

DomElement->has_attribute()

(No version information available, might be only in CVS)

DomElement->has_attribute() — 現在のノードに属性があるかどうかを調べる

説明

DomElement
bool **has_attribute** (string $name)

この関数は、現在のノードに名前 *name* を持つ属性があるかどうかを調べます。

パラメータ

name

検査する属性名

返り値

もし問い合わせた属性が存在する場合 **TRUE**、そうでない場合 **FALSE** を返します。

例

Example#1 属性の存在を調べる

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
     echo "Error while parsing the document¥n";
     exit;
}

$root = $dom->document_element();

$buffer = '<html';
if ($root->has_attribute('language')) {
    $buffer .= 'lang="' . $root->get_attribute('language') . '"';
}
$buffer .= '>';

?>
```

PHP 5 への移行

DOMElement->hasAttribute() を使用してください。

DomElement->remove_attribute()

(No version information available, might be only in CVS)

DomElement->remove_attribute() — 属性を削除する

説明

DomElement
bool **remove_attribute** (string $name)

現在の DomElement ノードから属性を削除します。

パラメータ

name

削除する属性名

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PHP 5 への移行

DOMElement->removeAttribute() を使用してください。

DomElement->set_attribute_node()

(No version information available, might be only in CVS)

DomElement->set_attribute_node() — 新規属性を追加する

説明

DomElement
DomNode **set_attribute_node** (DomNode $attr)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DomElement->set_attribute()

(No version information available, might be only in CVS)

DomElement->set_attribute() — 属性値を設定する

説明

DomElement
DomAttribute **set_attribute** (string $name , string $value)

名前 *name* を持つ属性値を *value* に設定します。

パラメータ

name

属性名。もし属性が存在しない場合、新たに作成されます。

value

属性値

返り値

古い DomAttribute ノード、 もしくは最初に属性が作成された場合は新しいノードを返します。

例

Example#1 属性を設定する

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

PHP 5 への移行

DOMElement->setAttribute() を使用してください。

参考

- DomElement->get_attribute_node()

- DomElement->get_attribute()

DomElement->tagname()

(No version information available, might be only in CVS)

DomElement->tagname() — 現在の要素名を返す

説明

DomElement
string **tagname** (void)

現在のノードの名前を返します。この関数をコールすることは、 tagname プロパティにアクセスする、 もしくは現在のノードに対して DomNode->node_name をコールすることと等価です。

返り値

現在の DomElement ノードの名前を返します。

例

Example#1 ノード名を取得する

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
     echo "Error while parsing the document¥n";
     exit;
}

$root = $dom->document_element();
echo $root->tagname();    // chapter
echo $root->tagname;      // chapter
echo $root->node_name(); // chapter


?>
```

PHP 5 への移行

DOMElement オブジェクトの tagName プロパティを使用してください。

DomNode->add_namespace

(No version information available, might be only in CVS)

DomNode->add_namespace — ノードに名前空間宣言を追加する

説明

DOMNode
bool **add_namespace** (string `$uri` , string `$prefix`)

このメソッドはノードに名前空間宣言を追加します。

> **注意**: このメソッドは DOM 規格の一部ではありません。

パラメータ

`uri`

ノードの名前空間 URI

`prefix`

ノードの名前空間の接頭辞

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PHP 5 への移行

DOMElement の名前空間 URI と接頭辞を設定する、 もしくは生成時に DOMDocument->createElementNS() か DOMDocument->createAttributeNS() を使用して DOMAttr を設定してください。

> **注意**: 属性は追加される要素からの名前空間を継承しないことを覚えておいてください。

参考

- DomDocument->create_element_ns
- DomNode->set_namespace

DomNode->append_child

(No version information available, might be only in CVS)

DomNode->append_child — 子ノードの最後に新規に子ノードを追加する

説明

DOMNode
DOMNode **append_child** (DOMNode $newnode)

この関数は既存の子ノードに子ノードを追加する、 もしくは新規子ノードを作成します。

パラメータ

newnode

追加するノード。これには、例えば DomDocument->create_element, DomDocument->create_text_node など、もしくは単純にあらゆる他のノードによって作成されたノードを 指定することが可能です。

> **注意**: このメソッドを使用して DOMAttribute を追加することはできません。代わりに DomElement->set_attribute() を使用してください。

返り値

成功時は追加されたノード、失敗時は **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	他の文章からのノードを挿入することは、もはや許可されません。
4.3.0	PHP 4.3.0 以前では、新規子ノードは追加する前に複製されます。 そのため、新規子ノードは完全に新しいコピーとなります。これは、 この関数に渡されたノードを変更することなしに修正することが可能です。 渡されたノードが子ノード自身の場合うまくコピーされますので、XML 文章の大きな部位を簡単にコピーすることができます。 戻り値は追加された子ノードです。 もし追加された子ノードを変更するつもりであれば、 返されたノードを使用する必要があります。
4.3.0 と 4.3.1	新規子ノード *newnode* がすでに DomNode の子ノードである場合、最初に既存のコンテキストから削除されます。 そのため、*newnode* は移動され、 コピーされません。この動作は W3C 規格に準じるものです。 もし古い動作をさせる必要がある場合、追加ではなく DomNode->clone_node を使用してください。
4.3.2	新規子ノード *newnode* がすでにツリーにある場合、 最初に既存のコンテキストから削除されます。同じ規則が適用されます。

例

以下の例は新規ドキュメントに新規要素ノードを追加し、 属性 *align* を *left* に設定します。

Example#1 子ノードを追加する

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

上記の例は、以下のようにも書くことができました:

Example#2 子ノードを追加する

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node->set_attribute("align", "left");
$newnode = $doc->append_child($node);
?>
```

より複雑な例は以下の通りです。最初に特定の要素を検索し、 子を含むノードを複製し、兄弟ノードとして追加します。 最後に新規属性を新規兄弟ノード能古ノードの一つに追加し、 文章全体をダンプします。

Example#3 子ノードを追加する

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$parent = $element->parent_node();
$newnode = $parent->append_child($element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
?>
```

上記の例は DomNode->append_child の代わりに DomNode->insert_before を用いても動作します。

PHP 5 への移行

DOMNode->appendChild() を使用してください。

参考

- DomNode->insert_before
- DomNode->clone_node

DomNode->append_sibling

(No version information available, might be only in CVS)

DomNode->append_sibling — 新規に兄弟をノードに追加する

説明

domelement **DomNode->append_sibling** (domelement $newnode)

この関数は既存のノードに兄弟ノードを追加します。 子ノードは、例えば DomDocument->create_element, DomDocument->create_text_node など、もしくは単純にあらゆる他のノードによって作成されたノードを 指定することが可能です。

新規兄弟ノードは追加する前に最初に複製されます。 そのため、新規子ノードは完全に新しいコピーとなります。これは、 この関数に渡されたノードを変更することなしに修正することが可能です。 渡されたノードが子ノード自身の場合うまくコピーされますので、 XML 文章の大きな部位を簡単にコピーすることができます。 戻り値は追加された子ノードです。 もし追加された子ノードを変更するつもりであれば、 返されたノードを使用する必要があります。

この関数は domnode_append_child() のど宇佐を提供するために追加され、PHP 4.2 まで動作します。

domnode_append_before() も参照ください。

DomNode->attributes

(No version information available, might be only in CVS)

DomNode->attributes — 属性の一覧を返す

説明

array **DomNode->attributes** (void)

この関数は、ノードが XML_ELEMENT_NODE 型の場合に属性の配列を返すだけです。

(PHP >= 4.3 のみ) 属性が存在しない場合、NULL が返されます。

DomNode->child_nodes

(No version information available, might be only in CVS)

DomNode->child_nodes — 子ノードを返す

説明

array **DomNode->child_nodes** (void)

全ての子ノードを返します。

domnode_next_sibling(), domnode_previous_sibling() も参照ください。

DomNode->clone_node

(No version information available, might be only in CVS)

DomNode->clone_node — ノードを複製する

説明

domelement **DomNode->clone_node** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DomNode->dump_node

(No version information available, might be only in CVS)

DomNode->dump_node — 単一ノードをダンプする

説明

string **DomNode->dump_node** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

domdocument_dump_mem() も参照ください。

DomNode->first_child

(No version information available, might be only in CVS)

DomNode->first_child — 最初の子ノードを返す

説明

domelement **DomNode->first_child** (void)

最初の子ノードを返します。

(PHP >= 4.3 のみ) 最初の子ノードが存在しない場合、NULL が返されます。

domnode_last_child(), domnode_next_sibling(), domnode_previous_sibling() も参照ください。

DomNode->get_content

(No version information available, might be only in CVS)

DomNode->get_content — ノードの内容を取得する

説明

string **DomNode->get_content** (void)

この関数は実ノードの内容を返します。

Example#1 内容を取得する

```
<?php
if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$root = $dom->document_element();

$node_array = $root->get_elements_by_tagname("element");

for ($i = 0; $i<count($node_array); $i++) {
    $node = $node_array[$i];
    echo "The element[$i] is: " . $node->get_content();
}

?>
```

DomNode->has_attributes

(No version information available, might be only in CVS)

DomNode->has_attributes — ノードが属性を有しているかを調べる

説明

bool **DomNode->has_attributes** (void)

この関数は、ノードが属性を有しているかを調べます。

domnode_has_child_nodes() も参照ください。

DomNode->has_child_nodes

(No version information available, might be only in CVS)

DomNode->has_child_nodes — ノードが子ノードを有しているかを調べる

説明

bool **DomNode->has_child_nodes** (void)

この関数は、ノードが子ノードを有しているかを調べます。

domnode_child_nodes() も参照ください。

DomNode->insert_before

(No version information available, might be only in CVS)

DomNode->insert_before — 新規ノードを子ノードとして挿入する

説明

domelement **DomNode->insert_before** (domelement $newnode , domelement $refnode)

この関数は新規ノード *newnode* をノード *refnode* の直前に挿入します。 戻り値は挿入されたノードです。 もし追加された子ノードを変更するつもりであれば、 返されたノードを使用する必要があります。

(PHP >= 4.3 のみ) *newnode* がすでに文章の一部である場合、最初に既存のコンテキストから削除されます。 もし *refnode* が NULL の場合、*newnode* は子ノードリストの最後に挿入されます。

domnode_insert_before() は domnode_append_child() に非常に似ており、 以下の例は domnode_append_child() にある例と同様のことを行うことを示しています。

Example#1 子ノードを追加する

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$newnode = $element->insert_before($element, $element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<pre>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</pre>";
?>
```

domnode_append_child() も参照ください。

DomNode->is_blank_node

(No version information available, might be only in CVS)

DomNode->is_blank_node — ノードが空かどうかを調べる

説明

bool **DomNode->is_blank_node** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DomNode->last_child

(No version information available, might be only in CVS)

DomNode->last_child — 最後の子ノードを返す

説明

domelement **DomNode->last_child** (void)

最後の子ノードを返します。

(PHP >= 4.3 のみ) 最後の子ノードが存在しない場合、NULL が返されます。

domnode_first_child(), domnode_next_sibling(), domnode_previous_sibling() も参照ください。

DomNode->next_sibling

(No version information available, might be only in CVS)

DomNode->next_sibling — 次の兄弟ノードを返す

説明

domelement **DomNode->next_sibling** (void)

この関数は現在のノードの次の兄弟ノードを返します。 次の兄弟ノードが存在しない場合、**FALSE** (< 4.3) もしくは null (>= 4.3) を返します。この関数を使用することで、 例に示されたようにノードの全ての子コードを走査する事ができます。

Example#1 子ノードを走査する

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$child = $element->first_child();

while ($child) {
   print_r($child);
   $child = $child->next_sibling();
}
?>
```

domnode_previous_sibling() も参照ください。

DomNode->node_name

(No version information available, might be only in CVS)

DomNode->node_name — ノード名を返す

説明

string **DomNode->node_name** (void)

ノード名を返します。名前は以下の表の通り、 異なるノードの型に対しては異なる意味を持ちます。

値の意味

型	意味
DomAttribute	属性値
DomAttribute	
DomCDataSection	#cdata セクション
DomComment	#comment
DomDocument	#document
DomDocumentType	文章型名
DomElement	タグ名
DomEntity	エンティティ名
DomEntityReference	エンティティ参照名
DomNotation	ノーテーション名
DomProcessingInstruction	ターゲット
DomText	#text

DomNode->node_type

(No version information available, might be only in CVS)

DomNode->node_type — ノードの型を返す

説明

int **DomNode->node_type** (void)

ノードの型を返します。全ての取りうる型は、 導入のページにある表に一覧されています。

```
<?php

include 'example.inc';

$dom = domxml_open_mem($xmlstr);

$chapter = $dom->document_element();

// 章に含まれる要素を見てみよう
foreach($chapter->child_nodes() as $node) {
  if ($node->node_type() == XML_ELEMENT_NODE) {
    echo $node->node_name() . "¥n";
  }
}

?>
```

上の例の出力は以下となります。

```
title
para
```

DomNode->node_value

(No version information available, might be only in CVS)

DomNode->node_value — ノードの値を返す

説明

string **DomNode->node_value** (void)

ノードの値を返します。値は以下の表の通り、 異なるノードの型に対しては異なる意味を持ちます。

値の意味

型	意味
DomAttribute	属性値
DomAttribute	
DomCDataSection	内容
DomComment	コメントの内容
DomDocument	null
DomDocumentType	null
DomElement	null
DomEntity	null
DomEntityReference	null
DomNotation	null
DomProcessingInstruction	ターゲット以外の全ての内容
DomText	テキストの内容

DomNode->owner_document

(No version information available, might be only in CVS)

DomNode->owner_document — このノードが属する文章を返す

説明

domdocument **DomNode->owner_document** (void)

この関数は現在のノードが属している文章を返します。

以下の例は 2 つの等しい子ノードリストを作成します。

Example#1 ノードの文章を検出する

```
<?php
$doc  = domxml_new_doc("1.0");
$node  = $doc->create_element("para");
$node  = $doc->append_child($node);
$children  = $doc->children();
print_r($children);

$doc2  = $node->owner_document();
$children  = $doc2->children();
print_r($children);
?>
```

domnode_insert_before() も参照ください。

DomNode->parent_node

(No version information available, might be only in CVS)

DomNode->parent_node — 親ノードを返す

説明

domnode **DomNode->parent_node** (void)

この関数は親ノードを返します。

(PHP >= 4.3 のみ) 親ノードが存在しない場合、NULL が返されます。

以下の例は 2 つの等しい子ノードリストを作成します。

Example#1 ノードの文章を検出する

```
<?php
$doc  = domxml_new_doc("1.0");
$node  = $doc->create_element("para");
$node  = $doc->append_child($node);
$children  = $doc->children();
print_r($children);

$doc2  = $node->parent_node();
$children  = $doc2->children();
print_r($children);
?>
```

DomNode->prefix

(No version information available, might be only in CVS)

DomNode->prefix — ノードの名前空間接頭辞を返す

説明

string **DomNode->prefix** (void)

ノードの名前空間接頭辞を返します。

DomNode->previous_sibling

(No version information available, might be only in CVS)

DomNode->previous_sibling — 前の兄弟ノードを返す

説明

domelement **DomNode->previous_sibling** (void)

この関数は現在のノードの前の兄弟ノードを返します。 前の兄弟ノードが存在しない場合、**FALSE** (< 4.3) もしくは **NULL** (>= 4.3) を返します。この関数を使用することで、 例に示されたようにノードの全ての子コードを走査する事ができます。

domnode_next_sibling() も参照ください。

DomNode->remove_child

(No version information available, might be only in CVS)

DomNode->remove_child — 子ノードのリストから子ノードを削除する

説明

domtext **DomNode->remove_child** (domtext $oldchild)

この関数は、子ノードのリストから子ノードを削除します。 もし子ノードが削除できなかった、もしくは子ノードではない場合、この関数は **FALSE** を返します。子ノードが削除できた場合、 この関数は古い子ノードを返します。

Example#1 子ノードを削除する

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$children = $element->child_nodes();
$child = $element->remove_child($children[0]);

echo "<PRE>";
$xmlfile = $dom->dump_mem(true);
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

domnode_append_child() も参照ください。

DomNode->replace_child

(No version information available, might be only in CVS)

DomNode->replace_child — 子ノードを置換する

説明

domelement **DomNode->replace_child** (domelement $newnode , domelement $oldnode)

(PHP 4.2) この関数は子ノード *oldnode* を渡された新規ノードに置換します。もし新規ノードがすでに子ノードの場合、 二度目は追加されません。もし古いノードが存在しない場合、 この関数は **FALSE** を返します。もし置換が成功した場合、 古いノードが返されます。

(PHP 4.3) この関数は新規ノードがすでに DomNode の子ノードだったとしても、 子ノード *oldnode* を渡された *newnode* に置換します。 もし *newnode* ガスでに文章に挿入されている場合、 最初に既存のコンテキストから削除されます。 もし古いノードが存在しない場合、この関数は **FALSE** を返し、 置換が成功した場合は古いノードが返されます (この動作は W3C 規格に準拠しています。)

domnode_append_child() も参照ください。

DomNode->replace_node

(No version information available, might be only in CVS)

DomNode->replace_node — ノードを置換する

説明

domelement **DomNode->replace_node** (domelement $newnode)

(PHP 4.2) この関数は既存のノードを渡された新規ノードに置換します。 文章にすでに存在しているノードが二度目に挿入されないようにするため、 *newnode* は置換前にコピーされます。 この動作は、置換前にノードに対して行う全ての修正、もしくは domnode_first_child(), domnode_child_nodes() などの関数を用いた後に挿入されたノードを再取得させることを強制します。

(PHP 4.3) この関数は既存ノードを渡された新規ノードに置換します。 もはやコピーはされません。もし *newnode* が文章に挿入されている場合、最初に既存のコンテキストから削除されます。 もし置換が成功した場合、古いノードが返されます。

domnode_append_child() も参照ください。

DomNode->set_content

(No version information available, might be only in CVS)

DomNode->set_content — ノードの内容を設定する

説明

bool **DomNode->set_content** (string $content)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DomNode->set_name

(No version information available, might be only in CVS)

DomNode->set_name — ノード名を設定する

説明

bool **DomNode->set_name** (void)

ノード名を設定します。

domnode_node_name() も参照ください。

DomNode->set_namespace

(No version information available, might be only in CVS)

DomNode->set_namespace — ノードの名前空間を設定する

説明

void **DomNode->set_namespace** (string $uri [, string $prefix])

ノードの名前空間を *uri* に設定します。 もしこのノードの親ノードの一つに同じ URI で名前空間が宣言されている場合、 名前空間の接頭辞が使用されます。そうでない場合、オプションパラメータ *prefix* で指定された接頭辞、 もしくはランダムに設定された接頭辞が使用されます。

domdocument_create_element_ns(), domnode_add_namespace() も参照ください。

DomNode->unlink_node

(No version information available, might be only in CVS)

DomNode->unlink_node — ノードを削除する

説明

void **DomNode->unlink_node** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DomProcessingInstruction->data

(No version information available, might be only in CVS)

DomProcessingInstruction->data — PI ノードのデータを返す

説明

DomProcessingInstruction
string **data** (void)

このメソッドは PI ノードのデータを取得します。

返り値

PI のデータを返します。

PHP 5 への移行

DOMProcessingInstruction の data プロパティを使用してください。

DomProcessingInstruction->target

(No version information available, might be only in CVS)

DomProcessingInstruction->target — PI ノードのターゲットを返す

説明

DomProcessingInstruction
string **target** (void)

このメソッドは PI ノードのターゲットを取得します。

返り値

PI ノードのターゲットを返します。

PHP 5 への移行

DOMProcessingInstruction の target プロパティを使用してください。

DomXsltStylesheet->process()

(No version information available, might be only in CVS)

DomXsltStylesheet->process() — DomDocument オブジェクトに XSLT 変換を適用する

説明

DomXsltStylesheet
DomDocument **process** (DomDocument `$xml_doc` [, array `$xslt_params` [, bool `$is_xpath_param` [, string `$profile_filename`]]])

与えられた DomDocument オブジェクトに XSLT 変換を適用します。

パラメータ

`xml_doc`

DomDocument オブジェクトとしての変換される XML 文章

xslt_params

パラメータ名と値のペアを持つ連想配列

is_xpath_param

FALSE に設定された場合、*xslt_params* の値は句オーとされます。これはデフォルトの動作です。 PHP 文字列として値を渡すことができます。

注意: もし文字列がシングルクォートとダブルクォートの両方を含んでいる場合、 このパラメータを **TRUE** に設定し、 ご自身で全ての値のクォーティングを行う必要があります。

profile_filename

プロファイリング情報を希望する場合、ファイル名のパスを設定します。

返り値

処理結果を DomDocument オブジェクトとして返します。

PHP 5 への移行

XSLTProcessor::setParameter と XSLTProcessor::transformToDoc を使用してください。

変更履歴

バージョン	説明
4.3.0	*profile_filename* パラメータが追加されました。

参考

- domxml_xslt_stylesheet()
- domxml_xslt_stylesheet_file()
- domxml_xslt_stylesheet_doc()

DomXsltStylesheet->result_dump_file()

(No version information available, might be only in CVS)

DomXsltStylesheet->result_dump_file() — XSLT 変換の結果をファイルにダンプする

説明

DomXsltStylesheet
string **result_dump_file** (DomDocument `$xmldoc` , string `$filename`)

DomXsltStylesheet->process() は、たとえ出力方式が

```
<xsl:output>
```

や同様の属性/要素であっても常に整形式の XML DomDocument を返し、HTML 4 やテキストデータを出力したい場合には あまり使用されません。

これに反して、この関数は

```
<xsl:output  method="html|text">
```

や他の出力制御ディレクティブを受け付けます。 使用法の説明については例を参照ください。

例

Example#1 XSLT 変換の結果をファイルに保存する

```php
<?php
$filename  = "stylesheet.xsl";
$xmldoc  = domxml_open_file("data.xml");
$xsldoc  = domxml_xslt_stylesheet_file($filename);
$result  =  $xsldoc->process($xmldoc);
echo $xsldoc->result_dump_file($result, "filename");
?>
```

参考

- DomXsltStylesheet->result_dump_mem()
- DomXsltStylesheet->process()

DomXsltStylesheet->result_dump_mem()

(No version information available, might be only in CVS)

DomXsltStylesheet->result_dump_mem() — XSLT 変換の結果を文字列にダンプする

説明

DomXsltStylesheet
string **result_dump_mem** (DomDocument `$xmldoc`)

DomXsltStylesheet->process() は、たとえ出力方式が

```
<xsl:output>
```

や同様の属性/要素であっても常に整形式の XML DomDocument を返し、HTML 4 やテキストデータを出力したい場合には あまり使用されません。

これに反して、この関数は

```
<xsl:output  method="html|text">
```

や他の出力制御ディレクティブを受け付けます。 使用法の説明については例を参照ください。

例

Example#1 XSLT 変換の結果を出力する

```
<?php
$filename  = "stylesheet.xsl";
$xmldoc  = domxml_open_file("data.xml");
$xsldoc  = domxml_xslt_stylesheet_file($filename);
$result  =  $xsldoc->process($xmldoc);
echo $xsldoc->result_dump_mem($result);
?>
```

参考

- DomXsltStylesheet->result_dump_file()
- DomXsltStylesheet->process()

---

domxml_new_doc

(PHP 4 >= 4.2.0)

domxml_new_doc — 空の新規 XMLドキュメントを作成する

説明

DomDocument **domxml_new_doc** (string `$version`)

スクラッチから新規 DOM ドキュメントを作成し、返します。

パラメータ

`version`

文章の XML バージョン番号

返り値

新規 DomDocument インスタンスを返します。

---

domxml_open_file

(PHP 4 >= 4.2.0)

domxml_open_file — XML ファイルから DOM オブジェクトを作成する

説明

DomDocument **domxml_open_file** (string $filename [, int $mode [, array &$error]])

この関数は、ファイルで与えられた XML 文章をパースします。

パラメータ

filename

XML ファイルへのパス。ファイルは読み込み専用モードでアクセスされます。

mode

このオプションパラメータにより、この関数の動作を変更する事が可能です。

以下の定数の 1 つを使用することが可能です: **DOMXML_LOAD_PARSING** (デフォルト), **DOMXML_LOAD_VALIDATING** もしくは **DOMXML_LOAD_RECOVERING** 。 ビット OR により **DOMXML_LOAD_DONT_KEEP_BLANKS**, **DOMXML_LOAD_SUBSTITUTE_ENTITIES** と **DOMXML_LOAD_COMPLETE_ATTRS** も追加することが可能です。

error

指定された場合、エラーメッセージが代入されます。 *error* は リファレンス として渡す必要があります。

返り値

与えられたファイルの DomDocument インスタンスを返します。

例

Example#1 ファイルから XML 文章をオープンする

```
<?php

if (!$dom = domxml_open_file("example.xml")) {
   echo "Error while parsing the document¥n";
   exit;
}

$root = $dom->document_element();
?>
```

変更履歴

バージョン	説明
4.3.0	パラメータ *mode* と *error* が追加されました。

参考

- domxml_open_mem()
- domxml_new_doc()

---

domxml_open_mem

(PHP 4 >= 4.2.0)

domxml_open_mem — XML 文章から DOM オブジェクトを作成する

説明

DomDocument **domxml_open_mem** (string $str [, int $mode [, array &$error]])

この関数は、文字列で与えられた XML 文章をパースします。

パラメータ

str

XML ファイルの内容

mode

このオプションパラメータにより、この関数の動作を変更する事が可能です。

以下の定数の 1 つを使用することが可能です: **DOMXML_LOAD_PARSING** (デフォルト), **DOMXML_LOAD_VALIDATING** もしくは **DOMXML_LOAD_RECOVERING** 。 ビット OR により **DOMXML_LOAD_DONT_KEEP_BLANKS**, **DOMXML_LOAD_SUBSTITUTE_ENTITIES** と **DOMXML_LOAD_COMPLETE_ATTRS** も追加することが可能です。

error

指定された場合、エラーメッセージが代入されます。 *error* は リファレンス として渡す必要があります。

返り値

与えられた XML コンテンツの DomDocument インスタンスを返します。

変更履歴

バージョン	説明
4.3.0	パラメータ *mode* および *error* が追加されました。

例

Example#1 文字列での XML 文章をオープンする

```
<?php
include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
   echo "Error while parsing the document¥n";
   exit;
}

$root = $dom->document_element();
?>
```

参考

- domxml_open_file()
- domxml_new_doc()

---

domxml_version

(PHP 4 >= 4.0.7)

domxml_version — XML ライブラリのバージョンを取得する

説明

string **domxml_version** (void)

この関数は、現在使用されている XML ライブラリのバージョンを取得します。

返り値

XML ライブラリのバージョンを文字列として返します。

例

Example#1 domxml_version() の例

```
<?php

echo domxml_version();

?>
```

上の例の出力は、たとえば以下のようになります。

```
20607
```

---

domxml_xmltree

(PHP 4 >= 4.2.0)

domxml_xmltree — XML 文章から PHP オブジェクトツリーを作成する

説明

DomDocument **domxml_xmltree** (string `$str`)

この関数は、*`str`* の XML ドキュメントをパースし、 パースされた文章としてPHPオブジェクトのツリーを返します。

他の関数はこのツリーにはアクセスできないため、 この関数は他の関数と異なっています。 例えばノードを追加する時のように、このツリーを修正することには、 現在 XML ファイルとしてツリーをダンプする手段がないため意味がありません。

しかしながら、この関数はファイルを読み込んで内容を調べたい場合には有用です。

パラメータ

`str`

XML ファイルの内容

返り値

DomDocument によって開始される DOM オブジェクトのツリーを返します。

---

domxml_xslt_stylesheet_doc

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet_doc — DomDocument オブジェクトから DomXsltStylesheet オブジェクトを作成する

説明

DomXsltStylesheet **domxml_xslt_stylesheet_doc** (DomDocument `$xsl_doc`)

与えられた XSL 文章から DomXsltStylesheet オブジェクトを作成します。

パラメータ

`xsl_doc`

DomDocument オブジェクトとしての XSL 文章。

返り値

DomXsltStylesheet の新規インスタンスを返します。

PHP 5 への移行

`xsl_doc` パラメータ付きで XSLTProcessor::importStylesheet をコールしてください。

参考

- DomXsltStylesheet->process()
- domxml_xslt_stylesheet()
- domxml_xslt_stylesheet_file()

---

domxml_xslt_stylesheet_file

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet_file — ファイル中の XSL 文章から DomXsltStylesheet オブジェクトを作成する

説明

DomXsltStylesheet **domxml_xslt_stylesheet_file** (string `$xsl_file`)

与えられた XSL ファイルから DomXsltStylesheet オブジェクトを作成します。

パラメータ

`xsl_file`

文字列としての XSL 文章へのパス

返り値

DomXsltStylesheet の新規インスタンスを返します。

PHP 5 への移行

パラメータとして *DOMDocument::load($xsl_file)* を渡して XSLTProcessor::importStylesheet をコールしてください。

参考

- DomXsltStylesheet->process()
- domxml_xslt_stylesheet()
- domxml_xslt_stylesheet_doc()

---

domxml_xslt_stylesheet

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet — 文字列での XSL 文章から DomXsltStylesheet オブジェクトを作成する

説明

DomXsltStylesheet **domxml_xslt_stylesheet** (string `$xsl_buf`)

与えられた XSL バッファから DomXsltStylesheet オブジェクトを作成します。

パラメータ

`xsl_buf`

文字列としての XSL 文章

返り値

DomXsltStylesheet の新規インスタンスを返します。

PHP 5 への移行

パラメータとして *DOMDocument::loadXML($xsl_buf)* を渡して XSLTProcessor::importStylesheet をコールしてください。

参考

- DomXsltStylesheet->process()
- domxml_xslt_stylesheet_file()
- domxml_xslt_stylesheet_doc()

---

domxml_xslt_version

(PHP 4 >= 4.2.0)

domxml_xslt_version — XSLT ライブラリのバージョンを取得する

説明

int **domxml_xslt_version** (void)

XSLT ライブラリのバージョンを取得します。

返り値

XSLT ライブラリのバージョン番号を整数で返します。

例

Example#1 domxml_xslt_version() の例

```
<?php

echo domxml_xslt_version();

?>
```

上の例の出力は、たとえば以下のようになります。

```
10112
```

参考

- domxml_version()

---

xpath_eval_expression

(PHP 4 >= 4.0.4)

xpath_eval_expression — 与えられた文字列で XPath のロケーションパスを評価する

説明

XPathContext
XPathObject **xpath_eval_expression** (string `$expression` [, domnode `$contextnode`])
XPathObject **xpath_eval_expression** (XPathContext `$xpath_context` , string `$expression` [, domnode `$contextnode`])

Example#1 xpath_eval_expression() の例

```
<?php

include("example.inc");

if (!$dom = domxml_open_mem($xmlstr)) {
     echo "Error while parsing the document¥n";
     exit;
}

$xpath = xpath_new_context($dom);
var_dump(xpath_eval_expression($xpath, '/chapter/@language'));

?>
```

上の例の出力は以下となります。

```
object(XPathObject)(2)  {
    ["type"]=>
    int(1)
    ["nodeset"]=>
    array(1) {
      [0]=>
      object(domattribute)(5) {
        ["type"]=>
        int(2)
        ["name"]=>
        string(8) "language"
        ["value"]=>
        string(2) "en"
        [0]=>
        int(7)
        [1]=>
```

```
        int(138004256)
    }
  }
}
```

xpath_eval() も参照ください。

---

xpath_eval

(PHP 4 >= 4.0.4)

xpath_eval — 与えられた文字列で XPath のロケーションパスを評価する

説明

XPathContext
XPathObject **xpath_eval** (string `$xpath_expression` [, domnode `$contextnode`])
XPathObject **xpath_eval** (XPathContext `$xpath_context` , string `$xpath_expression` [, domnode `$contextnode`])

オプション *contextnode* は、相対 XPath クエリを実行するために指定する事が可能です。

xpath_new_context() も参照ください。

---

xpath_new_context

(PHP 4 >= 4.0.4)

xpath_new_context — 新規 xpath コンテキストを作成する

説明

XPathContext **xpath_new_context** (domdocument `$dom_document`)

xpath_eval() も参照ください。

---

xpath_register_ns_auto

(No version information available, might be only in CVS)

xpath_register_ns_auto — 与えられた XPath コンテキストに与えられた名前空間を登録する

説明

bool **xpath_register_ns_auto** (XPathContext `$xpath_context` [, object `$context_node`])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- xpath_register_ns()

---

xpath_register_ns

(PHP 4 >= 4.2.0)

xpath_register_ns — 与えられた XPath コンテキストに与えられた名前空間を登録する

説明

bool **xpath_register_ns** (XPathContext `$xpath_context` , string `$prefix` , string `$uri`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- xpath_register_ns_auto()

---

xptr_eval

(PHP 4 >= 4.0.4)

xptr_eval — 指定した文字列の XPtr ロケーションパスを評価する

説明

XPathContext
int **xptr_eval** (string `$eval_str` [, domnode `$contextnode`])
int **xptr_eval** (XPathContext `$xpath_context` , string `$eval_str` [, domnode `$contextnode`])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xptr_new_context

(PHP 4 >= 4.0.4)

xptr_new_context — 新規 XPath コンテキストを作成する

説明

XPathContext **xptr_new_context** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

- DomDocument->dump_mem — 内部 XML ツリーを文字列にダンプする
- DomDocument->get_element_by_id — 特定の ID を持つ要素を検索する
- DomDocument->get_elements_by_tagname — 文章中に与えられたタグ名を持つノードの配列を返す。 もしくは、見つからない場合は空の配列を返す
- DomDocument->html_dump_mem — HTML として文字列に内部 XML ツリーをダンプする
- DomDocument->xinclude — DomDocument オブジェクトにおける XIncludes の代替
- DomDocumentType->entities() — エンティティの一覧を返す
- DomDocumentType->internal_subset() — 内部サブセットを返す
- DomDocumentType->name() — 文章型の名前を返す
- DomDocumentType->notations() — ノーテーションの一覧を返す
- DomDocumentType->public_id() — 文章型の PUBLIC ID を返す
- DomDocumentType->system_id() — 文章型のSYSTEM ID を返す
- DomElement->get_attribute_node() — 与えられた属性のノードを返す
- DomElement->get_attribute() — 与えられた属性の値を返す
- DomElement->get_elements_by_tagname() — タグ名により要素を取得する
- DomElement->has_attribute() — 現在のノードに属性があるかどうかを調べる
- DomElement->remove_attribute() — 属性を削除する
- DomElement->set_attribute_node() — 新規属性を追加する
- DomElement->set_attribute() — 属性値を設定する
- DomElement->tagname() — 現在の要素名を返す
- DomNode->add_namespace — ノードに名前空間宣言を追加する
- DomNode->append_child — 子ノードの最後に新規に子ノードを追加する
- DomNode->append_sibling — 新規に兄弟をノードに追加する
- DomNode->attributes — 属性の一覧を返す
- DomNode->child_nodes — 子ノードを返す
- DomNode->clone_node — ノードを複製する
- DomNode->dump_node — 単一ノードをダンプする
- DomNode->first_child — 最初の子ノードを返す
- DomNode->get_content — ノードの内容を取得する
- DomNode->has_attributes — ノードが属性を有しているかを調べる
- DomNode->has_child_nodes — ノードが子ノードを有しているかを調べる
- DomNode->insert_before — 新規ノードを子ノードとして挿入する
- DomNode->is_blank_node — ノードが空かどうかを調べる
- DomNode->last_child — 最後の子ノードを返す
- DomNode->next_sibling — 次の兄弟ノードを返す
- DomNode->node_name — ノード名を返す
- DomNode->node_type — ノードの型を返す
- DomNode->node_value — ノードの値を返す
- DomNode->owner_document — このノードが属する文章を返す
- DomNode->parent_node — 親ノードを返す
- DomNode->prefix — ノードの名前空間接頭辞を返す
- DomNode->previous_sibling — 前の兄弟ノードを返す
- DomNode->remove_child — 子ノードのリストから子ノードを削除する
- DomNode->replace_child — 子ノードを置換する
- DomNode->replace_node — ノードを置換する
- DomNode->set_content — ノードの内容を設定する
- DomNode->set_name — ノード名を設定する
- DomNode->set_namespace — ノードの名前空間を設定する
- DomNode->unlink_node — ノードを削除する
- DomProcessingInstruction->data — PI ノードのデータを返す
- DomProcessingInstruction->target — PI ノードのターゲットを返す
- DomXsltStylesheet->process() — DomDocument オブジェクトに XSLT 変換を適用する
- DomXsltStylesheet->result_dump_file() — XSLT 変換の結果をファイルにダンプする
- DomXsltStylesheet->result_dump_mem() — XSLT 変換の結果を文字列にダンプする
- domxml_new_doc — 空の新規 XMLドキュメントを作成する
- domxml_open_file — XML ファイルから DOM オブジェクトを作成する
- domxml_open_mem — XML 文章から DOM オブジェクトを作成する
- domxml_version — XML ライブラリのバージョンを取得する
- domxml_xmltree — XML 文章から PHP オブジェクトツリーを作成する
- domxml_xslt_stylesheet_doc — DomDocument オブジェクトから DomXsltStylesheet オブジェクトを作成する
- domxml_xslt_stylesheet_file — ファイル中の XSL 文章から DomXsltStylesheet オブジェクトを作成する
- domxml_xslt_stylesheet — 文字列での XSL 文章から DomXsltStylesheet オブジェクトを作成する
- domxml_xslt_version — XSLT ライブラリのバージョンを取得する
- xpath_eval_expression — 与えられた文字列で XPath のロケーションパスを評価する

- xpath_eval — 与えられた文字列で XPath のロケーションパスを評価する
- xpath_new_context — 新規 xpath コンテキストを作成する
- xpath_register_ns_auto — 与えられた XPath コンテキストに与えられた名前空間を登録する
- xpath_register_ns — 与えられた XPath コンテキストに与えられた名前空間を登録する
- xptr_eval — 指定した文字列の XPtr ロケーションパスを評価する
- xptr_new_context — 新規 XPath コンテキストを作成する

---

enchant 関数

導入

Enchant は、» Enchant ライブラリ を PHP から利用できるようにしたものです。Enchant は、 すべてのスペリングライブラリの上位に位置するもので、 統一された操作性を提供します。 また、各ライブラリが搭載していないかもしれない機能についても実装しています。 すべての機能が "動作します"。 いろんな意味で、あらゆる意味で "動作しています"。

Enchat は以下のバックエンドをサポートしています。

- *Aspell/Pspell (Ispell の後継です)*
- *Ispell (昔からあり、デファクトスタンダードとされています)*
- *MySpell/Hunspell (OOo プロジェクトや Mozilla が使用しています)*
- *Uspell (イディッシュ語、ヘブライ語および東欧諸国語用のもので、AbiWord の CVS 内で "uspell" モジュールとしてホストされています)*
- *Hspell (ヘブライ語)*
- *AppleSpell (Mac OSX)*

要件

このバージョンは、Dom Lachowicz による » Enchant ライブラリ の関数を使用しています。 Enchant 1.2.4 以降が必要です。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/enchant.

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールでは二種類のリソースを使用しています。ひとつは ブローカー (バックエンドマネージャ) で、もうひとつが辞書です。

例

Example#1 Enchant の使用例

```
<?php
$tag = 'en_US';
$r = enchant_broker_init();
$bprovides = enchant_broker_describe($r);
echo "現在のブローカーは、次のバックエンドを提供します。\n";
print_r($bprovides);

$dicts = enchant_broker_list_dicts($r);
print_r($dicts);
if (enchant_broker_dict_exists($r,$tag)) {
    $d = enchant_broker_request_dict($r, $tag);
    $dprovides = enchant_dict_describe($d);
    echo "dictionary $tag provides:¥n";
    $spellerrors = enchant_dict_check($d, "soong");
    print_r($dprovides);
    echo "found $spellerrors spell errors¥n";
    if ($spellerrors) {
        $suggs = enchant_dict_suggest($d, "soong");
        echo "'soong' の修正候補:";
        print_r($suggs);
```

```
    }
    enchant_broker_free_dict($d);
} else {
}
enchant_broker_free($r);
?>
```

---

enchant_broker_describe

(PECL enchant:0.1-1.0.1)

enchant_broker_describe — Enchant プロバイダを列挙する

説明

array **enchant_broker_describe** (resource `$broker`)

Enchant プロバイダを列挙し、その基本情報を通知します。 同じ情報が phpinfo() からでも取得できます。

パラメータ

broker

ブローカーリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 指定したブローカーが提供するバックエンドの一覧

```
<?php
$r = enchant_broker_init();
$bprovides  = enchant_broker_describe($r);
echo "現在のブローカーは、次のバックエンドを提供します。\n";
print_r($bprovides);

?>
```

上の例の出力は、たとえば以下のようになります。

```
現在のブローカーは、次のバックエンドを提供します。
Array
(
    [0] => Array
        (
            [name] => aspell
            [desc] => Aspell Provider
            [file] => /usr/lib/enchant/libenchant_aspell.so
        )

    [1] => Array
        (
            [name] => hspell
            [desc] => Hspell Provider
            [file] => /usr/lib/enchant/libenchant_hspell.so
        )

    [2] => Array
        (
            [name] => ispell
            [desc] => Ispell Provider
            [file] => /usr/lib/enchant/libenchant_ispell.so
        )

    [3] => Array
        (
            [name] => myspell
            [desc] => Myspell Provider
            [file] => /usr/lib/enchant/libenchant_myspell.so
        )

)
```

---

enchant_broker_dict_exists

(PECL enchant:0.1-1.0.1)

enchant_broker_dict_exists — 辞書が存在するかどうかを調べる。空でないタグを使用する

説明

bool **enchant_broker_dict_exists** (resource $broker , string $tag)

辞書が存在するかどうかを調べます。空でないタグを使用します。

パラメータ

broker

ブローカーリソース。

tag

空でない、LOCALE 形式のタグ。例: us_US、ch_DE など。

返り値

タグが存在する場合に **TRUE**、存在しない場合に **FALSE** を返します。

例

Example#1 enchant_broker_dict_exists() の例

```
<?php
$tag = 'en_US';
$r = enchant_broker_init();
if (enchant_broker_dict_exists($r,$tag)) {
    echo $tag . " の辞書が見つかりました。\n";
}
?>
```

参考

- enchant_broker_describe()

---

enchant_broker_free_dict

(PECL enchant:0.1-1.0.1)

enchant_broker_free_dict — 辞書リソースを開放する

説明

bool **enchant_broker_free_dict** (resource $dict)

辞書リソースを開放します。

パラメータ

dict

辞書リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- enchant_broker_request_dict()
- enchant_broker_request_pwl_dict()

---

enchant_broker_free

(PECL enchant:0.1-1.0.1)

enchant_broker_free — ブローカーリソースおよびその辞書を開放する

説明

bool **enchant_broker_free** (resource $broker)

ブローカーリソースを、その辞書とともに開放します。

パラメータ

broker

ブローカーリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- enchant_broker_init()

---

enchant_broker_get_error

(PECL enchant:0.1-1.0.1)

enchant_broker_get_error — ブローカーの直近のエラーを返す

説明

string **enchant_broker_get_error** (resource $broker)

このブローカーで発生した直近のエラーを返します。

パラメータ

broker

ブローカーリソース。

返り値

エラーが見つかった場合にメッセージ文字列、それ以外の場合に **FALSE** を返します。

---

enchant_broker_init

(PECL enchant:0.1-1.0.1)

enchant_broker_init — 要求を満たすブローカーオブジェクトを作成する

説明

resource **enchant_broker_init** (void)

パラメータ

返り値

成功した場合にブローカーリソース、それ以外の場合に **FALSE** を返します。

参考

- enchant_broker_free()

---

enchant_broker_list_dicts

(PECL enchant:1.0.1)

enchant_broker_list_dicts — 使用可能な辞書の一覧を返す

説明

mixed **enchant_broker_list_dicts** (resource $broker)

使用可能な辞書の一覧と、その詳細を返します。

パラメータ

broker

ブローカーリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 あるブローカーで使用可能なすべての辞書を表示する

```
<?php
$r = enchant_broker_init();
$dicts = enchant_broker_list_dicts($r);
print_r($dicts);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => Array
         (
               [lang_tag] => de
                [provider_name] => aspell
                [provider_desc] => Aspell Provider
                  [provider_file] => /usr/lib/enchant/libenchant_aspell.so
         )

     [1] => Array
         (
               [lang_tag] => de_DE
                [provider_name] => aspell
                [provider_desc] => Aspell Provider
                  [provider_file] => /usr/lib/enchant/libenchant_aspell.so
         )

     [3] => Array
         (
              [lang_tag] => en
               [provider_name] => aspell
               [provider_desc] => Aspell Provider
                 [provider_file] => /usr/lib/enchant/libenchant_aspell.so
         )

     [4] => Array
         (
               [lang_tag] => en_GB
                [provider_name] => aspell
                [provider_desc] => Aspell Provider
                  [provider_file] => /usr/lib/enchant/libenchant_aspell.so
         )

     [5] => Array
         (
               [lang_tag] => en_US
                [provider_name] => aspell
                [provider_desc] => Aspell Provider
                  [provider_file] => /usr/lib/enchant/libenchant_aspell.so
         )

     [6] => Array
         (
               [lang_tag] => hi_IN
                [provider_name] => myspell
                [provider_desc] => Myspell Provider
                  [provider_file] => /usr/lib/enchant/libenchant_myspell.so
         )

)
```

参考

- enchant_broker_describe ()

---

enchant_broker_request_dict

(PECL enchant:0.1-1.0.1)

enchant_broker_request_dict — タグを使用して新しい辞書を作成する

説明

resource **enchant_broker_request_dict** (resource `$broker` , string `$tag`)

タグを使用して新しい辞書を作成します。空でない言語タグで、 使用したい辞書を指定します ("en_US"、"de_DE"、...)。

パラメータ

broker

ブローカーリソース。

tag

ロケールを表すタグ。例えば en_US、de_DE など。

返り値

成功した場合に辞書リソース、失敗した場合に **FALSE** を返します。

例

Example#1 enchant_broker_request_dict() の例

辞書が存在するかどうかを enchant_broker_dict_exists() で調べ、それを要求します。

```
<?php
$tag = 'en_US';
$broker = enchant_broker_init();
if (enchant_broker_dict_exists($broker,$tag)) {
    $dict = enchant_broker_request_dict($r, $tag);
}
?>
```

参考

- enchant_dict_describe()
- enchant_broker_dict_exists()
- enchant_broker_dict_free()

---

enchant_broker_request_pwl_dict

(PECL enchant:0.1-1.0.1)

enchant_broker_request_pwl_dict — PWL ファイルを使用して辞書を作成する。PWL ファイルは、一行にひとつの単語を記述したパーソナル単語ファイルとなる。

説明

resource **enchant_broker_request_pwl_dict** (resource `$broker` , string `$filename`)

PWL ファイルを使用して辞書を作成します。 PWL ファイルは、一行にひとつの単語を記述したパーソナル単語ファイルです。

パラメータ

broker

ブローカーリソース。

filename

PWL ファイルへのパス。

返り値

成功した場合に辞書リソース、失敗した場合に **FALSE** を返します。

参考

- enchant_dict_describe()
- enchant_broker_dict_exists()
- **enchant_broker_dict_free()**

---

enchant_broker_set_ordering

(PECL enchant:0.1-1.0.1)

enchant_broker_set_ordering — その言語で使用する辞書の優先順位を宣言する

説明

bool **enchant_broker_set_ordering** (resource `$broker` , string `$tag` , string `$ordering`)

'タグ' で表される言語で使用する辞書の優先順位を宣言します。 順位は、プロバイダ名をカンマ区切りでつないだリストで表します。 例外として、言語タグに "*" を使用すると、あらゆる言語についての デフォルトの順位を宣言します。 明示的に順位を指定しなかった言語についてはこれが使用されます。

パラメータ

broker

ブローカーリソース。

tag

言語タグ。"*" を使用すると、あらゆる言語についての デフォルトの順位を宣言します。 明示的に順位を指定しなかった言語についてはこれが使用されます。

ordering

プロバイダ名をカンマ区切りでつないだリスト。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

enchant_dict_add_to_personal

(PECL enchant:0.1-1.0.1)

enchant_dict_add_to_personal — パーソナル単語リストに単語を追加する

説明

void **enchant_dict_add_to_personal** (resource `$dict` , string `$word`)

指定した辞書のパーソナル単語リストに、単語を追加します。

パラメータ

dict

辞書リソース。

word

追加する単語。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- enchant_broker_request_pwl_dict()
- enchant_broker_request_dict()

---

enchant_dict_add_to_session

(PECL enchant:0.1-1.0.1)

enchant_dict_add_to_session — '単語' を、このスペルチェックセッションに追加する

説明

void **enchant_dict_add_to_session** (resource `$dict` , string `$word`)

指定した辞書に単語を追加します。 追加されるのは、アクティブなスペルチェックセッションについてのみです。

パラメータ

dict

辞書リソース。

word

追加する単語。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- enchant_broker_request_dict()

---

enchant_dict_check

(PECL enchant:0.1-1.0.1)

enchant_dict_check — 単語のスペルが正しいかどうかを調べる

説明

bool **enchant_dict_check** (resource `$dict` , string `$word`)

単語のスペルが正しい場合に **TRUE**、 そうでない場合に **FALSE** を返します。

パラメータ

dict

辞書リソース。

word

調べる単語。

返り値

単語のスペルが正しい場合に **TRUE**、 そうでない場合に **FALSE** を返します。

---

enchant_dict_describe

(PECL enchant:0.1-1.0.1)

enchant_dict_describe — 個々の辞書について説明する

説明

mixed **enchant_dict_describe** (resource $dict)

辞書の詳細を返します。

パラメータ

dict

辞書リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 enchant_broker_dict_describe() の例

辞書が存在するかどうかを enchant_broker_dict_exists() で調べ、 その詳細を表示します。

```
<?php
$tag = 'en_US';
$broker = enchant_broker_init();
if (enchant_broker_dict_exists($broker,$tag)) {
    $dict = enchant_broker_request_dict($r, $tag);
    $dict_details = enchant_dict_describe($dict);
    print_r($dict_details);
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [lang] => en_US
     [name] => aspell
     [desc] => Aspell Provider
      [file] => /usr/lib/enchant/libenchant_aspell.so
)
```

---

enchant_dict_get_error

(PECL enchant:0.1-1.0.1)

enchant_dict_get_error — 現在のスペリングセッションの直近のエラーを返す

説明

string **enchant_dict_get_error** (resource $dict)

現在のスペリングセッションの、直近のエラーを返します。

パラメータ

dict

辞書リソース。

返り値

エラーメッセージを表す文字列、あるいはエラーが発生していない場合に **FALSE** を返します。

---

enchant_dict_is_in_session

(PECL enchant:0.1-1.0.1)

enchant_dict_is_in_session — このスペリングセッションに '単語' が存在するかどうかを調べる

説明

bool **enchant_dict_is_in_session** (resource `$dict` , string `$word`)

ある単語が、現在のセッション内に既に存在するかどうかを調べます。

パラメータ

`dict`

辞書リソース。

`word`

探す単語。

返り値

単語が存在する場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- enchant_dict_add_to_session()

---

enchant_dict_quick_check

(PECL enchant:0.2.0-1.0.1)

enchant_dict_quick_check — 単語のスペルが正しいかどうかを調べ、修正候補を提供する

説明

bool **enchant_dict_quick_check** (resource `$dict` , string `$word` [, array `&$suggestions`])

単語のスペルが正しい場合は **TRUE**、そうでない場合は **FALSE** を返します。変数 suggestions を指定している場合は、 そこに修正候補が格納されます。

パラメータ

`dict`

辞書リソース。

`word`

調べる単語。

`suggestions`

単語のスペルが間違っている場合に、 この変数の中に修正候補の配列が格納されます。

返り値

単語のスペルが正しい場合に **TRUE**、 そうでない場合に **FALSE** を返します。

例

Example#1 enchant_dict_quick_check() の例

```
<?php
$tag = 'en_US';
$r = enchant_broker_init();

if (enchant_broker_dict_exists($r,$tag)) {
    $d = enchant_broker_request_dict($r, $tag);
```

```
    enchant_dict_quick_check($d, 'soong', $suggs);
    print_r($suggs);
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => song
     [1] => snog
     [2] => soon
     [3] => Sang
     [4] => Sung
     [5] => sang
     [6] => sung
     [7] => sponge
     [8] => spongy
     [9] => snag
     [10] => snug
     [11] => sonic
     [12] => sing
     [13] => songs
     [14] => Son
     [15] => Sonja
     [16] => Synge
     [17] => son
     [18] => Sejong
     [19] => sarong
     [20] => sooner
     [21] => Sony
     [22] => sown
     [23] => scone
     [24] => song's
)
```

参考

- enchant_dict_check()
- enchant_dict_suggest()

---

enchant_dict_store_replacement

(PECL enchant:0.1-1.0.1)

enchant_dict_store_replacement — 単語の修正候補を追加する

説明

void **enchant_dict_store_replacement** (resource `$dict` , string `$mis` , string `$cor`)

'mis' の修正候補として 'cor' を使用します。 @mis が @cor に置き換えられることに注意しましょう。 今後 @mis が登場すると、それは @cor で置き換えられます。 そのため、@cor が修正候補の中で衝突するかもしれません。

パラメータ

dict

辞書リソース。

mis

修正する単語。

cor

正しい単語。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

enchant_dict_suggest

(PECL enchant:0.1-1.0.1)

enchant_dict_suggest — 修正候補となる値の一覧を返す

説明

array **enchant_dict_suggest** (resource $dict , string $word)

パラメータ

dict

辞書リソース。

word

修正候補を調べる単語。

返り値

単語のスペルが間違っている場合に、修正候補の配列を返します。

例

Example#1 enchant_dict_suggest() の例

```
<?php
$tag = 'en_US';
$r = enchant_broker_init();
if (enchant_broker_dict_exists($r,$tag)) {
    $d = enchant_broker_request_dict($r, $tag);

    $spellerrors  = enchant_dict_check($d, "soong");
    echo "found  $spellerrors  spell  errors¥n";
    if ($spellerrors) {
        $suggs  = enchant_dict_suggest($d, "soong");
        echo "'soong'  の修正候補:";
        print_r($suggs);
    }
    enchant_broker_free_dict($d);
}
enchant_broker_free($r);
?>
```

参考

- enchant_dict_check()
- enchant_dict_quick_check()

目次

エラー処理およびログ記録関数

導入

以下の関数は、エラー処理およびログ記録を行います。これらの関数によ り、独自のエラー処理規則を定義することが可能になり、同時にエラーの ログを記録する方法を修正することが可能になります。これにより、ニーズに 即したエラー出力の変更と拡張が可能になります。

ログ記録関数により他のマシンやemail(またはポケベルのゲートウエイ に!)、システムログ等に直接メッセージを送信することが可能になります。 これにより、ログを行うものを選択したり、アプリケーションやWebサイ トに最も重要な部分をモニタすることが可能になります。

エラー出力関数により、エラーのフィードバックのレベルと種類、 簡単な通知からカスタマイズされた関数までエラーの際に返すもの をカスタマイズすることが可能になります。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

エラーおよびロギング設定オプション

名前	デフォルト	変更の可否	変更履歴
error_reporting	NULL	PHP_INI_ALL	
display_errors	"1"	PHP_INI_ALL	
display_startup_errors	"0"	PHP_INI_ALL	PHP 4.0.3 以降で有効です。
log_errors	"0"	PHP_INI_ALL	
log_errors_max_len	"1024"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
ignore_repeated_errors	"0"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
ignore_repeated_source	"0"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
report_memleaks	"1"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
track_errors	"0"	PHP_INI_ALL	
html_errors	"1"	PHP_INI_ALL	PHP <= 4.2.3 では PHP_INI_SYSTEM 。PHP 4.0.2 以降で有効です。
docref_root	""	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
docref_ext	""	PHP_INI_ALL	PHP 4.3.2 以降で有効です。
error_prepend_string	NULL	PHP_INI_ALL	
error_append_string	NULL	PHP_INI_ALL	
error_log	NULL	PHP_INI_ALL	
warn_plus_overloading	NULL		このオプションは、PHP 4.0.0 以降には存在しません。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブの簡単な説明を示します。

`error_reporting` integer

エラー出力レベルを設定します。パラメータは、あるビットフィールドを表 す整数か定数名で指定します。このerror_reportingのレベルと定数は、 定義済の定数および *php.ini*に記述されています。 実行時に設定するには、 error_reporting() 関数を指定してください。 display_errors ディレクティブも参照してください。

PHP 4とPHP 5のデフォルトは E_ALL & ~E_NOTICE です。 この設定は**`E_NOTICE`**レベルのエラーは出力されません。 開発時にはこのエラーを表示させたい場合もあるかもしれません。

> **注意**: 開発時に**`E_NOTICE`**を有効にすることにはいくつ かの利点があります。デバッグのために、NOTICE メッセージはコード の中のバグの可能性について警告を与えます。例えば、代入されていな い値を使用した場合は、警告を発生します。 これは、書き間違いを見付け、デバッグの時間を節約するために非常 に有用です。NOTICEメッセージは、好ましくないコードに警告します。 例えば、$arr[item] は $arr['item'] と書く方が好ましいです。 これは、PHPが"item"を定数として取り扱うためです。 定数でない場

合、PHPは配列の添字ようの文字列と判断します。

注意: PHP 5では新しいエラーレベル**E_STRICT**を使用できます。 **E_STRICT**は**E_ALL**には 含まれないため、明示的にこのエラーレベルを設定する必要があります。 開発中に**E_STRICT**を有効にすることは いくつかの利点があります。STRICTメッセージは最新かつもっとも有効で 推奨されるコーディングメソッドを使用するように手助けしてくれます。 例えば推奨されない関数を使用したさいに警告を発します。

注意: **PHP 定数の、PHP 以外での使用** PHP の定数を、*httpd.conf* など PHP の外部で使用しても何の意味もありません。外部で使用する場合には、 integer 型の値を指定しなければなりません。 また、エラーレベルはこれからも追加されることがあるので、最大値 (**E_ALL** に対応する値) は変わる可能性があります。そこで、**E_ALL** を指定する場面では *2147483647* のような数を指定するようにしましょう。 これは現状の全ビットに対応した上で、かつ値が将来追加された場合にも対応できます。

PHP 3では、*(E_ERROR ¦ E_WARNING ¦ E_PARSE)*が デフォルトの設定で、同じことを意味しました。しかし、PHP 3の *php3.ini*では定数がサポートされていないため、 error_reportingの設定は数値で指定する必要があり、この場合は、 7とします。

display_errors string

エラーをHTML出力の一部として画面に出力するかどうかを定義します。

"stderr" を指定すると、エラーの内容を *stdout* (標準出力) ではなく *stderr* (標準エラー出力) に送ります。この値は PHP 5.2.4 以降で使用可能です。それより前のバージョンでは、 このディレクティブは boolean 型でした。

注意: 開発をサポートする仕組みであり、本番のシステムでは 使用すべきではありません (例えばインターネットに接続されたシステムなど)。

注意: display_errors は実行時にも設定可能（ini_set() 関数を用いて）ですが、スクリプトが致命的（fatal）なエラーを発生した場合は その設定は反映されません。なぜなら、要求されたアクションは 実行されなかったからです。

display_startup_errors boolean

display_errorsをonにした場合でも、PHPの起動シーケンスにおいて発 生したエラーは表示されません。デバッグ時を除き、display_startup_errorsをoffにしておくことが強く推奨されます。

log_errors boolean

エラーメッセージを、サーバーのエラーログまたはerror_logに記録するかどうかを指定 します。このオプションはサーバーに依存します。

注意: 実用Webサイトではエラー表示を行う変わりにエラーを記録することを 強く推奨します。

log_errors_max_len integer

log_errorsの最大長をキロバイト単位で設定します。 error_log には、 この設定で情報が追加されます。デフォルトは 1024 で、0 を指定すると 最大長の制限は全く適用されなくなります。 この長さはエラーログに記録され、 エラー表示と $php_errormsg に適用されます。

integerを使用する際、その値はバイト単位で測られます。 この FAQ に記載された短縮表記を使用することも可能です。

ignore_repeated_errors boolean

繰り返されるメッセージを記録しません。エラーの繰り返しは、 ignore_repeated_sourceが trueに設定されるまで同じファイルの同じ行で発生します。

ignore_repeated_source boolean

メッセージの繰り返しを無視する場合にメッセージのソースを無視しま す。この設定をOnにすると、異なるファイルまたはソース行からの同じエラーメッセージの繰り返しを記録しなくなります。

report_memleaks boolean

このパラメータをOffにした場合、(stdoutまたはログに)メモリーリー クは表示されなくなります。これは、デバッグ用コンパイル時にerror_reportingで E_WARNINGを有効にしている場合のみ有効です。

track_errors boolean

有効にした場合、直近のエラーメッセージが、 $php_errormsg変 数に常に代入されます。

html_errors boolean

エラーメッセージのHTMLタグをオフにします。htmlエラー用の新しい形 式では、ユーザがエラーまたはエラーを発生した関数を説明するページ に導くようクリック可能なメッセージを出力します。これらのリファレ ンスは、docref_root およ び docref_extの設定に依存 します。

docref_root string

新しいエラーフォーマットはエラーやエラーの原因となった関数に関するマニュアル のページの情報を含んでいます。マニュアルのページによっては母国語でダウンロードが 可能であり、このiniディレクティブをマニュアルのローカルコピーのURLにセット することができます。 マニュア

ルのローカルコピーが'/manual/'でアクセスできるとすると、単に **docref_root=/manual/** とするだけです。 ローカルコピーのファイルの拡張子は**docref_ext=.html** で指定できます。拡張リファレンスを使用することもできます。例えば **docref_root=http://manual/en/** または **docref_root="http://landonize.it/?how=url&theme=classic&filter=Landon&url=http%3A%2F%2Fwww.php.net%2F"** が使用できます。

ほとんどの場合docref_rootの値の最後を'/'にしようと思うでしょう。 しかし上の二つ目の例を見ではその必要はありません。

docref_ext string

docref_rootを参照して下さ い。

注意: docref_extの値はドット '.'で始まる必要があります。

error_prepend_string string

エラーメッセージの前に出力する文字列。

error_append_string string

エラーメッセージの後に出力する文字列。

error_log string

スクリプトエラーが記録されるファイル名です。 ファイルはウェブサーバユーザで書き込めなければなりません。 *syslog* が指定されると、エラーはファイルではなく システムロガーに送られます。これは Unix では syslog(3) であり Windows NT ではイベントログのことです。システムロガーは Windows 95 ではサポートされていません。 syslog() も参照してください。 このディレクティブが設定されていない場合、エラーは SAPI エラーロガーに送信されます。これは、例えば Apache のエラーログ、 あるいは CLI なら *stderr* になります。

warn_plus_overloading boolean

有効な場合、このオプションは加算演算子 (*+*) が文字列で使用されている場合に警告を出力します。 これにより、文字列結合演算子 (.) を用いて書き直す必要があるスクリプトを見付けることが容易になります。 このオプションは、PHP 4 には存在しません。

定義済み定数

以下の定数は、PHP コアに含まれており、常に利用可能です。

注意: 以下の定数を*php.ini*で使用することができますが、 *httpd.conf*のようなPHPの外部では、 代わりにビットマスク値を使用する必要があります。

エラーとロギング

値	定数	説明	注記
1	**E_ERROR** (integer)	重大な実行時エラー。これは、メモリ確保に関する問題のように復帰で きないエラーを示します。スクリプトの実行は中断されます。	
2	**E_WARNING** (integer)	実行時の警告 (致命的なエラーではない)。スクリプトの実行は中断さ れません。	
4	**E_PARSE** (integer)	コンパイル時のパースエラー。パースエラーはパーサでのみ生成されま す。	
8	**E_NOTICE** (integer)	実行時の警告。エラーを発しうる状況に遭遇したことを示す。 ただし通常のスクリプト実行の場合にもこの警告を発することがありうる。	
16	**E_CORE_ERROR** (integer)	PHPの初期始動時点での致命的なエラー。**E_ERROR**に 似ているがPHPのコアによって発行される点が違う。	PHP 4 より
32	**E_CORE_WARNING** (integer)	（致命的ではない）警告。PHPの初期始動時に発生する。 **E_WARNING**に似ているがPHPのコアによって発行される 点が違う。	PHP 4 より
64	**E_COMPILE_ERROR** (integer)	コンパイル時の致命的なエラー。**E_ERROR**に 似ているがZendスクリプティングエンジンによって発行される点が違う。	PHP 4 より
128	**E_COMPILE_WARNING** (integer)	コンパイル時の警告（致命的ではない）。**E_WARNING**に 似ているがZendスクリプティングエンジンによって発行される点が違う。	PHP 4 より
256	**E_USER_ERROR** (integer)	ユーザーによって発行されるエラーメッセージ。**E_ERROR** に似ているがPHPコード上でtrigger_error()関数を 使用した場合に発行される点が違う。	PHP 4 より
512	**E_USER_WARNING** (integer)	ユーザーによって発行される警告メッセージ。**E_WARNING** に似ているがPHPコード上でtrigger_error()関数を 使用した場合に発行される点が違う。	PHP 4 より
1024	**E_USER_NOTICE** (integer)	ユーザーによって発行される注意メッセージ。**E_NOTICE**に に似ているがPHPコード上でtrigger_error()関数を 使用した場合に発行される点が違う。	PHP 4 より
2048	**E_STRICT** (integer)	実行時の注意。コードの相互運用性や互換性を維持するために PHP がコードの変更を提案する。	PHP 5 より

値	定数	説明	注記
4096	**E_RECOVERABLE_ERROR** (integer)	キャッチできる致命的なエラー。危険なエラーが発生したが、 エンジンが不安定な状態になるほどではないことを表す。 ユーザ定義のハンドラでエラーがキャッチされなかった場合 (set_error_handler() も参照ください) は、 **E_ERROR** として異常終了する。	PHP 5.2.0 より
8191	**E_ALL** (integer)	サポートされる全てのエラーと警告。PHP < 6 では **E_STRICT** レベルのエラーは除く。	PHP 5.2.x では 6143、それより前のバージョンでは 2047 でした。

上記の値（数値も論理値も）はどのエラーをレポートするかを指定する ビットマスクを組み立てる。ビット演算子 を使用して値を組み合わせたり特定のエラータイプをマスクすることができる。 *php.ini* では'|', '~', '!', '^' and '&'のみが解釈されることに 注意すべきであるが、しかし、*php3.ini*では ビット演算子は解釈されないことにも注意すべきである。

例

エラー処理機能を PHP で使用するための例を示します。 ファイルに(XML 形式で)情報を記録し、論理的に致命的なエラーの場合に、 開発者に電子メールを送信するようなエラー処理関数を定義します。

Example#1 スクリプト内でのエラー処理

```php
<?php
// 自分のエラーハンドリングを行います
error_reporting(0);

// ユーザーの定義したエラーハンドリング関数
function userErrorHandler($errno, $errmsg, $filename, $linenum, $vars)
{
    // エラーエントリのタイムスタンプ
    $dt = date("Y-m-d H:i:s (T)");

    // エラー文字列の連想配列を定義します。
    // 実際のところ、考慮する必要があるのは
      // E_WARNING、E_NOTICE、E_USER_ERROR、
     // E_USER_WARNING そして E_USER_NOTICE だけです。
    $errortype = array (
                 E_ERROR              => 'Error',
                 E_WARNING            => 'Warning',
                 E_PARSE              => 'Parsing Error',
                 E_NOTICE             => 'Notice',
                 E_CORE_ERROR         => 'Core Error',
                 E_CORE_WARNING       => 'Core Warning',
                 E_COMPILE_ERROR      => 'Compile Error',
                 E_COMPILE_WARNING    => 'Compile Warning',
                 E_USER_ERROR         => 'User Error',
                 E_USER_WARNING       => 'User Warning',
                 E_USER_NOTICE        => 'User Notice',
                 E_STRICT             => 'Runtime Notice',
                 E_RECOVERABLE_ERROR  => 'Catchable Fatal Error'
                 );
    // set of errors for which a var trace will be saved
    $user_errors = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);

    $err = "<errorentry>¥n";
    $err .= "¥t<datetime>" . $dt . "</datetime>¥n";
    $err .= "¥t<errornum>" . $errno . "</errornum>¥n";
    $err .= "¥t<errortype>" . $errortype[$errno] . "</errortype>¥n";
    $err .= "¥t<errormsg>" . $errmsg . "</errormsg>¥n";
    $err .= "¥t<scriptname>" . $filename . "</scriptname>¥n";
    $err .= "¥t<scriptlinenum>" . $linenum . "</scriptlinenum>¥n";

    if (in_array($errno, $user_errors)) {
        $err .= "¥t<vartrace>" . wddx_serialize_value($vars, "Variables") . "</vartrace>¥n";
    }
    $err .= "</errorentry>¥n¥n";

    // テスト用
     // echo $err;

    // エラーログを保存し、重大なユーザーエラーは自分にメールする
    error_log($err, 3, "/usr/local/php4/error.log");
    if ($errno == E_USER_ERROR) {
        mail("phpdev@example.com", "Critical User Error", $err);
    }
}


function distance($vect1, $vect2)
{
    if (!is_array($vect1) || !is_array($vect2)) {
        trigger_error("Incorrect parameters, arrays expected", E_USER_ERROR);
         return NULL;
    }

    if (count($vect1) != count($vect2)) {
        trigger_error("Vectors need to be of the same size", E_USER_ERROR);
         return NULL;
    }

    for ($i=0; $i<count($vect1); $i++) {
```

```
        $c1 = $vect1[$i]; $c2 = $vect2[$i];
        $d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("Coordinate $i in vector 1 is not a number, using zero",
                          E_USER_WARNING);
            $c1 = 0.0;
        }
        if (!is_numeric($c2)) {
            trigger_error("Coordinate $i in vector 2 is not a number, using zero",
                          E_USER_WARNING);
            $c2 = 0.0;
        }
        $d += $c2*$c2 - $c1*$c1;
    }
    return sqrt($d);
}

$old_error_handler  = set_error_handler("userErrorHandler");

// 未定義定数による警告の生成
$t = I_AM_NOT_DEFINED;

// いくつかの「ベクタ」定義
$a = array(2, 3, "foo");
$b = array(5.5, 4.3, -1.6);
$c = array(1, -3);

// ユーザエラーの生成
$t1 = distance($c, $b) . "¥n";

// 他のユーザエラーの生成
$t2 = distance($b, "i am not an array") . "¥n";

// 警告の生成
$t3 = distance($a, $b) . "¥n";

?>
```

参考

syslog()も参照してください。

---

debug_backtrace

(PHP 4 >= 4.3.0, PHP 5)

debug_backtrace — バックトレースを生成する

説明

array **debug_backtrace** (void)

debug_backtrace() は PHP バックトレースを生成します。

返り値

連想配列を返します。連想配列の要素として返される可能性があるものは以下のとおりです。

debug_backtrace() から返される可能性がある要素

名前	型	説明
function	string	カレントの関数名。 __FUNCTION__ も参照してください。
line	integer	カレントの行番号。 __LINE__ も参照してください。
file	string	カレントのファイル名。 __FILE__ も参照してください。
class	string	カレントのクラス名。 __CLASS__ も参照してください。
object	object	カレントのオブジェクト。
type	string	カレントのコール方式。メソッド呼び出しの場合は "->"、 静的なメソッド呼び出しの場合は "::" が返されます。 関数呼び出しの場合は何も返されません。
args	array	関数の内部の場合、関数の引数のリストとなります。 インクルードされたファイル内では、 読み込まれたファイルの名前となります。

変更履歴

バージョン	説明
5.1.1	カレントのオブジェクトを返せるようになりました。

例

Example#1 debug_backtrace() の例

```
<?php
// ファイル名: a.php

function a_test($str)
{
    echo "¥nHi: $str";
    var_dump(debug_backtrace());
}

a_test('friend');
?>

<?php
// ファイル名: b.php
include_once '/tmp/a.php';
?>
```

/tmp/b.php を実行した際の結果は以下のようになります。

```
Hi: friend
array(2) {
[0]=>
array(4) {
     ["file"] => string(10) "/tmp/a.php"
     ["line"] => int(10)
     ["function"] => string(6) "a_test"
     ["args"]=>
     array(1) {
        [0] => &string(6) "friend"
    }
}
[1]=>
array(4) {
     ["file"] => string(10) "/tmp/b.php"
     ["line"] => int(2)
     ["args"] =>
     array(1) {
        [0] => string(10) "/tmp/a.php"
    }
     ["function"] => string(12) "include_once"
  }
}
```

参考

- trigger_error()
- debug_print_backtrace()

---

debug_print_backtrace

(PHP 5)

debug_print_backtrace — バックトレースを表示する

説明

void **debug_print_backtrace** (void)

debug_print_backtrace() は PHP バックトレースを 表示します。関数のコール、include / require されているファイル、 そして eval() された内容などが表示されます。

パラメータ

この関数はパラメータを使用しません。

返り値

値を返しません。

例

Example#1 debug_print_backtrace() の例

```
<?php
// ファイル名: include.php
```

```php
function a() {
    b();
}

function b() {
    c();
}

function c(){
    debug_print_backtrace();
}

a();

?>
<?php
// ファイル名: test.php
// このファイルを実行する

include 'include.php';
?>
```

上の例の出力は、たとえば以下のようになります。

```
#0   eval() called at [/tmp/include.php:5]
#1   a() called at [/tmp/include.php:17]
#2   include(/tmp/include.php) called at [/tmp/test.php:3]

#0   c() called at [/tmp/include.php:10]
#1   b() called at [/tmp/include.php:6]
#2   a() called at [/tmp/include.php:17]
#3   include(/tmp/include.php) called at [/tmp/test.php:3]
```

参考

- debug_backtrace()

---

error_get_last

(PHP 5 >= 5.2.0)

error_get_last — 最後に発生したエラーを取得する

説明

array **error_get_last** (void)

最後に発生したエラーについての情報を取得します。

返り値

最後に発生したエラーについての情報を連想配列で返します。連想配列のキーは "type"、"message"、"file" および "line" となります。 エラーが発生していない場合は **NULL** を返します。 yet.

例

Example#1 error_get_last() の例

```php
<?php
echo $a;
print_r(error_get_last());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [type] => 8
     [message] => Undefined variable: a
     [file] => C:¥WWW¥index.php
     [line] => 2
)
```

参考

- エラー定数

- 変数 *$php_errormsg*
- ディレクティブ *display_errors*

error_log

(PHP 4, PHP 5)

error_log — エラーメッセージを送信する

説明

bool **error_log** (string `$message` [, int `$message_type` [, string `$destination` [, string `$extra_headers`]]])

エラーメッセージを Web サーバのエラーログ、 TCP ポート、あるいはファイルに送ります。

パラメータ

message

ログに記録されるエラーメッセージ。

message_type

メッセージをどこに送るのかを指定します。以下の中から指定できます。

error_log() ログタイプ

0	*message* は、オペレーティング・システム のシステムログのメカニズムまたはファイルのいずれかを使って PHP のシステム・ロガーに送られます。どちらが使われるかは、 設定ディレクティブ error_log の内容により決定されます。これはデフォルトのオプションです。
1	*message* は、*destination* パラメータで指定されたアドレスに、電子メール により送られます。このメッセージタイプの場合にのみ、 4 番目のパラメータである *extra_headers* が使われます。
2	*message* は、PHP のデバッグ用接続を 通して送られます。このオプションは、 リモートデバッグが有効になっている場合のみ使用できます。 このケースでは、*destination* パラメータにより、 デバッグ情報を受け取るソケットのホスト名または IP アドレス、 およびオプションでポート番号を指定します。このオプションは PHP 3 でのみ有効です。
3	*message* は *destination* で指定されたファイルに追加されます。 明示的に指定しない限り、*message* の 最後には改行文字は追加されません。

destination

メッセージの送信先。その設定は、上で説明している *message_type* パラメータの値によります。

extra_headers

追加のヘッダ。*message_type* パラメータが *1* に設定される場合に利用されます。 このメッセージタイプは、mail() と同様に 内部関数を利用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 error_log() の例

```
<?php
// データベースに接続できない場合、
// サーバログを通してエラーを通知する。
if (!Ora_Logon($username, $password)) {
    error_log("オラクルのデータベースが使用できません!", 0);
}

// FOO に失敗したら、管理者に email で通知する
if (!($foo = allocate_new_foo())) {
    error_log("大変です。FOO に失敗しました!", 1,
              "operator@example.com");
}

// これ以外の error_log() のコール方法：
error_log("大変だ!", 2, "127.0.0.1:7000");
error_log("大変だ!", 2, "loghost");
error_log("大変だ!", 3, "/var/tmp/my-errors.log");
?>
```

error_reporting

(PHP 4, PHP 5)

error_reporting — 出力する PHP エラーの種類を設定する

説明

int **error_reporting** ([int `$level`])

error_reporting() 関数は、 error_reporting ディレクティブを 実行時に設定します。PHP には多くのエラーレベルがあり、 この関数によりスクリプトの持続時間(実行時間)のレベルが設定されます。

パラメータ

`level`

新しい error_reporting レベル。ビットマスクまたは名前つき定数のどちらかです。将来の バージョンとの互換性を保証するために、名前つき定数の使用が 強く推奨されています。エラーレベルが追加されると、整数の幅は増加します。 そのため、以前の整数を使用するエラーレベルは常に期待通りに動作するとは 限りません。

利用可能なエラーレベル定数の一覧を以下に示します。 これらのエラーの実際の意味は、 定義済みの定数に 記述されています。

error_reporting() レベル定数とビット値

値	定数
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
6143	E_ALL
2048	E_STRICT
4096	E_RECOVERABLE_ERROR

返り値

変更前の error_reporting レベルを返します。

変更履歴

バージョン	説明
5.0.0	**E_STRICT** が追加されました (これは **E_ALL** には含まれません)。
5.2.0	**E_RECOVERABLE_ERROR** が追加されました。
6	**E_STRICT** が **E_ALL** に含まれるようになりました。

例

Example#1 error_reporting() の例

```
<?php

// 全てのエラー出力をオフにする
error_reporting(0);

// 単純な実行時エラーを表示する
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// E_NOTICE を表示させるのもおすすめ（初期化されていない
// 変数、変数名のスペルミスなど…）
```

```php
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE);

// E_NOTICE 以外の全てのエラーを表示する
// これは php.ini で設定されているデフォルト値
error_reporting(E_ALL ^ E_NOTICE);

// 全ての PHP エラーを表示する（ビット 63 は PHP 3 で利用される）
error_reporting(E_ALL);

// error_reporting(E_ALL); と同じ
ini_set('error_reporting', E_ALL);

?>
```

注意

警告

ほとんどの **E_STRICT** レベルのエラーは スクリプトのコンパイル時に発生します。そのため、 error_reporting で **E_STRICT** を含むように設定されている環境では これらのエラーを検出できません (逆も同様です)。

参考

- display_errors ディレクティブ
- ini_set()

---

restore_error_handler

(PHP 4 >= 4.0.1, PHP 5)

restore_error_handler — 以前のエラーハンドラ関数を回復する

説明

bool **restore_error_handler** (void)

set_error_handler() を使用してエラーハンドラ関数を 変更した後、元のエラーハンドラ(組込またはユーザ定義関数)に戻すために 使用されます。

返り値

この関数は常に **TRUE** を返します。

例

Example#1 restore_error_handler() の例

unserialize() がエラーを発生した場合に 元のエラーハンドラに戻すことにする

```php
<?php
function unserialize_handler($errno, $errstr)
{
    echo "Invalid serialized value.¥n";
}

$serialized = 'foo';
set_error_handler('unserialize_handler');
$original = unserialize($serialized);
restore_error_handler();
?>
```

上の例の出力は以下となります。

```
Invalid serialized value.
```

注意

> **注意**: *error_handler* 関数の中から **restore_error_handler()** がコールされた場合、 それは無視されます。

参考

- error_reporting()
- set_error_handler()
- restore_exception_handler()
- trigger_error()

restore_exception_handler

(PHP 5)

restore_exception_handler — 以前の例外ハンドラ関数を回復する

説明

bool **restore_exception_handler** (void)

set_exception_handler() を使用して例外ハンドラ関数を 変更した後、元の例外ハンドラ(組込またはユーザ定義関数)に戻すために 使用されます。

返り値

この関数は常に **TRUE** を返します。

参考

- set_exception_handler()
- set_error_handler()
- restore_error_handler()
- error_reporting()

set_error_handler

(PHP 4 >= 4.0.1, PHP 5)

set_error_handler — ユーザ定義のエラーハンドラ関数を設定する

説明

mixed **set_error_handler** (callback `$error_handler` [, int `$error_types`])

スクリプトのエラー処理を行うユーザ関数 (*error_handler*)を設定します。

この関数は、実行時のエラー処理をユーザが定義するために使用します。 例えば、致命的なエラーの際にデータやファイルを消去する必要があるようなアプリケーションや、ある条件のもとに (trigger_error()を使用して)エラーを発生する必要がある アプリケーションがこの場合にあたります。

PHP の標準のエラーハンドラは完全にバイパスされることに注意してください。 error_reporting() の設定にかかわらず、どのような場合でも ユーザが設定したエラーハンドラがコールされます。ただし、この場合でも ハンドラで error_reporting() のカレントの値を読み、 それにあわせて適切に動作させることは可能です。エラーを発生した命令の前に @ エラー制御演算子 が付加されている場合、この値は 0 となることには注意しましょう。

ユーザハンドラ関数は、必要に応じて die() を コールする責任があることにも注意しましょう。エラーハンドラ関数が リターンした場合、スクリプトの実行は、エラーを発生した命令の次の命令に 継続されます。

以下のエラータイプは、ユーザ定義の関数では扱えません。 **E_ERROR**, **E_PARSE**, **E_CORE_ERROR**, **E_CORE_WARNING**, **E_COMPILE_ERROR**, **E_COMPILE_WARNING** および **set_error_handler()** がコールされたファイルで発生した 大半の **E_STRICT** 。

(ファイルアップロードのように)スクリプトが実行される前にエラーが 発生した場合、カスタムエラーハンドラはコールされません。 これは、その時点では登録されていないためです。

パラメータ

error_handler

ユーザ関数は、エラーコードとエラーを記述する文字列の 2 つの引数を 受け取る必要があります。さらにオプションのパラメータとして 3 つの引数が 追加されています。これらは、エラーが発生したファイル名、エラーが 発生した行、発生したエラーのコンテキスト(エラーが発生した場所での アクティブなシンボルテーブルを指す配列)です。関数は以下のようになります。

handler (int `$errno` , string `$errstr` [, string `$errfile` [, int `$errline` [, array `$errcontext`]]])

errno

最初のパラメータ *errno* は、発生させる エラーのレベルを整数で格納します。

errstr

2 番目のパラメータ *errstr* は、 エラーメッセージを文字列で格納します。

errfile

3 番目のパラメータ *errfile* はオプションで、 エラーが発生したファイルの名前を文字列で格納します。

errline

4 番目のパラメータ *errline* はオプションで、 エラーが発生した行番号を整数で格納します。

errcontext

5 番目のパラメータ *errcontext* はオプションで、 エラーが発生した場所のアクティブシンボルテーブルを指す配列です。 つまり、エラーが発生したスコープ内でのすべての変数の内容を格納した 配列が *errcontext* だということです。 ユーザエラーハンドラは、決してエラーコンテキストを書き換えては いけません。

この関数が **FALSE** を返した場合は、通常のエラーハンドラが処理を引き継ぎます。

error_types

設定パラメータ error_reporting で表示するエラーを制御するのと全く同様に、 *error_handler* の起動を制御する際に 使用可能です。 このマスクを指定しない場合、 *error_handler* は error_reporting の設定によらず 全てのエラーに関してコールされます。

返り値

前に定義されたエラーハンドラ(ある場合)を含む文字列、または エラーの場合には **NULL** を返します。 前に定義されたハンドラがクラスメソッドの場合、この関数は、 クラスとメソッド名からなる添字配列を返します。

変更履歴

バージョン	説明
5.2.0	$php_errormsg の内容を設定するため、エラーハンドラは必ず **FALSE** を返さなければなりません。
5.0.0	*error_types* パラメータが追加されました。
4.3.0	*error_handler* として、関数名のかわりに オブジェクトへのリファレンスとメソッド名を含む配列を指定することも できます。
4.0.2	ユーザ定義関数 *error_handler* で 3 つの オプションパラメータが利用できるようになりました。filename, line number, および context です。

例

Example#1 set_error_handler() および trigger_error() によるエラー処理

以下の例では、エラーを発生させることによる内部例外の処理や それらをユーザ定義関数で処理する方法を説明します。

```
<?php
// エラーハンドラ関数
function myErrorHandler($errno, $errstr, $errfile, $errline)
{
    switch ($errno) {
    case E_USER_ERROR:
        echo "<b>My ERROR</b> [$errno] $errstr<br />¥n";
        echo "  Fatal error on line $errline in file $errfile";
        echo ", PHP " . PHP_VERSION . " (" . PHP_OS . ")<br />¥n";
        echo "Aborting...<br />¥n";
        exit(1);
        break;

    case E_USER_WARNING:
        echo "<b>My WARNING</b> [$errno] $errstr<br />¥n";
        break;

    case E_USER_NOTICE:
        echo "<b>My NOTICE</b> [$errno] $errstr<br />¥n";
        break;

    default:
        echo "Unknown error type: [$errno] $errstr<br />¥n";
        break;
    }

    /* PHP の内部エラーハンドラを実行しません */
    return true;
}

// エラー処理のテスト用関数
function scale_by_log($vect, $scale)
{
    if (!is_numeric($scale) || $scale <= 0) {
        trigger_error("log(x) for x <= 0 is undefined, you used: scale = $scale", E_USER_ERROR);
    }

    if (!is_array($vect)) {
        trigger_error("Incorrect input vector, array of values expected", E_USER_WARNING);
        return null;
    }

    $temp = array();
    foreach($vect as $pos => $value) {
        if (!is_numeric($value)) {
            trigger_error("Value at position $pos is not a number, using 0 (zero)", E_USER_NOTICE);
            $value = 0;
```

```
        }
        $temp[$pos] = log($scale) * $value;
    }

    return $temp;
}

// 定義したエラーハンドラを設定する
$old_error_handler = set_error_handler("myErrorHandler");

// エラーを発生します。まず、数値でない項目が混ざった配列を定義します。
echo "vector a¥n";
$a = array(2, 3, "foo", 5.5, 43.3, 21.11);
print_r($a);

// 二番目の配列を生成します。
echo "----¥nvector b - a notice (b = log(PI) * a)¥n";
/* Value at position $pos is not a number, using 0 (zero) */
$b = scale_by_log($a, M_PI);
print_r($b);

// 配列の代わりに文字列を渡しており、問題を発生します。
echo "----¥nvector c - a warning¥n";
/* Incorrect input vector, array of values expected */
$c = scale_by_log("not array", 2.3);
var_dump($c); // NULL

// ゼロまたは負数の対数が定義されないという致命的なエラーを発生します。
echo "----¥nvector d - fatal error¥n";
/* log(x) for x <= 0 is undefined, you used: scale = $scale" */
$d = scale_by_log($a, -2.5);
var_dump($d); // ここには到達しません
?>
```

上の例の出力は、たとえば以下のようになります。

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
vector b - a notice (b = log(PI) * a)
<b>My NOTICE</b> [1024] Value at position 2 is not a number, using 0 (zero)<br />
Array
(
    [0] => 2.2894597716988
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
----
vector c - a warning
<b>My WARNING</b> [512] Incorrect input vector, array of values expected<br />
NULL
----
vector d - fatal error
<b>My ERROR</b> [256] log(x) for x <= 0 is undefined, you used: scale = -2.5<br />
  Fatal error on line 35 in file trigger_error.php, PHP 5.2.1 (FreeBSD)<br />
Aborting...<br />
```

参考

- error_reporting()
- restore_error_handler()
- trigger_error()
- エラーレベル定数
- callback 型に関する情報

---

set_exception_handler

(PHP 5)

set_exception_handler — ユーザ定義の例外ハンドラ関数を設定する

説明

string **set_exception_handler** (callback `$exception_handler`)

例外が try/catch ブロックの中でキャッチされなかった場合の デフォルトの例外ハンドラを設定します。 例外は、*exception_handler* がコールされた後に 停止します。

パラメータ

exception_handler

キャッチされない例外が発生した際にコールされる関数の名前。 この関数は、**set_exception_handler()** をコールする前に定義する必要があります。 このハンドラ関数は、パラメータをひとつとる必要があります。 このパラメータは、スローされた例外オブジェクトとなります。

返り値

前に定義された例外ハンドラの名前、またはエラー発生時に **NULL** を返します。 前にハンドラが定義されていない場合にも **NULL** が返されます。

例

Example#1 set_exception_handler() の例

```
<?php
function exception_handler($exception) {
  echo "Uncaught exception: " , $exception->getMessage(), "¥n";
}

set_exception_handler('exception_handler');

throw new Exception('Uncaught Exception');
echo "Not Executed¥n";
?>
```

参考

restore_exception_handler(), restore_error_handler(), error_reporting(), callback 型に関する情報, そして PHP 5 例外.

---

trigger_error

(PHP 4 >= 4.0.1, PHP 5)

trigger_error — ユーザレベルのエラー/警告/通知メッセージを生成する

説明

bool **trigger_error** (string $error_msg [, int $error_type])

ユーザエラーを発生するために使用され、 組み込みのエラーハンドラまたは新しいエラーハンドラ (set_error_handler()) として設定済みのユーザ定義関数と組み合わせて使用されます。

この関数は、実行時の例外に特定の応答を生成する必要がある場合に便利です。

パラメータ

error_msg

このエラーに割り当てられたメッセージ。長さは最大 1024 文字までです。 1024 文字を超える部分は切り捨てられます。

error_type

このエラーに割り当てられたエラー型。E_USER 関連の定数のみが指定可能で、 デフォルトは **E_USER_NOTICE** です。

返り値

この関数は、間違った *error_type* が指定された場合に **FALSE** を、それ以外の場合に **TRUE** を返します。

例

Example#1 trigger_error() の例

より拡張した例については set_error_handler() を参照ください。

```
<?php
if (assert($divisor == 0)) {
    trigger_error("ゼロで割ることはできません", E_USER_ERROR);
}
?>
```

参考

- error_reporting()
- set_error_handler()
- restore_error_handler()
- エラーレベル定数

---

user_error

(PHP 4, PHP 5)

user_error — trigger_error() のエイリアス

説明

この関数は次の関数のエイリアスです。 trigger_error().

目次

Exif 関数

導入

exif 拡張モジュールを使用すると、画像のメタデータを扱うことが可能と なります。例えば、デジタルカメラで撮影した画像ファイルから JPEG や TIFF 画像のヘッダ情報を 読み込むために exif 関数を使用することができます。

要件

--enable-exif オプションを含めて PHP がコンパイル されている必要があります。exif モジュールを作成するために、追加の ライブラリは一切必要ありません。Windows ユーザは、 mbstring 拡張モジュールを有効に する必要があります。

インストール手順

exif サポートを有効にするには、PHP の configure 時に --enable-exif を指定します。

Windows ユーザは、*php.ini* で *php_mbstring.dll* および *php_exif.dll* の両方の DLL を有効にする 必要があります。*php_mbstring.dll* DLL は、必ず *php_exif.dll* DLL より *先に* 読み込まれていなければなりません。そうなるように *php.ini* で設定 してください。

実行時設定

php.ini の設定により動作が変化します。

mbstring が有効になっている場合、 Exif 拡張モジュールはユーザコメントの文字エンコーディングの変換 （Unicode と JIS）を自動的に行います。この処理が行われるのは、 指定した文字セットを使用して最初にコメントをデコードする際です。 その結果は、*HTTP* 出力用の文字コードにエンコード されます。

Exif 設定オプション

名前	デフォルト	変更の可否	変更履歴
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
exif.encode_jis	""	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
exif.decode_jis_intel	"JIS"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

exif.encode_unicode string

exif.encode_unicode は、UNICODE ユーザコメント を処理する文字セットを定義します。デフォルトは ISO-8859-15 で、 アジア以外のほとんどの国で動作します。この設定は、空白 あるいは mbstring のサポートするエンコーディングである必要があります。 空白の場合は mbstring の内部エンコーディング設定が使用されます。

exif.decode_unicode_motorola string

exif.decode_unicode_motorola は、画像の バイトオーダーがモトローラ形式（ビッグエンディアン）であった場合に Unicode のユーザコメントを扱うための内部文字セットを定義します。 この設定は空白にすることはできず、mbstring がサポートしている エンコーディングの中から指定します。デフォルトは UCS-2BE です。

exif.decode_unicode_intel string

exif.decode_unicode_intel は、画像の バイトオーダーがインテル形式（リトルエンディアン）であった場合に Unicode のユーザコメントを扱うための内部文字セットを定義します。 この設定は空白にすることはできず、mbstring がサポートしている エンコーディングの中から指定します。デフォルトは UCS-2LE です。

exif.encode_jis string

exif.encode_jis は、JIS ユーザコメントを 処理する文字セットを定義します。デフォルトは空白で、これは mbstring の内部設定を使用させることを意味します。

exif.decode_jis_motorola string

exif.decode_jis_motorola は、画像の バイトオーダーがモトローラ形式（ビッグエンディアン）であった場合に JIS のユーザコメントを扱うための内部文字セットを定義します。 この設定は空白にすることはできず、mbstring がサポートしている エンコーディングの中から指定します。デフォルトは JIS です。

exif.decode_jis_intel string

exif.decode_jis_intel は、画像の バイトオーダーがインテル形式（リトルエンディアン）であった場合に JIS のユーザコメントを扱うための内部文字セットを定義します。 この設定は空白にすることはできず、mbstring がサポートしている エンコーディングの中から指定します。デフォルトは JIS です。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

EXIF_USE_MBSTRING (integer)

exif_imagetype() は、関連するいくつかの組み込み定数を 一覧表示します。

---

exif_imagetype

(PHP 4 >= 4.3.0, PHP 5)

exif_imagetype — イメージの型を定義する

説明

int **exif_imagetype** (string `$filename`)

exif_imagetype() を画像の先頭バイトを読み そのサインを調べます。

exif_imagetype() は、他の exif 関数がサポートしていないファイル形式で コールされるの防いだり、*$_SERVER['HTTP_ACCEPT']* と組み合わせて閲覧者が画像を見る権限を持っているかどうかを調べたりするために 使用可能です。

パラメータ

`filename`
　　調べる画像。

返り値

正しいサインが見つかった場合は適切な定数、それ以外の場合は **FALSE** を返します。返り値は getimagesize() がインデックス 2 に対して返す値と同じですが、**exif_imagetype()** のほうがずっと早く動作します。

変更履歴

バージョン	説明
4.3.2	JPC、JP2、 JPX、JB2、 XBM および WBMP がサポートされます。
4.3.0	SWC がサポートされます。

定義済み定数

以下の定数が定義されており、**exif_imagetype()** の返り値を表しています。

Imagetype 定数

値	定数
1	**IMAGETYPE_GIF**
2	**IMAGETYPE_JPEG**
3	**IMAGETYPE_PNG**
4	**IMAGETYPE_SWF**
5	**IMAGETYPE_PSD**
6	**IMAGETYPE_BMP**
7	**IMAGETYPE_TIFF_II** (intel byte order)
8	**IMAGETYPE_TIFF_MM** (motorola byte order)
9	**IMAGETYPE_JPC**
10	**IMAGETYPE_JP2**
11	**IMAGETYPE_JPX**
12	**IMAGETYPE_JB2**
13	**IMAGETYPE_SWC**
14	**IMAGETYPE_IFF**
15	**IMAGETYPE_WBMP**
16	**IMAGETYPE_XBM**

例

Example#1 exif_imagetype() の例

```
<?php
if (exif_imagetype('image.gif') != IMAGETYPE_GIF) {
    echo 'The picture is not a gif';
}
?>
```

参考

- getimagesize()

---

exif_read_data

(PHP 4 >= 4.2.0, PHP 5)

exif_read_data — JPEG あるいは TIFF から EXIF ヘッダを読み込む

説明

array **exif_read_data** (string `$filename` [, string `$sections` [, bool `$arrays` [, bool `$thumbnail`]]])

exif_read_data() は、JPEG あるいは TIFF の画像ファイルから EXIF ヘッダを読み込みます。この方法で、デジタルカメラが生成したメタデータを 読み込むことが可能です。

Exif ヘッダは、デジタルカメラが作成した JPEG/TIFF 画像によく含まれています。しかし残念なことに、 そのタグ付けの方法はメーカーによって異なります。 したがって、特定の Exif ヘッダが常に存在すると仮定することはできません。

Height および *Width* は、getimagesize() と同じ方法で計算されます。 よって、これらの値は決してヘッダの一部として返してはいけません。 また、*html* は高さ/幅を表すテキスト文字列で、 通常の HTML の中で用いられます。

Exif ヘッダに著作権表示が含まれている場合、それ自身には 2 つの値を 含めることが可能です。Exif 2.10 のこの矛盾した規格に対応するため、COMPUTED セクションは *Copyright.Photographer* および *Copyright.Editor* の両方を返します。 また IFD0 セクションには 2 つのエントリを NULL 文字で区切った バイト配列を含めます。データ型が間違っている場合は最初のエントリのみを かえします（Exif の通常の挙動）。COMPUTED には、元の著作権文字列あるいは カンマで区切られた写真と編集者の著作権表示のどちらかを *Copyright* エントリに含めることが可能です。

UserComment タグにも Copyright タグと同様の問題があります。ここにも 2 つの値を格納することが可能です。それは使用しているエンコーディングと 値自身の 2 つです。そうなると、IFD セクションにはエンコーディングのみを 含めるか、あるいはバイト配列を格納することになります。COMPUTED セクションは *UserCommentEncoding* および *UserComment* を両方格納することができます。*UserComment* はどちらの 場合でも有効なので、IFD0 セクションではこちらを優先すべきです。

exif_read_data() は、EXIF 仕様（» http://exif.org/Exif2-2.PDF, 20 ページ）に基づいて EXIF データタグの検証も行います。

> **注意**: Windows ME/XP は、カメラと接続した際に Exif ヘッダを書き換えることが可能です。 詳細な情報は » http://www.canon.co.jp/Imaging/NOTICE/011214-e.html を参照ください。

パラメータ

`filename`

読み込む画像ファイルの名前。 URL 形式は使用できません。

`sections`

結果の配列を作成するために存在する必要のあるセクションのカンマ区切り リスト。要求されたセクションがひとつも見つからなかった場合の返り値は **FALSE** となります。

FILE	FileName, FileSize, FileDateTime, SectionsFound
COMPUTED	html, Width, Height, IsColor, および他の取得可能なもの。Height および Width は getimagesize() と同じ方法で取得した もので、その値はヘッダの一部ではありません。また、html は 通常の HTML 内で使用される height/width の 文字列です。
ANY_TAG	タグを有するすべての情報。例えば IFD0, EXIF, ...
IFD0	IFD0 のすべてのタグつきデータ。通常の画像ファイルでは、 ここに画像のサイズが含まれます。
THUMBNAIL	2 番目の IFD がある場合、ファイルにサムネイルが含まれている 可能性があります。埋め込まれたサムネイルに関するすべてのタグつき情報はこのセクションに格納されます。
COMMENT	JPEG 画像のコメントヘッダ。
EXIF	EXIF セクションは IFD0 のサブセクションです。ここには 画像に関する詳細な情報が含まれています。これらのエントリの ほとんどはデジタルカメラに関連するものです。

`arrays`

各セクションを配列とするかどうかを指定します。 *COMPUTED*、 *THUMBNAIL* および *COMMENT* のセクションは常に配列となります。これは、これらのセクションに 含まれる値の名前が他のセクションと衝突する可能性があるからです。

`thumbnail`

TRUE を指定すると、サムネイル本体を読み込みます。それ以外の場合は タグつきデータのみを読み込みます。

返り値

ヘッダ名がキー・ヘッダの内容が値となる連想配列を返します。 返されるデータがない場合は **exif_read_data()** は **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	埋め込まれた IFD データを、配列を含めて すべて読み込めます（そして返せます）。また、埋め込まれた サムネイルの大きさもサブ配列 *THUMBNAIL* に 格納され、TIFF フォーマットのサムネイルを 返すことが可能です。さらに、（メモリの制限に達しない限り） 返り値の長さの最大長には制限はありません。
4.3.0	PHP の mbstring サポートが 有効になっている場合、ユーザコメントのエンコーディングは自動的に 変換されます。また、ユーザコメントが Unicode あるいは JIS エンコーディングを使用している場合は *php.ini* の exif 設定に応じて エンコーディングが自動的に変換されます。
4.3.0	画像に何らかの IFD0 データが含まれている場合、COMPUTED にはエントリ ByteOrderMotorola が含まれます。このエントリは、バイトオーダーが リトルエンディアン（intel）の場合に 0 、ビッグエンディアン （motorola）の場合に 1 となります。また、COMPUTED および UserComment は、仮にデータ型が間違っていても最初の copyright エントリだけを格納するということはありません。

例

Example#1 exif_read_data() の例

```
<?php
echo "test1.jpg:<br /> ¥n";
$exif = exif_read_data('tests/test1.jpg', 'IFD0');
echo $exif===false ? "No header data found.<br /> ¥n" : "Image contains headers<br /> ¥n";

$exif = exif_read_data('tests/test2.jpg', 0, true);
echo "test2.jpg:<br /> ¥n";
foreach ($exif as $key => $section) {
     foreach ($section as $name => $val) {
         echo "$key.$name: $val<br /> ¥n";
     }
}
?>
```

最初のコールは失敗します。画像がヘッダ情報を有していないためです。

上の例の出力は、たとえば以下のようになります。

```
test1.jpg:
No header data found.
test2.jpg:
FILE.FileName: test2.jpg
FILE.FileDateTime: 1017666176
FILE.FileSize: 1240
FILE.FileType: 2
FILE.SectionsFound: ANY_TAG, IFD0, THUMBNAIL, COMMENT
COMPUTED.html: width="1" height="1"
COMPUTED.Height: 1
COMPUTED.Width: 1
COMPUTED.IsColor: 1
COMPUTED.ByteOrderMotorola: 1
COMPUTED.UserComment: Exif test image.
COMPUTED.UserCommentEncoding: ASCII
COMPUTED.Copyright: Photo (c) M.Boerger, Edited by M.Boerger.
COMPUTED.Copyright.Photographer: Photo (c) M.Boerger
COMPUTED.Copyright.Editor: Edited by M.Boerger.
IFD0.Copyright: Photo (c) M.Boerger
IFD0.UserComment: ASCII
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.JPEGInterchangeFormatLength: 523
COMMENT.0: Comment #1.
COMMENT.1: Comment #2.
COMMENT.2: Comment #3end
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.Thumbnail.Height: 1
THUMBNAIL.Thumbnail.Height: 1
```

参考

- exif_thumbnail()
- getimagesize()

---

exif_tagname

(PHP 4 >= 4.2.0, PHP 5)

exif_tagname — インデックスに対応するヘッダ名を取得する

説明

string **exif_tagname** (string $index)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

index

画像のインデックス。

返り値

ヘッダ名を返します。もし *index* が未定義の場合は **FALSE** を返します。 undefined.

参考

- exif_imagetype()

---

exif_thumbnail

(PHP 4 >= 4.2.0, PHP 5)

exif_thumbnail — TIFF あるいは JPEG 形式の画像に埋め込まれたサムネイルを取得する

説明

string **exif_thumbnail** (string $filename [, int &$width [, int &$height [, int &$imagetype]]])

exif_thumbnail() は、 TIFF あるいは JPEG 画像に 埋め込まれたサムネイルを読み込みます。

この関数を使用してサムネイルを出力したい場合は、 header() 関数を使用して mimetype 情報を送信する 必要があります。

exif_thumbnail() が画像を作成することはできないが、 そのサイズだけはわかるという可能性があります。そのような場合、返り値は **FALSE** となりますが *width* および *height* を設定されています。

パラメータ

filename

読み込む画像ファイルの名前。サムネイルが埋め込まれています。

width

返されるサムネイルの幅が格納されます。

height

返されるサムネイルの高さが格納されます。

imagetype

返されるサムネイルの画像タイプが格納されます。 TIFF あるいは JPEG のどちらかです。

返り値

埋め込まれたサムネイルを返します。画像がサムネイルを含まない場合は **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	オプションで *width* 、 *height* および *imagetype* が使用可能になりました。
4.3.0	サムネイルを TIFF フォーマットで返すことがあります。

例

Example#1 exif_thumbnail() の例

```
<?php
if (array_key_exists('file', $_REQUEST)) {
```

```
    $image = exif_thumbnail($_REQUEST['file'], $width, $height, $type);
} else {
    $image = false;
}
if ($image!==false) {
    header('Content-type: ' .image_type_to_mime_type($type));
    echo $image;
    exit;
} else {
    // no thumbnail available, handle the error here
    echo 'No thumbnail available';
}
?>
```

参考

- exif_read_data()
- image_type_to_mime_type()

---

read_exif_data

(PHP 4 >= 4.0.1, PHP 5)

read_exif_data — exif_read_data() のエイリアス

説明

この関数は次の関数のエイリアスです。 exif_read_data()

目次

Expect 関数

導入

この拡張モジュールは、PTY を経由したプロセス同士の対話機能を提供します。 ファイルシステム関数 で *expect://* ラッパ を利用すれば、シンプルでより直感的なインターフェースが利用できるでしょう。

要件

このモジュールは » expect ライブラリの 関数を使用します。libexpect バージョン >= 5.43.0 が必要です。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/expect.

PHP 4 の場合、この PECL 拡張モジュールのソースは、PHP のソースの *ext/* ディレクトリ、または上の PECL リンクで入手可能です。 これらの関数を使用するには、--with-expect[=DIR] オプションを使用して expect サポートつきで PHP をコンパイルする必要があります。

Windows ユーザは、これらの関数を使用するためには *php.ini* の中で *php_expect.dll* を有効にします。 PHP 4 の場合、この DLL は PHP の Windows ダウンロードバイナリの *extensions/* ディレクトリにあります。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

php.ini の設定により動作が変化します。

expect 拡張モジュールを設定するために、 設定ファイル *php.ini* に 設定項目が用意されています。

Expect 設定オプション

名前	デフォルト	変更可能	変更履歴
expect.timeout	"10"	PHP_INI_ALL	
expect.loguser	"1"	PHP_INI_ALL	
expect.logfile	""	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

expect.timeout integer

データを待ち受ける際のタイムアウト時間です。 expect_expectl() 関数で使用されます。

"-1" を指定すると、タイムアウトを発生させないようにします。

注意: "0" を指定すると、expect_expectl() 関数は 結果を直ちに返します。

expect.loguser boolean

expect が、子プロセスの出力を標準出力に送るかどうかを指定します。 典型的な対話型プログラムは入力した内容を表示するので、これを使用すれば 対話の両方の側を表示することができます。

expect.logfile string

子プロセスの出力内容が書き込まれるファイルの名前。もしファイルが 存在しない場合は新しく作成されます。

注意: この設置が空欄でなかった場合、 expect.loguser の設定内容に かかわらず出力が書き込まれます。

リソース型

expect_popen() は、オープンした PTY ストリームを返します。これを expect_expectl() で使用します。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

EXP_GLOB (integer)
パターンが、glob 形式の文字列パターンであることを示します。
EXP_EXACT (integer)
パターンが、単なる文字列であることを示します。
EXP_REGEXP (integer)
パターンが、正規表現形式の文字列パターンであることを示します。
EXP_EOF (integer)
EOF に到達した際に expect_expectl() が返す値です。
EXP_TIMEOUT (integer)
expect.timeout で指定した秒数を こえた際に expect_expectl() が返す値です。
EXP_FULLBUFFER (integer)
一致するパターンがなかった際に expect_expectl() が返す値です。

例

この例ではリモートホストに SSH 経由で接続し、接続先の稼働時間を表示します。

Example#1 Expect の使用例

```
<?php
ini_set ("expect.loguser", "Off");

$stream = fopen ("expect://ssh root@remotehost uptime", "r");

$cases = array (
   array (0 => "password:", 1 => PASSWORD)
);

switch (expect_expectl ($stream, $cases)) {
 case PASSWORD:
  fwrite ($stream, "password¥n");
   break;

 default:
```

```
  die ("リモートホストへの接続時にエラーが発生しました!\n");
}

while ($line = fgets ($stream)) {
   print $line;
}
fclose ($stream);
?>
```

次の例は、リモートホストに接続して インストールされている OS が 32 ビットか 64 ビットかを確認し、 それぞれのパッケージのアップデートを実行します。

Example#2 もうひとつの Expect の使用例

```
<?php
ini_set ("expect.timeout", -1);
ini_set ("expect.loguser", "Off");

$stream = expect_popen ("ssh root@remotehost");

while (true) {
     switch (expect_expectl ($stream, array (
            array ("password:", PASSWORD), // SSH がパスワードを問い合わせます
           array ("yes/no)?", YESNO), // SSH がホストエントリを保存するかどうかを問い合わせます
           array ("~$ ", SHELL, EXP_EXACT), // シェルにたどり着きました!
    ))) {
         case PASSWORD:
             fwrite ($stream, "secret¥n");
              break;

         case YESNO:
             fwrite ($stream, "yes¥n");
              break;

         case SHELL:
             fwrite ($stream, "uname -a¥n");
              while (true) {
                       switch (expect_expectl ($stream, array (
                              array ("~$ ", SHELL, EXP_EXACT), // シェルにたどり着きました!
                              array ("^Linux.*$", UNAME, EXP_REGEXP), // uname -a の出力
                     ), $match)) {
                          case UNAME:
                              $uname .= $match[0];
                               break;

                          case SHELL:
                            // アップデートの実行
                             if (strstr ($uname, "x86_64")) {
                                     fwrite ($stream, "rpm -Uhv http://mirrorsite/somepath/some_64bit.rpm¥n");
                              } else {
                                     fwrite ($stream, "rpm -Uhv http://mirrorsite/somepath/some_32bit.rpm¥n");
                              }
                              fwrite ($stream, "exit¥n");
                               break 2;

                          case EXP_TIMEOUT:
                          case EXP_EOF:
                              break 2;

                         default:
                            die ("エラーが発生しました!\n");
                     }
             }
              break 2;

         case EXP_TIMEOUT:
         case EXP_EOF:
              break 2;

          default:
             die ("エラーが発生しました!\n");
     }
}

fclose ($stream);
?>
```

---

expect_expectl

(PECL expect:0.1-0.2.2)

expect_expectl — プロセスの出力がパターンに一致する・指定した時間が経過する・ あるいは EOF に達するのいずれかにあてはまるまで待ち続ける

説明

int **expect_expectl** (resource `$expect` , array `$cases` [, array `&$match`])

プロセスの出力がパターンに一致する・指定した時間が経過する・ あるいは EOF に達するのいずれかにあてはまるまで待ち続けます。

match を指定すると、検索結果がそこに保存されます。 一致した文字列が *match[0]* に保存され、 元のパターンの中の (括弧で囲まれた) 部分に一致する文字列が *match[1]* 、 *match[2]* 、 と順に、最大 *match[9]* まで (libexpect の制限です) 保存されます。

パラメータ

expect

事前に expect_popen() でオープンした Expect ストリーム。

cases

expect case の配列。個々の expect case は数値添字の配列で、 以下のような形式となります。

Expect Case の配列

添字	値の型	説明	必須かどうか	デフォルト値
0	string	ストリームからの出力との比較対象となるパターン。	yes	
1	mixed	パターンに一致した場合にこの関数が返す値。	yes	
2	integer	パターンの形式。 EXP_GLOB、 EXP_EXACT あるいは EXP_REGEXP のいずれかひとつです。	no	EXP_GLOB

返り値

一致したパターンに関連付けられた値を返します。

この関数の実行に失敗した場合は、 EXP_EOF、 EXP_TIMEOUT あるいは EXP_FULLBUFFER を返します。

変更履歴

バージョン	説明
0.2.1	バージョン 0.2.1 より前では、*match* パラメータに返されるのはマッチした文字列であり、 部分文字列の配列ではありませんでした。

例

Example#1 expect_expectl() の例

```
<?php
// ファイルをリモートホストにコピーします
ini_set ("expect.timeout", 30);

$stream = fopen ("expect://scp  user@remotehost:/var/log/messages  /home/user/messages.txt", "r");

$cases = array (
   array (0 => "password:", 1 => PASSWORD),
   array (0 => "yes/no)?", 1 => YESNO)
);

while (true) {
  switch (expect_expectl ($stream, $cases))
  {
    case PASSWORD:
     fwrite ($stream, "password¥n");
      break;

    case YESNO:
     fwrite ($stream, "yes¥n");
      break;

    case EXP_TIMEOUT:
    case EXP_EOF:
     break 2;

    default:
    die ("エラーが発生しました!\n");
  }
}

fclose ($stream);
?>
```

参考

- expect_popen()

---

expect_popen

(PECL expect:0.1-0.2.2)

expect_popen — Bourne シェル経由でコマンドを実行し、プロセスへの PTY ストリームをオープンする

説明

resource **expect_popen** (string $command)

Bourne シェル経由でコマンドを実行し、プロセスへの PTY ストリームを オープンします。

パラメータ

command

実行するコマンド。

返り値

プロセスの標準入力・標準出力・標準エラー出力への PTY ストリームを返します。

失敗した場合は、この関数は **FALSE** を返します。

例

Example#1 expect_popen() の例

```
<?php
// PHP.net の CVS リポジトリにログインします
$stream = expect_popen ("cvs -d :pserver:anonymous@cvs.php.net:/repository login");
sleep (3);
fwrite ($stream, "phpfi¥n");
fclose ($stream);
?>
```

参考

- popen()

目次

ファイル改変監視関数（FAM)

導入

FAM はファイルやディレクトリを監視し、変更点を調査を行うアプリケーションに 通知します。FAM についての詳細な情報は » http://oss.sgi.com/projects/fam/ で得られます。

PHP スクリプトは、この拡張モジュールにより提供される関数を用いて FAM に一連の ファイルを指定することができます。

FAM プロセスは、最初にアプリケーションから接続された時に開始され、 全ての接続がクローズされた時に終了します。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.0.

> **注意**: この拡張モジュールは Windows 環境では利用できません。

要件

この拡張モジュールは、SGI が開発した » FAM ライブラリの関数を使用しています。そのため、FAM ライブラリを ダウンロードしてインストールする必要があります。

インストール手順

PHP の FAM サポートを使用するには、--with-fam[=DIR] を指定して PHP をコンパイルする 必要があります。DIR は lib および include ディレクトリを含む ディレクトリの場所です。

実行時設定

設定ディレクティブは定義されていません。

リソース型

FAM モジュールでは二種類のリソース型を使用します。まず最初が FAM サービスとの接続を表すリソースで、これは fam_open() が返します。二番目はモニタリングリソースで、これは *fam_monitor_XXX* 関数が返します。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

FAM イベント定数

定数	説明
FAMChanged (integer)	ファイルあるいはディレクトリの、fstat(1) で取得できる値のうちの 何かが変更されました。
FAMDeleted (integer)	ファイルあるいはディレクトリが削除あるいはリネームされました。
FAMStartExecuting (integer)	実行可能ファイルが実行されました。
FAMStopExecuting (integer)	実行可能ファイルの実行が終了しました。
FAMCreated (integer)	ディレクトリ内にファイルが作成されました。
FAMMoved (integer)	このイベントは決して発生しません。
FAMAcknowledge (integer)	fam_cancel_monitor() に対する応答イベントです。
FAMExists (integer)	ファイルやディレクトリの監視を要求するイベントです。 ディレクトリが監視されている場合、ディレクトリおよびその中に 含まれるすべてのファイルについてのイベントが発生します。
FAMEndExist (integer)	最後の FAMEExists イベントの後に発生します。

fam_cancel_monitor

(PHP 5 <= 5.0.5)

fam_cancel_monitor — 監視を終了する

説明

bool **fam_cancel_monitor** (resource `$fam` , resource `$fam_monitor`)

リソースの監視を終了します。

さらに **FAMAcknowledge** イベントが発生します。

パラメータ

`fam`

A resource representing a connection to the FAM service returned by fam_open()

`fam_monitor`

fam_monitor_XXX 関数のいずれかが返すリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fam_monitor_file()
- fam_monitor_directory()
- fam_monitor_collection()
- fam_suspend_monitor()

fam_close

(PHP 5 <= 5.0.5)

fam_close — FAM 接続を閉じる

説明

void **fam_close** (resource $fam)

FAM サービスへの接続をクローズします。

パラメータ

fam

fam_open() が返す、 FAM サービスへの接続を表すリソース。

返り値

値を返しません。

参考

- **fam_monitor_open()**

---

fam_monitor_collection

(PHP 5 <= 5.0.5)

fam_monitor_collection — 指定したディレクトリにあるファイルの変更を監視する

説明

resource **fam_monitor_collection** (resource $fam , string $dirname , int $depth , string $mask)

ディレクトリ内のファイルの監視を要求します。

ファイルの状態が変化したときに FAM イベントが発生します。 発生しうるイベントコードの詳細は、このセクションの 定数 の欄にあります。

パラメータ

fam

fam_open() が返す、 FAM サービスへの接続を表すリソース。

dirname

ファイルを監視するディレクトリへのパス。

depth

このディレクトリから最大 *depth* 階層まで掘り下げたディレクトリが検索対象となります。

mask

シェルパターン *mask* により、検索するファイル名を制限します。

返り値

モニタリングリソース、あるいはエラー時に **FALSE** を返します。

参考

- fam_monitor_file()
- fam_monitor_directory()
- fam_cancel_monitor()
- fam_suspend_monitor()
- fam_resume_monitor()

---

fam_monitor_directory

(PHP 5 <= 5.0.5)

fam_monitor_directory — ディレクトリの変更を監視する

説明

resource **fam_monitor_directory** (resource `$fam` , string `$dirname`)

指定したディレクトリおよびそれに 含まれる全てのファイルを監視するよう指示します。

指定したディレクトリのステータス(すなわち、このディレクトリに関する stat() の結果)またはその内容 (すなわち、このディレクトリに関する readdir() の結果) が変更される度に FAM イベントが生成されます。

発生しうるイベントコードの詳細は、このセクションの 定数 の欄にあります。

パラメータ

fam

fam_open() が返す、 FAM サービスへの接続を表すリソース。

dirname

監視するディレクトリへのパス。

返り値

モニタリングリソース、あるいはエラー時に **FALSE** を返します。

参考

- fam_monitor_file()
- fam_monitor_collection()
- fam_cancel_monitor()
- fam_suspend_monitor()
- fam_resume_monitor()

---

fam_monitor_file

(PHP 5 <= 5.0.5)

fam_monitor_file — 通常のファイルの変更を監視する

説明

resource **fam_monitor_file** (resource `$fam` , string `$filename`)

指定した 1 個のファイルの 監視を指示します。このファイルのステータス(すなわち、このファイルに関する 関数 stat() の結果)が変化する度に FAM イベントが 発生します。

発生しうるイベントコードの詳細は、このセクションの 定数 の欄にあります。

パラメータ

fam

fam_open() が返す、 FAM サービスへの接続を表すリソース。

filename

監視するファイルへのパス。

返り値

モニタリングリソース、あるいはエラー時に **FALSE** を返します。

参考

- fam_monitor_directory()
- fam_monitor_collection()
- fam_cancel_monitor()
- fam_suspend_monitor()
- fam_resume_monitor()

---

fam_next_event

(PHP 5 <= 5.0.5)

fam_next_event — 次の待機中の FAM イベントを返す

説明

array **fam_next_event** (resource `$fam`)

次の待機中の FAM イベントを返します。

この関数は、fam_pending() を用いてチェックできる イベントが検出されるまでブロックします。

パラメータ

`fam`

fam_open() が返す、 FAM サービスへの接続を表すリソース。

返り値

FAM イベントコードを要素 '*code*' に、このイベントが適用されるファイルのパスを 要素 '*filename*' に、 オプションでホスト名を要素 '*hostname*' に含む配列を返します。

発生しうるイベントコードの詳細は、このセクションの 定数 の欄にあります。

参考

- fam_pending()

---

fam_open

(PHP 5 <= 5.0.5)

fam_open — FAM デーモンへの接続をオープンする

説明

resource **fam_open** ([string `$appname`])

FAM サービスデーモンへの接続をオープンします。

パラメータ

`appname`

ログ記録用にアプリケーションを特定する文字列。

返り値

成功した場合に FAM サービスへの接続を表すリソース、 失敗した場合に **`FALSE`** を返します。

参考

- fam_close()

---

fam_pending

(PHP 5 <= 5.0.5)

fam_pending — 待機中の FAM イベントの有無を調べる

説明

int **fam_pending** (resource `$fam`)

待機中の FAM イベントを調べます。

パラメータ

`fam`

fam_open() が返す、 FAM サービスへの接続を表すリソース。

返り値

fam_next_event() により取得可能な待機中のイベントが ある場合にゼロ以外の値、ない場合にゼロを返します。

参考

- fam_next_event()

---

fam_resume_monitor

(PHP 5 <= 5.0.5)

fam_resume_monitor — 中断された監視処理を再開する

説明

bool **fam_resume_monitor** (resource `$fam` , resource `$fam_monitor`)

fam_suspend_monitor() により中断された 指定したリソースへの監視を再開します。

パラメータ

`fam`

fam_open() が返す、 FAM サービスへの接続を表すリソース。

`fam_monitor`

fam_monitor_XXX 関数のいずれかが返すリソース。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- fam_suspend_monitor()

---

fam_suspend_monitor

(PHP 5 <= 5.0.5)

fam_suspend_monitor — 監視を一時的に中断する

説明

bool **fam_suspend_monitor** (resource `$fam` , resource `$fam_monitor`)

fam_suspend_monitor() は、リソースへの監視を一時的に中断します。

監視は、fam_resume_monitor() により後で再開することができ、 その際、全く新たに監視を指示する必要はありません。

パラメータ

fam

fam_open() が返す、 FAM サービスへの接続を表すリソース。

fam_monitor

fam_monitor_XXX 関数のいずれかが返すリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fam_cancel_monitor()
- fam_resume_monitor()

目次

Forms Data Format 関数

導入

Forms Data Format (FDF)は、PDF ドキュメント内部のフォームを処理する ためのフォーマットです。FDF の詳細および一般的な用途については、 » http://partners.adobe.com/asn/acrobat/forms.jsp にある ドキュメントを参照ください。

FDF の基本的な考えは HTML フォームに似ています。基本的な違いは、 投稿ボタンが押された際にフォームに埋められたデータをサーバに送信する 方法のフォーマット(これが Form Data Format そのものです)およびフォームの フォーマット自体(これは Portable Document Format、PDF です)です。 FDF データの処理は、fdf 関数が提供する機能の一つです。しかし他にも 機能はあります。既存の PDF フォームを用いてフォーム自体を修正せずに 入力フィールドのデータを入力させることもあるかと思われます。 このような場合、FDF ドキュメント を作成し (fdf_create())、各入力フィールドの値を設定し (fdf_set_value())、PDF フォームとそれを関連付けます (fdf_set_file())。最後にブラウザに MIME 型 *application/vnd.fdf* で送信する必要があります。 ブラウザの Acrobat Reader プラグイン がこの MIME 型を認識し、 関連する PDF フォームを読み込み、FDF ドキュメントからのデータを 書き込みます。

テキストエディタで FDF ドキュメントの中を見てみると、 *FDF* という名前のカタログオブジェクトが見つかる はずです。このようなオブジェクトには *Fields*、 *F*、*Status* といった複数の エントリが含まれます。 最も多く使用されるエントリは *Fields* および *F* で、*Fields* は 一連の入力フィールドを指し、*F* は このデータが属する PDF ドキュメントのファイル名を保持します。 これらのエントリは、FDF ドキュメントの中で /F-Key または /Status-Key として参照されています。fdf_set_file() および fdf_set_status() のような関数により、これらの エントリを修正することが可能です。Fields は、 fdf_set_value()、fdf_set_opt() 等で修整可能です。

要件

» http://partners.adobe.com/asn/acrobat/forms.jsp から取得可能な FDF toolkit SDKが必要です。 PHP 4.3.0 以降では、少なくとも SDK version 5.0 が必要です。 FDF toolkit library はバイナリ版のみ利用可能で、 Adobe によりサポートされているプラットフォームは Win32、Linux、Solaris、AIX です。

インストール手順

--with-fdftk[=DIR] を指定して PHP を コンパイルする必要があります。

> **注意**: fdftk サポートを指定して PHP を設定した際に問題を発生した場合、 ヘッダファイル *fdftk.h* および ライブラリ *libfdftk.so* が正しい場所にあることを 確認してください。configure スクリプトは、FDF SDK 配布ファイルおよび 通常の *DIR/include*、 *DIR/lib* といった配置のディレクトリ構造をともに サポートします。このため、配布ファイルを展開したディレクトリにそのまま おいておくか、あるいは */usr/local/include* および */usr/local/lib* のように使用するプラットフォームに 応じた適当なディレクトリに移動して --with-fdftk=/usr/local を指定して 設定することが可能です。

> **注意**: **Win32 ユーザへの注意** この拡張モジュールを動作させるには、 Windows システムの *PATH* が通った場所に DLL ファイルが存在する必要があります。 FAQ の "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" で、その方法を説明しています。 DLL ファイルを PHP のフォルダから Windows のシステムディレクトリにコピーしても動作します (システムディレクトリは、デフォルトで *PATH* に含まれるからです) が、これは推奨しません。 *この拡張モジュールを使用するには、以下のファイルが PATH の通った場所にある必要があります。 fdftk.dll*

実行時設定

設定ディレクティブは定義されていません。

リソース型

fdf

多くの fdf 関数は、最初のパラメータとして *fdf* リソースを必要とします。 *fdf* リソースは、オープンした fdf ファイルの ハンドルです。*fdf* リソースは、 fdf_create()、fdf_open()、 fdf_open_string() を用いて取得することが可能です。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

FDFValue (integer)
FDFStatus (integer)
FDFFile (integer)
FDFID (integer)
FDFFf (integer)
FDFSetFf (integer)
FDFClearFf (integer)
FDFFlags (integer)
FDFSetF (integer)
FDFClrF (integer)
FDFAP (integer)
FDFAS (integer)
FDFAction (integer)
FDFAA (integer)
FDFAPRef (integer)
FDFIF (integer)
FDFEnter (integer)
FDFExit (integer)
FDFDown (integer)
FDFUp (integer)
FDFFormat (integer)
FDFValidate (integer)
FDFKeystroke (integer)
FDFCalculate (integer)
FDFNormalAP (integer)
FDFRolloverAP (integer)
FDFDownAP (integer)

例

以下の例でフォームデータの評価に関して説明します。

Example#1 FDF ドキュメントの評価

```
<?php
// 拡張モジュールが提供する入力文字列から fdf をオープンします。
// pdf フォームには volume, date, comment, publisher, preparer
// という名前のテキスト入力フィールドと 2 つのチェックボックス
// show_publisher および show_preparer があるものとします。
$fdf = fdf_open_string($HTTP_FDF_DATA);
$volume = fdf_get_value($fdf, "volume");
echo "volume フィールドの値は '<b>$volume</b>'<br />";

$date = fdf_get_value($fdf, "date");
echo "date フィールドの値は '<b>$date</b>'<br />";

$comment = fdf_get_value($fdf, "comment");
echo "comment フィールドの値は '<b>$comment</b>'<br />";

if (fdf_get_value($fdf, "show_publisher") == "On") {
  $publisher = fdf_get_value($fdf, "publisher");
  echo "publisher フィールドの値は '<b>$publisher</b>'<br />";
} else
  echo "Publisher は非公開です。<br />";

if (fdf_get_value($fdf, "show_preparer") == "On") {
  $preparer = fdf_get_value($fdf, "preparer");
  echo "preparer フィールドの値は '<b>$preparer</b>'<br />";
} else
  echo "Preparer は非公開です。<br />";
fdf_close($fdf);
?>
```

---

fdf_add_doc_javascript

(PHP 4 >= 4.3.0, PHP 5)

fdf_add_doc_javascript — FDF ドキュメントに javascript コードを追加する

説明

bool **fdf_add_doc_javascript** (resource $fdf_document , string $script_name , string $script_code)

スクリプトを FDF に追加します。Acrobat は、FDF が読み込まれた際に それをドキュメントレベルのスクリプトとして追加します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

script_name

スクリプト名。

script_code

スクリプトのコード。コード内での改行は、できるだけ '\r' を使用するようにしてください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 JavaScript コードを FDF に追加する

```
<?php
$fdf = fdf_create();
fdf_add_doc_javascript($fdf, "PlusOne", "function PlusOne(x)¥r{¥r   return x+1;¥r}¥r");
fdf_save($fdf);
?>
```

これは、以下のような FDF を出力します。

```
%FDF-1.2
%âãÏÓ
1 0 obj
<<
/FDF << /JavaScript << /Doc [ (PlusOne)(function PlusOne¥(x¥)¥r{¥r   return x+1;¥r}¥r)] >> >>
>>
endobj
trailer
<<
/Root 1 0 R
```

```
>>
%%EOF
```

---

fdf_add_template

(PHP 4, PHP 5)

fdf_add_template — テンプレートを FDF ドキュメントに追加する

説明

bool **fdf_add_template** (resource `$fdf_document` , int `$newpage` , string `$filename` , string `$template` , int `$rename`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

fdf_close

(PHP 4, PHP 5)

fdf_close — FDF ドキュメントを閉じる

説明

void **fdf_close** (resource `$fdf_document`)

FDF ドキュメントを閉じます。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

返り値

値を返しません。

参考

- fdf_open()

---

fdf_create

(PHP 4, PHP 5)

fdf_create — 新規 FDF ドキュメントを作成する

説明

resource **fdf_create** (void)

新規 FDF ドキュメントを作成します。

この関数は、PDF ドキュメントの入力フィールドにデータを書きこみたい場合に必要です。

返り値

FDF ドキュメントハンドル、あるいはエラー時に **FALSE** を返します。

例

Example#1 PDF ドキュメントを公開する

```
<?php
$outfdf  = fdf_create();
fdf_set_value($outfdf, "volume", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type:  application/vnd.fdf");
$fp  = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

参考

- fdf_close()
- fdf_save()
- fdf_open()

---

fdf_enum_values

(PHP 4 >= 4.3.0, PHP 5)

fdf_enum_values — 各ドキュメントの値に対してユーザ定義関数をコールする

説明

bool **fdf_enum_values** (resource `$fdf_document` , callback `$function` [, mixed `$userdata`])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

fdf_errno

(PHP 4 >= 4.3.0, PHP 5)

fdf_errno — 直近の fdf 操作に関するエラーコードを返す

説明

int **fdf_errno** (void)

直近の FDF 関数コールによって設定されたエラーコードを取得します。

エラーの内容についてのテキストを取得するには fdf_error() を使用します。

返り値

エラーコードを表す整数値、あるいはエラーがない場合にゼロを返します。

参考

- fdf_error()

---

fdf_error

(PHP 4 >= 4.3.0, PHP 5)

fdf_error — 直近の fdf エラーコードについての説明を返す

説明

string **fdf_error** ([int `$error_code`])

error_code で指定したエラーコードについての説明テキストを取得します。

パラメータ

error_code

fdf_errno() で取得したエラーコード。 省略した場合は、直近の操作で設定された内部エラーコードを使用します。

返り値

エラーメッセージを表す文字列、あるいは何も問題がない場合は *no error* を返します。

参考

- fdf_errno()

---

fdf_get_ap

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_ap — フィールドの外観を取得する

説明

bool **fdf_get_ap** (resource `$fdf_document` , string `$field` , int `$face` , string `$filename`)

field の外観 (つまり /AP キーの値) を取得し、ファイルに保存します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

field

face

FDFNormalAP、**FDFRolloverAP** および **FDFDownAP** のいずれか。

filename

ここに外観を保存します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

fdf_get_attachment

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_attachment — FDF に埋め込まれている、アップロードされたファイルを展開する

説明

array **fdf_get_attachment** (resource `$fdf_document` , string `$fieldname` , string `$savepath`)

「ファイル選択」フィールド *fieldname* によって アップロードされたファイルを展開し、それを *savepath* に保存します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

fieldname

savepath

ファイル名または既存のディレクトリ名を指定し、ディレクトリ名を 指定した場合はその下に元のファイル名で保存されます。 同名のファイルが存在した場合は上書きします。

注意: *savepath* にディレクトリを使用してファイルを 保存してその名前を調べる以外には、元のファイル名を知る方法はないと 思われます。

返り値

返される配列は以下のようなフィールドを保持します。

- *path* - ファイルが保存された場所
- *size* - 保存されたファイルのバイト数
- *type* - (もし FDF 指定されていた場合) FDF の mimetype

例

Example#1 アップロードされたファイルを保存する

```
<?php
  $fdf = fdf_open_string($HTTP_FDF_DATA);
  $data  = fdf_get_attachment($fdf, "filename", "/tmpdir");
  echo "アップロードされたファイルが  $data[path]  に保存されました。";
?>
```

---

fdf_get_encoding

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_encoding — /Encoding キーの値を取得する

説明

string **fdf_get_encoding** (resource `$fdf_document`)

/Encoding キーの値を取得します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

返り値

エンコーディングを文字列で返します。 デフォルトの *PDFDocEncoding/Unicode* スキームを使用している場合は空の文字列を返します。

参考

- fdf_set_encoding()

---

fdf_get_file

(PHP 4, PHP 5)

fdf_get_file — /F キーの値を得る

説明

string **fdf_get_file** (resource `$fdf_document`)

/F キーの値を取得します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

返り値

キーの値を文字列を返します。

参考

- fdf_set_file()

---

fdf_get_flags

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_flags — フィールドのフラグを取得する

説明

int **fdf_get_flags** (resource `$fdf_document` , string `$fieldname` , int `$whichflags`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

fdf_get_opt

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_opt — フィールドのオプション配列から値を取得する

説明

mixed **fdf_get_opt** (resource `$fdf_document` , string `$fieldname` [, int `$element`])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

fdf_get_status

(PHP 4, PHP 5)

fdf_get_status — /STATUS キーの値を得る

説明

string **fdf_get_status** (resource `$fdf_document`)

/STATUS キーの値を取得します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

返り値

キーの値を文字列で返します。

参考

- fdf_set_status()

---

fdf_get_value

(PHP 4, PHP 5)

fdf_get_value — フィールドの値を得る

説明

mixed **fdf_get_value** (resource `$fdf_document` , string `$fieldname` [, int `$which`])

指定したフィールドの値を取得します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`fieldname`

FDF フィールドの名前を表す文字列。

`which`

このオプションパラメータを渡すことで、配列フィールドの 要素が取得可能です。番号はゼロから始まります。配列以外のフィールドでは、 このオプションは無視されます。

返り値

フィールドの値を返します。

変更履歴

バージョン	説明
4.3.0	配列のサポート、およびオプションのパラメータ *`which`* が追加されました。

参考

- fdf_set_value()

---

fdf_get_version

(PHP 4 >= 4.3.0, PHP 5)

fdf_get_version — FDF API あるいはファイルのバージョンを取得する

説明

string **fdf_get_version** ([resource `$fdf_document`])

指定したドキュメントの FDF バージョン、 あるいはパラメータを指定しなかった場合はツールキット API のバージョンを返します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

返り値

バージョンを表す文字列を返します。 現在の FDF ツールキット 5.0 では API のバージョン番号は *5.0* で、 ドキュメントのバージョン番号は *1.2 1.3* あるいは *1.4* です。

参考

- fdf_set_version()

---

fdf_header

(PHP 4 >= 4.3.0, PHP 5)

fdf_header — FDF 固有の出力ヘッダをセットする

説明

void **fdf_header** (void)

この関数は、簡便に FDF の出力に適当な HTTP ヘッダを付加できるよう 用意されており、*Content-type:* を *application/vnd.fdf* に設定します。

返り値

値を返しません。

---

fdf_next_field_name

(PHP 4, PHP 5)

fdf_next_field_name — 次のフィールド名を得る

説明

string **fdf_next_field_name** (resource `$fdf_document` [, string `$fieldname`])

指定したフィールドの後のフィールドの名前を返します。 この名前をその他の関数で使用します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`fieldname`

FDF フィールド名を表す文字列。 省略した場合は最初のフィールドとみなされます。

返り値

フィールド名を文字列で返します。

例

Example#1 FDF のすべてのフィールド名を検出する

```
<?php
$fdf = fdf_open($HTTP_FDF_DATA);
for ($field = fdf_next_field_name($fdf);
     $field != "";
     $field = fdf_next_field_name($fdf, $field)) {
     echo "field: $field¥n";
}
?>
```

参考

- fdf_get_value()

---

fdf_open_string

(PHP 4 >= 4.3.0, PHP 5)

fdf_open_string — 文字列から FDF ドキュメントを読み込む

説明

resource **fdf_open_string** (string `$fdf_data`)

文字列からデータを読み込みます。

fdf_open_string() を *$HTTP_FDF_DATA* とあわせて使用することで、リモート クライアントからの FDF フォーム入力を処理することが可能です。

パラメータ

fdf_data

fdf_create()、 fdf_open() あるいは **fdf_open_string()** が返す FDF ドキュメントハンドル。

返り値

FDF ドキュメントハンドル、あるいはエラー時に **FALSE** を返します。

例

Example#1 フォームデータへのアクセス

```
<?php
$fdf = fdf_open_string($HTTP_FDF_DATA);
/* ... */
fdf_close($fdf);
?>
```

参考

- fdf_open()
- fdf_close()
- fdf_create()
- fdf_save_string()

---

fdf_open

(PHP 4, PHP 5)

fdf_open — FDF ドキュメントをオープンする

説明

resource **fdf_open** (string `$filename`)

フォームデータを含むファイルをオープンします。

fdf_open_string() を用いて、 PDF フォームの POST リクエストを処理することもできます。

パラメータ

filename

FDF ファイルへのパス。このファイルは、PDF フォームから返されたものか、 fdf_create() および fdf_save() で作成したものである必要があります。

返り値

FDF ドキュメントのハンドル、あるいはエラー時に **FALSE** を返します。

例

Example#1 フォームデータへのアクセス

```
<?php
// FDF データをテンポラリファイルに保存します
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// テンポラリファイルをオープンし、データを評価します
$fdf = fdf_open("test.fdf");
/* ... */
fdf_close($fdf);
?>
```

参考

- fdf_open_string()
- fdf_close()
- fdf_create()
- fdf_save()

---

fdf_remove_item

(PHP 4 >= 4.3.0, PHP 5)

fdf_remove_item — フォームのターゲットフレームを設定する

説明

bool **fdf_remove_item** (resource `$fdf_document` , string `$fieldname` , int `$item`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

fdf_save_string

(PHP 4 >= 4.3.0, PHP 5)

fdf_save_string — FDF ドキュメントを文字列として返す

説明

string **fdf_save_string** (resource `$fdf_document`)

FDF ドキュメントを文字列として返します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

返り値

ドキュメントを表す文字列、あるいはエラー時に **FALSE** を返します。

例

Example#1 FDF の文字列での取得

```
<?php
$fdf = fdf_create();
fdf_set_value($fdf, "foo", "bar");
$str = fdf_save_string($fdf);
fdf_close($fdf);
echo $str;
?>
```

上の例の出力は以下となります。

```
%FDF-1.2
%âãÐÓ
1 0 obj
<<
/FDF << /Fields 2 0 R >>
>>
endobj
2 0 obj
[
<< /T (foo)/V (bar)>>
]
endobj
trailer
<<
/Root 1 0 R

>>
%%EOF
```

参考

- fdf_open_string()
- fdf_close()
- fdf_create()

- fdf_save()

---

fdf_save

(PHP 4, PHP 5)

fdf_save — FDF ドキュメントを保存する

説明

bool **fdf_save** (resource `$fdf_document` [, string `$filename`])

FDF ドキュメントを保存します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`filename`

指定した場合は、結果の FDF がここに書き込まれます。 それ以外の場合は、この関数は FDF をデフォルトの PHP 出力ストリームに書き込みます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fdf_close()
- fdf_create()
- fdf_save_string()

---

fdf_set_ap

(PHP 4, PHP 5)

fdf_set_ap — フィールドの外観を設定する

説明

bool **fdf_set_ap** (resource `$fdf_document` , string `$field_name` , int `$face` , string `$filename` , int `$page_number`)

フィールドの外観 (すなわち、*/AP* キーの値) を設定します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`field_name`

`face`

FDFNormalAP、 **FDFRolloverAP** および **FDFDownAP** のいずれかが使用可能です。

`filename`

`page_number`

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

fdf_set_encoding

(PHP 4 >= 4.0.7, PHP 5)

fdf_set_encoding — FDF 文字エンコーディングを設定する

説明

bool **fdf_set_encoding** (resource `$fdf_document` , string `$encoding`)

FDF ドキュメントの文字エンコーディングを設定します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`encoding`

エンコーディング名。現在サポートしているのは "*Shift-JIS*"、"*UHC*"、 "*GBK*"、"*BigFive*" です。

空の文字列を指定すると、エンコーディングをデフォルトの *PDFDocEncoding/Unicode* スキームに設定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

fdf_set_file

(PHP 4, PHP 5)

fdf_set_file — FDF データを表示する PDF ドキュメントを設定する

説明

bool **fdf_set_file** (resource `$fdf_document` , string `$url` [, string `$target_frame`])

フォームの結果を表示する PDF ドキュメントとして、もとのフォームとは別のものを指定します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`url`

絶対 URL で指定する必要があります。

`target_frame`

このパラメータを使用して、ドキュメントを表示するフレームを指定します。 このパラメータのデフォルト値を fdf_set_target_frame() で指定することもできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 FDF データを 2 番目のフォームに渡す

```
<?php
  /* Adobe FDF の content type を設定します */
  fdf_header();

  /* fdf を開始します */
  $fdf = fdf_create();

  /* "foo" フィールドに、値 "bar" を設定します */
  fdf_set_value($fdf, "foo", "bar");
```

```
  /* "fdf_form.pdf" を使用して fdf を表示することをクライアントに通知します */
  fdf_set_file($fdf, "http://www.example.com/fdf_form.pdf");

  /* fdf を出力します */
  fdf_save($fdf);

  /* 後始末を行います */
  fdf_close($fdf);
?>
```

参考

- fdf_get_file()
- fdf_set_target_frame()

---

fdf_set_flags

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_flags — フィールドのフラグを設定する

説明

bool **fdf_set_flags** (resource $fdf_document , string $fieldname , int $whichFlags , int $newFlags)

指定したフィールドのフラグを設定します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

fieldname

FDF フィールド名を表す文字列。

whichFlags

newFlags

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fdf_set_opt()

---

fdf_set_javascript_action

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_javascript_action — フィールドの javascript アクションを設定する

説明

bool **fdf_set_javascript_action** (resource $fdf_document , string $fieldname , int $trigger , string $script)

指定したフィールドの javascript アクションを設定します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

fieldname

FDF フィールド名を表す文字列。

trigger

script

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fdf_set_submit_form_action()

---

fdf_set_on_import_javascript

(PHP 4 >= 4.3.0, PHP 5)

fdf_set_on_import_javascript — Acrobat が FDF をオープンした際に実行される javascript のコードを追加する

説明

bool **fdf_set_on_import_javascript** (resource `$fdf_document` , string `$script` , bool `$before_data_import`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- fdf_add_doc_javascript()
- fdf_set_javascript_action()

---

fdf_set_opt

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_opt — フィールドのオプションを設定する

説明

bool **fdf_set_opt** (resource `$fdf_document` , string `$fieldname` , int `$element` , string `$str1` , string `$str2`)

指定したフィールドのオプションを設定します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

fieldname

FDF フィールド名を表す文字列。

element

str1

str2

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fdf_set_flags()

fdf_set_status

(PHP 4, PHP 5)

fdf_set_status — /STATUS キーの値を設定する

説明

bool **fdf_set_status** (resource `$fdf_document` , string `$status`)

/STATUS キーの値を設定します。 クライアントが FDF とともにステータスセットを受信すると、その内容を アラートボックスに表示します

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`status`

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fdf_get_status()

---

fdf_set_submit_form_action

(PHP 4 >= 4.0.2, PHP 5)

fdf_set_submit_form_action — フィールドの投稿フォームアクションを設定する

説明

bool **fdf_set_submit_form_action** (resource `$fdf_document` , string `$fieldname` , int `$trigger` , string `$script` , int `$flags`)

指定したフィールドの投稿フォームアクションを設定します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`fieldname`

FDF フィールド名を表す文字列。

`trigger`

`script`

`flags`

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fdf_set_javascript_action()

---

fdf_set_target_frame

(PHP 4 >= 4.3.0, PHP 5)

fdf_set_target_frame — フォームの表示対象となるフレームを設定する

説明

bool **fdf_set_target_frame** (resource `$fdf_document` , string `$frame_name`)

fdf_save_file() で定義した PDF の結果を出力する 対象のフレームを設定します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`frame_name`

フレーム名を表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- **fdf_save_file()**

---

fdf_set_value

(PHP 4, PHP 5)

fdf_set_value — フィールドの値を設定する

説明

bool **fdf_set_value** (resource `$fdf_document` , string `$fieldname` , mixed `$value` [, int `$isName`])

指定したフィールドに値 *`value`* を設定します。

パラメータ

`fdf_document`

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

`fieldname`

FDF フィールド名を表す文字列。

`value`

このパラメータは、配列形式でない限り文字列で保存されます。 配列の場合は、配列のすべての要素が配列として保存されます。

`isName`

注意: FDF ツールキットの古いバージョンでは、最後のパラメータは フィールドの値を PDF Name に変換する (= 1) か、PDF String に設定する (= 0) かを定義します。
ツールキットの現在のバージョン 5.0 では、もはやこのパラメータは 使用されません。PHP 4.3 以降、互換性を確保するためにこのパラメータが オプションとして残されていますが、内部ではこのパラメータは無視されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	*`value`* で配列をサポートするようになりました。

参考

- fdf_get_value()
- fdf_remove_item()

---

fdf_set_version

(PHP 4 >= 4.3.0, PHP 5)

fdf_set_version — FDF ファイルのバージョン番号を設定する

説明

bool **fdf_set_version** (resource `$fdf_document` , string `$version`)

指定したドキュメントの FDF バージョンを *version* に設定します。

この拡張モジュールが提供する機能のうちのいくつかは、新しいバージョンの FDF でのみ動作します。

パラメータ

fdf_document

fdf_create()、 fdf_open() あるいは fdf_open_string() が返す FDF ドキュメントハンドル。

version

バージョンを表す文字列。 現在の FDF ツールキット 5.0 では、これは *1.2*、 *1.3* あるいは *1.4* となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fdf_get_version()

目次

- fdf_set_flags — フィールドのフラグを設定する
- fdf_set_javascript_action — フィールドの javascript アクションを設定する
- fdf_set_on_import_javascript — Acrobat が FDF をオープンした際に実行される javascript のコードを追加する
- fdf_set_opt — フィールドのオプションを設定する
- fdf_set_status — /STATUS キーの値を設定する
- fdf_set_submit_form_action — フィールドの投稿フォームアクションを設定する
- fdf_set_target_frame — フォームの表示対象となるフレームを設定する
- fdf_set_value — フィールドの値を設定する
- fdf_set_version — FDF ファイルのバージョン番号を設定する

Fileinfo 関数

導入

このモジュールの関数は、ファイル内の特定の位置から *magic* バイトシーケンスを見つけることで、 ファイルの content type とエンコーディングを推測します。 これは完全な手法ではありませんが、経験上かなりうまく動作しています。

要件

この拡張モジュールをビルドするには *magic_open* ライブラリが必要です。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 »
http://pecl.php.net/package/fileinfo

実行時設定

設定ディレクティブは定義されていません。

リソース型

Fileinfo 拡張モジュールでは、一種類のリソースが使用されています。 それは、finfo_open() が返す magic データベース記述子です。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

FILEINFO_NONE (integer)
特別な処理を行いません。
FILEINFO_SYMLINK (integer)
シンボリックリンクのリンク先をたどります。
FILEINFO_MIME (integer)
テキスト表現ではなく、mime 文字列を返します。
FILEINFO_COMPRESS (integer)
圧縮されたファイルを伸張します。
FILEINFO_DEVICES (integer)
ブロックデバイスあるいはキャラクタデバイスの内容を探します。
FILEINFO_CONTINUE (integer)
最初に見つかったものだけでなく、一致するものをすべて返します。
FILEINFO_PRESERVE_ATIME (integer)
可能な限り、元の最終アクセス時刻を保持します。
FILEINFO_RAW (integer)
表示できない文字を ¥*ooo* 形式の 8 進表現に変換しません。

---

finfo_buffer

(PECL fileinfo:0.1-1.0.4)

finfo_buffer — 文字列バッファの情報を返す

説明

string **finfo_buffer** (resource $finfo , string $string [, int $options [, resource $context]])
finfo
string **buffer** (string $string [, int $options [, resource $context]])

この関数は、バイナリデータの情報を文字列形式で返すために使用します。

パラメータ

finfo

finfo_open() が返す fileinfo リソース。

string

調べるファイルの内容。

options

ひとつあるいは複数のFileinfo 定数の組み合わせ。

context

返り値

string のテキスト表現、あるいはエラーが発生した場合に **FALSE** を返します。

例

Example#1 finfo_buffer() の例

```
<?php
$finfo = new finfo(FILEINFO_MIME);
echo $finfo->buffer($_POST["script"]) . "¥n";
?>
```

上の例の出力は、たとえば以下のようになります。

```
application/x-sh
```

参考

- finfo_file()

---

finfo_close

(PECL fileinfo:0.1-1.0.4)

finfo_close — fileinfo リソースを閉じる

説明

bool **finfo_close** (resource $finfo)
finfo
__destruct (void)

この関数は、finfo_open() がオープンしたリソースを閉じます。

パラメータ

finfo

finfo_open() が返す fileinfo リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

finfo_file

(PECL fileinfo:0.1-1.0.4)

finfo_file — ファイルについての情報を返す

説明

string **finfo_file** (resource `$finfo` , string `$file_name` [, int `$options` [, resource `$context`]])
finfo
string **file** (string `$file_name` [, int `$options` [, resource `$context`]])

この関数は、ファイルについての情報を返すために使用します。

パラメータ

`finfo`

finfo_open() が返す fileinfo リソース。

`file_name`

調べるファイルの名前。

`options`

ひとつあるいは複数のFileinfo 定数の組み合わせ。

`context`

contexts の説明については、ストリーム を参照ください。

返り値

`filename` のテキスト表現、あるいはエラーが発生した場合に **FALSE** を返します。

例

Example#1 finfo_file() の例

```
<?php
$finfo = finfo_open(FILEINFO_MIME); // return mime type ala mimetype extension
foreach (glob("*") as $filename) {
    echo finfo_file($finfo, $filename) . "¥n";
}
finfo_close($finfo);
?>
```

上の例の出力は、たとえば以下のようになります。

```
text/html
image/gif
application/vnd.ms-excel
```

参考

- finfo_buffer()

---

finfo_open

finfo->__construct()

(No version information available, might be only in CVS)

finfo->__construct() — 新しい fileinfo リソースを作成する

説明

手続き型

resource **finfo_open** ([int $options [, string $arg]])

オブジェクト指向型 (コンストラクタ)

finfo
__construct ([int $options [, string $magic_file]])

この関数は、magic データベースをオープンしてそのリソースを返します。

パラメータ

options

ひとつあるいは複数のFileinfo 定数の組み合わせ。

magic_file

magic データベースファイルの名前。通常は */path/to/magic.mime* のようになります。 指定されなかった場合は、環境変数 *MAGIC* の値が使用されます。この環境変数も設定されていない場合、 */usr/share/misc/magic* をデフォルトで使用します。 必要に応じて、*.mime* や *.mgc* が追加されます。

返り値

成功した場合に magic データベースリソース、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$finfo = new finfo(FILEINFO_MIME, "/usr/share/misc/magic"); // return mime type ala mimetype extension

if (!$finfo) {
    echo "fileinfo データベースのオープンに失敗しました";
     exit();
}

/* 指定したファイルの mime タイプを取得します */
$filename = "/usr/local/something.txt";
echo $finfo->file($filename);

/* 接続を閉じます */
$finfo->close();
?>
```

Example#2 手続き型

```
<?php
$finfo = finfo_open(FILEINFO_MIME, "/usr/share/misc/magic"); // return mime type ala mimetype extension

if (!$finfo) {
    echo "fileinfo データベースのオープンに失敗しました";
     exit();
}

/* 指定したファイルの mime タイプを取得します */
$filename = "/usr/local/something.txt";
echo finfo_file($finfo, $filename);

/* 接続を閉じます */
finfo_close($finfo);
?>
```

上の例の出力は以下となります。

```
text/plain
```

参考

- finfo_close()

---

finfo_set_flags

(PECL fileinfo:0.1-1.0.4)

finfo_set_flags — libmagic のオプションを設定する

説明

bool **finfo_set_flags** (resource `$finfo` , int `$options`)
finfo
bool **set_flags** (int `$options`)

この関数は、さまざまな Fileinfo オプションを設定します。 finfo_open() やその他の Fileinfo 関数で、 オプションを直接指定することも可能です。

パラメータ

`finfo`

finfo_open() が返す fileinfo リソース。

`options`

ひとつあるいは複数のFileinfo 定数の組み合わせ。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

目次

---

filePro 関数

導入

これらの関数により、filePro データベースに保存されたデータに 読み込みのみのアクセスが可能になります。

filePro は、fP Technologies 社の登録商標です。filePro に関する詳細な情報は » http://www.fptech.com/ で得る事が出来ます。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.2.0.

インストール手順

PHP の filePro サポートは、デフォルトでは有効となっていません。 組み込みの *read-only* filePro サポートを 有効にするには、PHP をコンパイルする際に設定オプション --enable-filepro を指定する必要があります。

---

filepro_fieldcount

(PHP 4, PHP 5 <= 5.1.6)

filepro_fieldcount — filePro データベース中のフィールド数を返す

説明

int **filepro_fieldcount** (void)

オープンした filePro データベースのフィールド（カラム）の数を返します。

返り値

オープンした filePro データベースのフィールド数、 あるいはエラーの場合に **FALSE** を返します。

参考

- filepro()

---

filepro_fieldname

(PHP 4, PHP 5 <= 5.1.6)

filepro_fieldname — フィールド名を取得する

説明

string **filepro_fieldname** (int `$field_number`)

`field_number` に対応するフィールド名を返します。

パラメータ

`field_number`

フィールド番号。

返り値

フィールド名を表す文字列、あるいはエラーの場合に **FALSE** を返します。

---

filepro_fieldtype

(PHP 4, PHP 5 <= 5.1.6)

filepro_fieldtype — フィールド型を取得する

説明

string **filepro_fieldtype** (int `$field_number`)

`field_number` に対応した編集フィールド型を返します。

パラメータ

`field_number`

フィールド番号。

返り値

編集フィールド型を表す文字列、あるいはエラーの場合に **FALSE** を返します。

---

filepro_fieldwidth

(PHP 4, PHP 5 <= 5.1.6)

filepro_fieldwidth — フィールド幅を取得する

説明

int **filepro_fieldwidth** (int `$field_number`)

`field_number` に対応するフィールドの幅を返します。

パラメータ

field_number

フィールド番号。

返り値

フィールドの幅を表す整数値、あるいはエラーの場合に **FALSE** を返します。

---

filepro_retrieve

(PHP 4, PHP 5 <= 5.1.6)

filepro_retrieve — filePro データベースからデータを取得する

説明

string **filepro_retrieve** (int `$row_number` , int `$field_number`)

データベースの指定された位置から読み込んだデータを返します。

> **注意**: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

パラメータ

row_number

行番号。ゼロから 全行数マイナス 1 までの間である必要があります (0..filepro_rowcount() - 1)。

field_number

フィールド番号。ゼロから 全フィールド数マイナス 1 までの間である必要があります (0..filepro_fieldcount() - 1)。

返り値

指定したデータ、あるいはエラーの場合に **FALSE** を返します。

---

filepro_rowcount

(PHP 4, PHP 5 <= 5.1.6)

filepro_rowcount — filePro データベースの行の数を返します

説明

int **filepro_rowcount** (void)

オープンされた filePro データベースにおける行の数を返します。

> **注意**: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

返り値

オープンされた filePro データベースにおける行の数を返します。 エラーの場合は **FALSE** を返します。

参考

- filepro()

---

filepro

(PHP 4, PHP 5 <= 5.1.6)

filepro — map ファイルの読み込みと照合を行う

説明

bool **filepro** (string `$directory`)

この関数は、map ファイルを読み込み、照合を行います。 続いて、フィールドの数および情報を保存します。

データベースのロックを行わないため、PHP でオープンしている間は、 filePro データベースの修正を避ける必要があります。

> **注意**: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

パラメータ

`directory`

map ディレクトリ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

目次

ファイルシステム関数

導入

要件

この拡張モジュールを構築するには外部ライブラリを必要としませんが、 Linux 上で LFS (ラージファイル) をサポートする PHP を希望する場合は、最新の glibc を入手し、次のコンパイラフラグ *-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64* を付けて PHP をコンパイルする必要があります。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

ファイルシステムおよびストリーム設定オプション

名前	デフォルト	変更の可否	変更履歴
allow_url_fopen	"1"	PHP_INI_ALL	PHP <= 4.3.4 では PHP_INI_ALL、PHP < 6 では PHP_INI_SYSTEM。 PHP 4.0.4 から利用可能です。
allow_url_include	"0"	PHP_INI_ALL	PHP 5 では PHP_INI_SYSTEM。PHP 5.2.0 から利用可能です。
user_agent	NULL	PHP_INI_ALL	PHP 4.3.0 から利用可能です。
default_socket_timeout	"60"	PHP_INI_ALL	PHP 4.3.0 から利用可能です。
from	""	PHP_INI_ALL	
auto_detect_line_endings	"0"	PHP_INI_ALL	PHP 4.3.0 から利用可能です。

以下に設定ディレクティブに関する簡単な説明を示します。

allow_url_fopen boolean

このオプションにより、URL対応のfopenラッパーが使用可能となり、 ファイルのようにURLオブジェクトをアクセスできるようになります。 デフォルトのラッパーが、ftpまたはhttpプロトコルを用いて リモートファイルに アクセスするために提供されています。zlibのようないくつかの拡張モジュールが ラッパーを追加することがあります。

注意: この設定はセキュリティ上の理由で php.ini 中でのみ設定可能です。

注意: このオプションは、バージョン4.0.3のリリース直後に追加されました。 4.0.3を含む以前のバージョンでは、この機能は、設定スイッチ *--disable-url-fopen-wrapper* を使用することに より、コンパイル時にのみ無効にすることができます。

警告

PHP 4.3より前のWindows版では、以下の関数は、リモートファイルの アクセスをサポートしません。: include(), include_once(), require(), require_once(), イメージ 拡張モジュールの imagecreatefromXXX

allow_url_include boolean

このオプションを指定すると include()、include_once()、 require()、require_once() で URL 対応の fopen ラッパーが使用できるようになります。

注意: この設定を使用するには、allow_url_fopen が on でないといけません。

user_agent string

送信する PHP 用のユーザエージェントを定義します。

default_socket_timeout integer

ソケットベースのストリームのデフォルトの有効時間(単位は秒)を定義します。

注意: この設定は、PHP 4.3で追加されました。

from string

匿名ftp用パスワード(自分のemailアドレス)を定義します。

auto_detect_line_endings boolean

onにした場合、PHPは fgets() および file() により読み込まれたデータを評価し、UNIX、MS-DOS、Machintoshの行末 表記を使用しているかどうかを調べます。

これにより、PHPがMacintoshシステムと相互運用できるようになりますが、 デフォルトはOffとなっています。これは、最初の行の行末表記を検出 する際にごく僅かな性能劣化があるためと、UNIXシステムのもとで復改 文字を項目セパレータとして使用している人が従来のバージョンと互換 性がない動作であると感じる可能性があるためです。

注意: この設定オプションは、PHP 4.3で追加されました。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

GLOB_BRACE (integer)
GLOB_ONLYDIR (integer)
GLOB_MARK (integer)
GLOB_NOSORT (integer)
GLOB_NOCHECK (integer)
GLOB_NOESCAPE (integer)
PATHINFO_DIRNAME (integer)
PATHINFO_BASENAME (integer)
PATHINFO_EXTENSION (integer)
PATHINFO_FILENAME (integer)
PHP 5.2.0 以降。
FILE_USE_INCLUDE_PATH (integer)
filename を include_path から探します (PHP 5 以降)。
FILE_APPEND (integer)

既存のファイルに追記します。

FILE_IGNORE_NEW_LINES (integer)
EOL (行末) 文字を取り除きます (PHP 5 以降)。

FILE_SKIP_EMPTY_LINES (integer)
空行を読み飛ばします (PHP 5 以降)。

FILE_BINARY (integer)
バイナリモード (PHP 6 以降)。

FILE_TEXT (integer)
テキストモード (PHP 6 以降)。

参考

関連する関数については、ディレクトリ およびプログラム実行の節を 参照してください。

リモートファイルとして使用することができる種々のURLラッパーの一覧 と説明については、サポートされるプロトコル/ラッパーも参照してください。

---

basename

(PHP 4, PHP 5)

basename — パス中のファイル名の部分を返す

説明

string **basename** (string $path [, string $suffix])

この関数は、ファイルへのパスを有する文字列を引数とし、 ファイルのベース名を返します。

パラメータ

path

パス。

Windows では、スラッシュ (/) とバックスラッシュ (¥) の両方がディレクトリ区切り文字として使われます。 その他の環境ではスラッシュ (/) になります。

suffix

ファイル名が、 *suffix* で終了する場合、 この部分もカットされます。

返り値

指定した *path* のベース名を返します。

変更履歴

バージョン	説明
4.1.0	パラメータ *suffix* が追加されました。

例

Example#1 basename() の例

```
<?php
$path = "/home/httpd/html/index.php";
$file = basename($path);          // $file は "index.php" に設定される
$file = basename($path,".php");   // $file は "index" に設定される
?>
```

参考

- dirname()

---

chgrp

(PHP 4, PHP 5)

chgrp — ファイルのグループを変更する

説明

bool **chgrp** (string `$filename` , mixed `$group`)

(名前または番号で指定した)ファイル *filename* のグループを *group* に変更しようと試みます。

スーパーユーザのみがファイルのグループを任意に変更できます。 その他のユーザは、ファイルのグループをそのユーザがメンバーとして 属しているグループに変更できます。

パラメータ

filename

ファイルへのパス。

group

グループ名あるいはグループ番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

注意: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- chown()
- chmod()

---

chmod

(PHP 4, PHP 5)

chmod — ファイルのモードを変更する

説明

bool **chmod** (string `$filename` , int `$mode`)

指定されたファイルのモードを *mode* で指定したものに変更しようと試みます。

パラメータ

filename

ファイルへのパス。

mode

mode は自動的には 8 進数と見なされないので注意してください。 このため、("g+w" のような)文字列は正常に動作しません。 意図した操作を行うには、*mode* の前にゼロ(0)を付ける必要があります。

```
<?php
chmod("/somedir/somefile", 755);   // 10 進数; おそらく間違い
chmod("/somedir/somefile", "u+rwx,go+rx"); // 文字列; 正しくない
chmod("/somedir/somefile", 0755);  // 8 進数; 正しいモードの値
?>
```

mode 引数は 3 つの 8 進法による数値で構成され、 所有者自身、所有者が属するグループ、その他のユーザーの順で アクセス制限を設定します。一つ一つの数字はそのターゲットに対し 許可を与えます。1 は実行権限、2 はファイルに対する書き込み権限、 4 はファイルに対する読み

込み権限を与えます。 必要な権限にあわせ数値を加算してください。 許可モードに関する詳細は Unix システムの 「man 1 chmod」や「man 2 chmod」をご覧ください。

```
<?php
// 所有者に読み込み、書き込みの権限を与え、その他には何も許可しない。
chmod("/somedir/somefile", 0600);

// 所有者に読み込み、書き込みの権限を与え、その他には読み込みだけ許可する。
chmod("/somedir/somefile", 0644);

// 所有者に全ての権限を与え、その他には読み込みと実行を許可する。
chmod("/somedir/somefile", 0755);

// 所有者に全ての権限を与え、所有者が属するグループに読み込みと実行を許可する。
chmod("/somedir/somefile", 0750);
?>
```

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: 現在のユーザは PHP を実行しているユーザです。 これは普通のシェルや FTP アクセスでのユーザとはたいてい違います。 たいていのシステムでは、ファイルの所有者のみがそのモードを 変更可能です。

注意: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

注意: セーフモード が有効な場合、操作しようとしているファイルあるいは ディレクトリの UID（所有者）がスクリプトの実行ユーザと同じかどうかを PHP がチェックします。さらに、SUID・SGID や sticky ビットを 設定することはできません。

参考

- chown()
- chgrp()

---

chown

(PHP 4, PHP 5)

chown — ファイルの所有者を変更する

説明

bool **chown** (string `$filename` , mixed `$user`)

ファイル*filename* の所有者を(名前または番号で指定した) ユーザ*user* に変更しようと試みます。 スーパーユーザのみがファイルの所有者を変更できます。

パラメータ

filename

ファイルへのパス。

user

ユーザ名あるいはユーザ番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

注意: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- chmod()

---

clearstatcache

(PHP 4, PHP 5)

clearstatcache — ファイルのステータスのキャッシュをクリアする

説明

void **clearstatcache** (void)

stat や lstat 、 またはその他の関数（後述）を使用すると、PHPはパフォーマンス向上のために それらの関数の戻り値をキャッシュします。しかし、ケースによっては、 キャッシュされた情報を消去したい場合もあるでしょう。 例えば、一つのスクリプト上で同じファイルが何度もチェックされ、 そのファイルが変更されたり削除されたりする可能性がある場合、 ステータスキャッシュを消去しなければならないと感じるでしょう。 このようなケースでは、**clearstatcache()**を使用することで ファイルの情報に関してPHPが持っているキャッシュをクリアすることができます。

PHP は存在しないファイルについての情報はキャッシュしないことにも 注意してください。もし存在しないファイルに対して file_exists() をコールする場合、ファイルを作成するまで この関数は **FALSE** を返します。もしファイルを作成した場合、 たとえファイルを削除したとしても **TRUE** を返します。 しかし、unlink() はキャッシュを自動的にクリアします。

> **注意**: この関数は特定のファイルに関する情報をキャッシュします。したがって、 同じファイルについて複数回の操作を行いそのファイルに関する情報を キャッシュされないようにするためには、 **clearstatcache()**をコールするだけです。

影響を受ける関数を以下に示します。 stat(), lstat(), file_exists(), is_writable(), is_readable(), is_executable(), is_file(), is_dir(), is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(), filetype(), および fileperms().

返り値

値を返しません。

---

copy

(PHP 4, PHP 5)

copy — ファイルをコピーする

説明

bool **copy** (string $source , string $dest [, resource $context])

ファイル *source* を *dest* にコピーします。

ファイルを移動したいならは、rename() 関数を使用してください。

パラメータ

source

コピー元ファイルへのパス。

dest

コピー先のパス。*dest* が URL の場合、 既存ファイルへの上書きをラッパーがサポートしていない場合にはコピーが失敗します。

警告

コピー先のファイルが既に存在する場合、上書きされます。

context

stream_context_create() で作成した有効なコンテキストリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.3.0	コンテキストのサポートが追加されました。
4.3.0	"fopen wrappers" が有効の場合は *source* と *dest* のどちらにも URL を指定することができます。 詳細は fopen() を参照ください。

例

Example#1 copy() の例

```
<?php
$file = 'example.txt';
$newfile = 'example.txt.bak';

if (!copy($file, $newfile)) {
    echo "failed to copy $file...¥n";
}
?>
```

参考

- move_uploaded_file()
- rename()
- マニュアルの ファイルアップロード処理

---

delete

(PECL zip:1.1.0-1.4.1)

delete — unlink() か unset() を参照してください

説明

void **delete** (void)

この関数はダミーの関数エントリであり、間違った場所で unlink() または unset() を要求する人の要求を満足させるためのものです。

返り値

値を返しません。

参考

- ファイルを削除するには unlink()
- 変数を削除するには unset()

---

dirname

(PHP 4, PHP 5)

dirname — パス中のディレクトリ名の部分を返す

説明

string **dirname** (string `$path`)

この関数は、ファイルへのパス名を有する文字列を引数とし、 ディレクトリの名前を返します。

パラメータ

path

パス。

Windows では、スラッシュ(/)とバックスラッシュ (¥)の両方がディレクトリ区切り文字として使われます。 その他の環境ではスラッシュ(/)になります。

返り値

ディレクトリの名前を返します。 *path* にスラッシュが無い場合は、 カレントディレクトリを示すドット ('.') を返します。それ以外の場合は、スラッシュ以降の */component* 部分を取り除いた *path* を返します。

変更履歴

バージョン	説明
5.0.0	**dirname()** がバイナリセーフとなりました。
4.0.3	**dirname()** が POSIX 準拠となりました。

例

Example#1 dirname() の例

```
<?php
$path = "/etc/passwd";
$file = dirname($path); // $file は "/etc" となります
?>
```

注意

> **注意**: PHP 4.3.0 以降では、これまでは **dirname()** が空文字列を返していたような状況で スラッシュやドットを返すことが多くなりました。

以下の例で変更点をご確認ください。

```
<?php

// PHP 4.3.0 より前のバージョン
dirname('c:/'); // '.' を返します

// PHP 4.3.0 以降のバージョン
dirname('c:/x'); // 'c:\' を返します
dirname('c:/Temp/x'); // 'c:/Temp' を返します
dirname('/x'); // '\' を返します

?>
```

参考

- basename()
- pathinfo()
- realpath()

---

disk_free_space

(PHP 4 >= 4.0.7, PHP 5)

disk_free_space — ディレクトリの利用可能なスペースを返す

説明

float **disk_free_space** (string `$directory`)

ディレクトリを指定することにより、 この関数は対応するファイルシステムまたはディスクパーティションで 利用可能なバイト数を返します。

パラメータ

directory

> ファイルシステムのディレクトリあるいはディスクパーティション。
>
> > **注意**: ディレクトリのかわりにファイル名を指定したときの挙動は未定義です。 OS や PHP のバージョンによって結果は異なります。

返り値

利用可能なバイト数を float 型で返します。

例

Example#1 disk_free_space() の例

```
<?php
```

```
// $df は「/」で利用可能なバイト数となります。
$df = disk_free_space("/");

// Windows の場合:
disk_free_space("C:");
disk_free_space("D:");
?>
```

注意

> **注意**: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

参考

- disk_total_space()

---

disk_total_space

(PHP 4 >= 4.0.7, PHP 5)

disk_total_space — ディレクトリの全体サイズを返す

説明

float **disk_total_space** (string `$directory`)

ディレクトリを含む文字列を指定してください。この関数は、 ファイルシステムまたはディスクパーティションに対応する全体バイト数を返します。

パラメータ

`directory`

> ファイルシステムのディレクトリあるいはディスクパーティション。

返り値

総バイト数を float 型で返します。

例

Example#1 disk_total_space() の例

```
<?php
// $df は、「/」で利用可能な全体バイト数
$df = disk_total_space("/");

// Windows の場合:
disk_total_space("C:");
disk_total_space("D:");
?>
```

注意

> **注意**: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

参考

- disk_free_space()

---

diskfreespace

(PHP 4, PHP 5)

diskfreespace — disk_free_space() のエイリアス

説明

この関数は次の関数のエイリアスです。 disk_free_space().

---

fclose

(PHP 4, PHP 5)

fclose — オープンされたファイルポインタをクローズする

説明

bool **fclose** (resource $handle)

handle が指しているファイルをクローズします。

パラメータ

handle

ファイルポインタは有効なものでなければならず、また fopen()またはfsockopen() で正常にオープンされたファイルを指している必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 シンプルな fclose() の例

```
<?php
$handle = fopen('somefile.txt', 'r');
fclose($handle);
?>
```

参考

- fopen()
- fsockopen()

---

feof

(PHP 4, PHP 5)

feof — ファイルポインタがファイル終端に達しているかどうか調べる

説明

bool **feof** (resource $handle)

ファイルポインタがファイル終端に達しているかどうかを調べます。

パラメータ

handle

ファイルポインタは、有効なファイルポインタである必要があり、 fopen() または fsockopen() で正常にオープンされた（そしてまだ fclose() でクローズされていない）ファイルを指している必要があります。

返り値

ファイルポインタが EOF に達しているかまたはエラー (ソケットタイムアウトを含みます) の場合に **TRUE** 、 その他の場合に **FALSE** を返します。

注意

警告

もし fsockopen() でオープンされた接続がサーバによって 閉じられていない場合、**feof()** はタイムアウトになるまで待ち、**TRUE** を返します。 デフォルトのタイムアウト値は 60 秒です。この値を変更するには stream_set_timeout() を使用します。

警告

無効なファイルポインタを渡した場合、無限ループに陥ることがあります。 なぜなら EOF が TRUE を返すことができないからです。

Example#1 feof() に無効なファイルポインタを使用する例

```
<?php
// ファイルを読み込めなかったりファイルが存在しなかったりした場合、
// fopen 関数は FALSE を返します。
$file = @fopen("no_such_file", "r");

// fopen からの FALSE が警告を発生させ、ここで無限ループとなります。
while (!feof($file)) {
}

fclose($file);
?>
```

---

fflush

(PHP 4 >= 4.0.1, PHP 5)

fflush — 出力をファイルにフラッシュする

説明

bool **fflush** (resource $handle)

この関数は、バッファリングされた全ての出力をファイルファイルハンドル *handle* が指すリソースに強制的に書き込みます。

パラメータ

handle

ファイルポインタは、有効なファイルポインタである必要があり、 fopen() または fsockopen() で正常にオープンされた（そしてまだ fclose() でクローズされていない）ファイルを指している必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

fgetc

(PHP 4, PHP 5)

fgetc — ファイルポインタから1文字取り出す

説明

string **fgetc** (resource $handle)

指定したファイルポインタから 1 文字読み出します。

パラメータ

handle

ファイルポインタは、有効なファイルポインタである必要があり、 fopen() または fsockopen() で正常にオープンされた（そしてまだ fclose() でクローズされていない）ファイルを指している必要があります。

返り値

handle が指すファイルポインタから 1 文字読み出し、 その文字からなる文字列を返します。EOF の場合に **FALSE** を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される *0* や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

例

Example#1 fgetc() の例

```
<?php
$fp = fopen('somefile.txt', 'r');
if (!$fp) {
    echo 'somefile.txt をオープンできませんでした';
}
while (false !== ($char = fgetc($fp))) {
    echo "$char¥n";
}
?>
```

注意

> **注意**: この関数はバイナリデータに対応しています。

参考

- fread()
- fopen()
- popen()
- fsockopen()
- fgets()

---

fgetcsv

(PHP 4, PHP 5)

fgetcsv — ファイルポインタから行を取得し、CSVフィールドを処理する

説明

array **fgetcsv** (resource `$handle` [, int `$length` [, string `$delimiter` [, string `$enclosure` [, string `$escape`]]]])

fgets() に動作は似ていますが、 **fgetcsv()** は行を CSV フォーマットのフィールドとして読込み処理を行い、 読み込んだフィールドを含む配列を返すという違いがあります。

パラメータ

handle

> ファイルポインタは有効なものでなければならず、また fopen(), popen(), もしくは fsockopen() で正常にオープンされたファイルを指している必要があります。

length

> (行末文字を考慮して) CSV ファイルにある最も長い行よりも大きい必要があります。 PHP 5 でオプションになりました。このパラメータを省略 (もしくは PHP 5.0.4 かそれ以降で 0 を設定) すると、 最大行長は制限されません。この場合、若干動作が遅くなります。

delimiter

> フィールドのデリミタ (1 文字のみ) を設定します。 デフォルトはカンマです。

enclosure

> フィールド囲いこみ文字 (1 文字のみ) を設定します。 デフォルトはダブルクォーテーションマークです。

escape

> エスケープ文字 (1 文字のみ) を設定します。 デフォルトはバックスラッシュ (¥) です。

返り値

読み込んだフィールドの内容を含む数値添字配列を返します。

> **注意**: CSV ファイルの空行は null フィールドを一つだけ含む配列として返され、 エラーにはなりません。

> **注意**: マッキントッシュコンピュータ上で作成されたファイルを読み込む際に、 *PHP* が行末を認識できないという問題が発生した場合、実行時の設定オプションauto_detect_line_endings を有効にする必要が生じるかもしれません。

fgetcsv() は、ファイルの終端に達した場合を含めてエラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.3.0	*escape* パラメータが追加されました。
4.3.5	**fgetcsv()** はバイナリセーフとなりました。
4.3.0	*enclosure* パラメータが追加されました。

例

Example#1 CSV ファイルの全てのコンテンツを読み込み、表示する

```
<?php
$row = 1;
$handle = fopen("test.csv", "r");
while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
    $num = count($data);
    echo "<p> $num fields in line $row: <br /></p>¥n";
    $row++;
    for ($c=0; $c < $num; $c++) {
        echo $data[$c] . "<br />¥n";
    }
}
fclose($handle);
?>
```

注意

> **注意**: この関数はロケール設定を考慮します。もし *LANG* が例えば *en_US.UTF-8* の場合、 ファイル中の 1 バイトエンコーディングは間違って読み込まれます。

参考

- str_getcsv()
- explode()
- file()
- pack()
- fputcsv()

---

fgets

(PHP 4, PHP 5)

fgets — ファイルポインタから 1 行取得する

説明

string **fgets** (resource `$handle` [, int `$length`])

ファイルポインタから 1 行取得します。

パラメータ

handle

> ファイルポインタは、有効なファイルポインタである必要があり、 fopen() または fsockopen() で正常にオープンされた（そしてまだ fclose() でクローズされていない）ファイルを指している必要があります。

length

> 読み出しは、*length* - 1 バイト読み出したか、(返り値に含まれる) 改行文字を検出したか、EOF に達したかのいずれかが起こった時点で終了します。 length が指定されない場合は、行末に達するまで読み続けます。
>
> > **注意**: PHP 4.3.0 より前のバージョンでは、もしこのパラメータが省略された場合、 行の長さを 1024 と仮定していました。 もしもファイル内の行の多くが 8KB を超えている場合、 行の長さの最大値を特定するためにスクリプトはリソースの影響を より大きく受けることになります。

返り値

handle で指定したファイルポインタから最大 *length* - 1 バイト読み出し、 その文字列を返します。

エラーが起こった場合、**FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	**fgets()** はバイナリセーフとなりました。
4.2.0	*length* パラメータがオプションとなりました。

例

Example#1 行毎にファイルを読み込む

```
<?php
$handle = @fopen("/tmp/inputfile.txt", "r");
if ($handle) {
    while (!feof($handle)) {
        $buffer = fgets($handle, 4096);
        echo $buffer;
    }
    fclose($handle);
}
?>
```

注意

> **注意**: マッキントッシュコンピュータ上で作成されたファイルを読み込む際に、 *PHP* が行末を認識できないという問題が発生した場合、実行時の設定オプションauto_detect_line_endings を有効にする必要が生じるかもしれません。
>
> **注意**: C 言語の **fgets()** の動作に慣れている人は、 *EOF* を返す条件の違いについて注意する必要があります。

参考

- fgetss()
- fread()
- fgetc()
- stream_get_line()
- fopen()
- popen()
- fsockopen()
- stream_set_timeout()

---

fgetss

(PHP 4, PHP 5)

fgetss — ファイルポインタから 1 行取り出し、HTML タグを取り除く

説明

string **fgetss** (resource `$handle` [, int `$length` [, string `$allowable_tags`]])

fgets() と同じですが、 **fgetss()** は読み込んだテキストから HTML および PHP のタグを取り除こうとすることが異なります。

パラメータ

handle

> ファイルポインタは、有効なファイルポインタである必要があり、 fopen() または fsockopen() で正常にオープンされた（そしてまだ fclose() でクローズされていない）ファイルを指している必要があります。

length

> 取得したいデータの長さ。

allowable_tags

> オプションの 3 番目の引数を使用して、 取り除く必要がないタグを指定することができます。

返り値

handle で指定したファイルポインタから最大 *length* - 1 バイト読み出し、 HTML や PHP コードを取り除いた文字列を返します。

エラーが発生した場合は **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	*length* パラメータがオプションとなりました。
3.0.13 および 4.0.0	*allowable_tags* パラメータが追加されました。

注意

> **注意**: マッキントッシュコンピュータ上で作成されたファイルを読み込む際に、 *PHP* が行末を認識できないという問題が発生した場合、実行時の設定オプションauto_detect_line_endings を有効にする必要が生じるかもしれません。

参考

- fgets()
- fopen()
- popen()
- fsockopen()
- strip_tags()

---

file_exists

(PHP 4, PHP 5)

file_exists — ファイルまたはディレクトリが存在するかどうか調べる

説明

bool **file_exists** (string `$filename`)

ファイルあるいはディレクトリが存在するかどうかを調べます。

パラメータ

filename

> ファイルあるいはディレクトリへのパス。
>
> Windows 上でネットワーク共有上のファイルを調べるには、 *//computername/share/filename* または *¥¥computername¥share¥filename* のように指定してください。

返り値

filename で指定したファイルまたはディレクトリが存在すれば **TRUE** を返し、そうでなければ **FALSE** を返します。

警告

この関数は セーフモード の制限のためファイルにアクセスできない場合 **FALSE** を返します。 しかし safe_mode_include_dir で指定されたディレクトリに存在する場合は include することができます。

> **注意**: チェックは、実効ユーザではなく実ユーザの UID/GID で行います。

例

Example#1 あるファイルが存在するかどうか調べる

```
<?php
$filename = '/path/to/foo.txt';

if (file_exists($filename)) {
    echo "$filename が存在します";
} else {
    echo "$filename は存在しません";
}
?>
```

注意

> **注意**: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- is_readable()
- is_writable()
- is_file()
- file()

---

file_get_contents

(PHP 4 >= 4.3.0, PHP 5)

file_get_contents — ファイルの内容を全て文字列に読み込む

説明

string **file_get_contents** (string `$filename` [, int `$flags` [, resource `$context` [, int `$offset` [, int `$maxlen`]]]])

この関数は file() と似ていますが、 *offset* で指定した場所から開始し *maxlen* バイト分だけ ファイルの内容を文字列に読み込むという点が異なります。 失敗した場合、**file_get_contents()** は **FALSE** を返します。

file_get_contents()はファイルの内容を文字列に読み込む 方法として好ましいものです。もしOSがサポートしていれば パフォーマンス向上のためにメモリマッピング技術が使用されます。

> **注意**: 空白のような特殊な文字を有する URI をオープンする場合には、 urlencode() でその URI をエンコードする必要があります。

パラメータ

filename

データを読み込みたいファイルの名前。

flags

警告

PHP 6 より前のバージョンでは、このパラメータは *use_include_path* という名前の bool パラメータでした。この *flags* パラメータは PHP 6 以降でのみ使用可能です。古いバージョンを使用しており、 *filename* を インクルードパス から探したい場合は、このパラメータには **TRUE** を設定することになるでしょう。 PHP 6 以降の場合は、同じ動作をさせたい場合は **FILE_USE_INCLUDE_PATH** フラグを指定します。

flags の値は、以下のフラグを組み合わせたものとなります (組み合わせ方には多少制限があります)。組み合わせる際には、論理 OR (|) 演算子で連結します。

使用できるフラグ

フラグ	説明
FILE_USE_INCLUDE_PATH	*filename* をインクルードディレクトリから探します。 詳細な情報は include_path を参照ください。
FILE_TEXT	unicode が有効な場合、データ読み込み時のデフォルトのエンコーディングは UTF-8 です。 別のエンコーディングを指定するには、 独自のコンテキストを作成するか、あるいは **stream_default_encoding()** でデフォルトを変更します。 このフラグは **FILE_BINARY** と同時に使用することはできません。
FILE_BINARY	このフラグを指定すると、ファイルをバイナリモードで読み込みます。これはデフォルトの設定で、 **FILE_TEXT** と同時に使用することはできません。

context

stream_context_create() で作成したコンテキストリソース。 独自のコンテキストを使用する必要がない場合は、このパラメータに **NULL** を指定します。

offset

読み込みを開始するオフセット位置。

maxlen

読み込むデータの最大バイト数。

返り値

読み込んだデータを返します。失敗した場合は **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	コンテキストサポートが追加されました。
5.1.0	*offset* と *maxlen* パラメータが追加されました。
6.0.0	*use_include_path* パラメータが *flags* パラメータに置き換えられました。

注意

注意: この関数はバイナリデータに対応しています。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

警告

IIS のような、いくつかの標準に対応してない Web サーバは、PHP に警告を発生させるような手順でデータを送信します。このようなサーバを使用する場合は、 error_reporting を警告を発生しないレベルまで小さくする必要があります。 PHP 4.3.7 以降では、https:// ラッパーでストリームをオープンする際にバグがある IIS サーバソフトウエアを検出することができ、この警告を抑制することができます。あなたが ssl:// ソケットを作性するために fsockopen() を使用している場合、自らこの警告を検出し、抑制する必要があります。

参考

- file()
- fgets()
- fread()
- readfile()
- file_put_contents()
- stream_get_contents()
- stream_context_create()

---

file_put_contents

(PHP 5)

file_put_contents — 文字列をファイルに書き込む

説明

int **file_put_contents** (string `$filename` , mixed `$data` [, int `$flags` [, resource `$context`]])

この関数は、fopen()、fwrite()、 fclose() を続けてコールしてデータをファイルに書き込むのと等価です。

filename が存在しない場合はファイルを作成します。 存在する場合はそのファイルを上書きします。ただし **FILE_APPEND** フラグが設定されている場合は別です。

パラメータ

filename

データを書き込むファイルへのパス。

data

書き込むデータ。文字列, 配列 もしくは ストリーム リソース (上述) のいずれかを指定可能です。

data が ストリーム リソースの場合は、 ストリームのバッファに残っている内容が指定したファイルにコピーされます。 これは、 stream_copy_to_stream() の挙動と似ています。

data に一次元の配列を指定することもできます。この場合は *file_put_contents($filename, implode('', $array))* と同じ意味になります。

flags

flags の値は、以下のフラグを組み合わせたものとなります (組み合わせ方には多少制限があります)。組み合わせる際には、論理 OR (|) 演算子

で連結します。

使用できるフラグ

フラグ	説明
FILE_USE_INCLUDE_PATH	*filename* をインクルードディレクトリから探します。 詳細な情報は include_path を参照ください。
FILE_APPEND	*filename* がすでに存在する場合に、 データをファイルに上書きするするのではなく追記します。
LOCK_EX	書き込み処理中に、ファイルに対する排他ロックを確保します。
FILE_TEXT	*data* をテキストモードで書き込みます。 unicode が有効な場合、デフォルトのエンコーディングは UTF-8 です。 別のエンコーディングを指定するには、 独自のコンテキストを作成するか、あるいは **stream_default_encoding()** でデフォルトを変更します。 このフラグは **FILE_BINARY** と同時に使用することはできません。 このフラグは PHP 6 以降でのみ使用可能です。
FILE_BINARY	*data* をバイナリモードで書き込みます。これはデフォルトの設定で、 **FILE_TEXT** と同時に使用することはできません。 このフラグは PHP 6 以降でのみ使用可能です。

context

stream_context_create() で作成したコンテキストリソース。

返り値

この関数はファイルに書き込まれたバイト数を返します。 あるいは失敗した場合には **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	コンテキストがサポートされるようになりました。
5.1.0	**LOCK_EX** のサポートが追加され、 *data* パラメータにストリームリソースを指定することが可能になりました。
6.0.0	**FILE_TEXT** および **FILE_BINARY** がサポートされるようになりました。

注意

注意: この関数はバイナリデータに対応しています。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

参考

- fopen()
- fwrite()
- file_get_contents()
- stream_context_create()

---

file

(PHP 4, PHP 5)

file — ファイル全体を読み込んで配列に格納する

説明

array **file** (string `$filename` [, int `$flags` [, resource `$context`]])

ファイル全体を配列に読み込みます。

注意: ファイルの内容を文字列として返すには file_get_contents() を使用します。

パラメータ

filename

ファイルへのパス。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen() 、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

flags

オプションのパラメータ *flags* は、以下の定数のうちのひとつ、あるいは複数の組み合わせとなります。

FILE_USE_INCLUDE_PATH
include_path のファイルを探します。

FILE_IGNORE_NEW_LINES
配列の各要素の最後に改行文字を追加しません。

FILE_SKIP_EMPTY_LINES
空行を読み飛ばします。

FILE_TEXT
コンテンツを UTF-8 エンコーディングで返します。 別のエンコーディングを指定するには、独自のコンテキストを作成します。 このフラグは **FILE_BINARY** と同時に使用することはできません。 このフラグは PHP 6 以降でのみ使用可能です。

FILE_BINARY
コンテンツをバイナリデータとして読み込みます。 これはデフォルトの設定で、 **FILE_TEXT** と同時に使用することはできません。 このフラグは PHP 6 以降でのみ使用可能です。

context

stream_context_create() 関数で作成したコンテキストリソース。

注意: コンテキストのサポートは、 PHP 5.0.0 で追加されました。*contexts* の説明に関しては、 ストリーム を参照してください。

返り値

ファイルを配列に入れて返します。 配列の各要素はファイルの各行に対応します。改行記号はついたままとなります。 失敗すると **file()** は **FALSE** を返します。

注意: **FILE_IGNORE_NEW_LINES** を指定しない限り、 配列に取り込まれた各行は行末文字も含みます。 行末文字を取り除きたい場合には rtrim() を使用する必要があります。

注意: マッキントッシュコンピュータ上で作成されたファイルを読み込む際に、 *PHP* が行末を認識できないという問題が発生した場合、実行時の設定オプションauto_detect_line_endings を有効にする必要が生じるかもしれません。

変更履歴

バージョン	説明
6.0.0	**FILE_TEXT** フラグおよび **FILE_BINARY** フラグをサポートするようになりました。
5.0.0	*context* パラメータが追加されました。
5.0.0	PHP 5.0.0 より前のバージョンでは、パラメータ *flags* でカバーしているのは include_path の設定だけでした。これを有効にするには 1 を指定します。
4.3.0	**file()** はバイナリセーフとなりました。

例

Example#1 file() の例

```
<?php
// ファイルの内容を配列に取り込みます。
// この例ではHTTPを通してURL上のHTMLソースを取得します。
$lines = file('http://www.example.com/');

// 配列をループしてHTMLをHTMLソースとして表示し、行番号もつけます。
foreach ($lines as $line_num => $line) {
    echo "Line #<b>{$line_num}</b> : " . htmlspecialchars($line) . "<br />¥n";
}

// 他の例として、Webページを文字列に取り込みます。file_get_contents()も参照してください。
$html = implode('', file('http://www.example.com/'));
?>
```

注意

警告

IIS のような、いくつかの標準に対応してない Web サーバは、PHP に警告を発生させるような手順でデータを送信します。このようなサーバを使用す

る場合は、 error_reporting を警告を発生しないレベルまで小さくする必要があります。 PHP 4.3.7 以降では、https:// ラッパーでストリームをオープンする際にバグがある IIS サーバソフトウエアを検出することができ、この警告を抑制することができます。あなたが ssl:// ソケットを作性するために fsockopen() を使用している場合、自らこの警告を検出し、抑制する必要があります。

参考

- readfile()
- fopen()
- fsockopen()
- popen()
- file_get_contents()
- include()
- stream_context_create()

---

fileatime

(PHP 4, PHP 5)

fileatime — ファイルの最終アクセス時刻を取得する

説明

int **fileatime** (string `$filename`)

指定したファイルの最終アクセス時刻を取得します。

パラメータ

`filename`

ファイルへのパス。

返り値

ファイルの最終アクセス時刻を返し、エラーの場合は **FALSE** を返します。 時間は Unix タイムスタンプとして返されます。

例

Example#1 fileatime() の例

```
<?php

// 出力例 somefile.txt was last accessed: December 29 2002 22:16:23.

$filename = 'somefile.txt';
if (file_exists($filename)) {
    echo "$filename was last accessed: " . date("F d Y H:i:s.", fileatime($filename));
}

?>
```

注意

注意: ファイルの atime は、 ファイルのデータブロックが読み込まれる度に変更されるとみなされます。 この仕様は、アプリケーションが非常に多くのファイルまたはディレクトリに 常にアクセスする場合に性能上の負荷となる可能性があります。
Unix のファイルシステムの中には、 このようなアプリケーションの性能を向上させるために atime の更新を無効としてマウントできるものもあります。 USENET のニュースプールが一般的な例です。 このようなファイルシステムでは、この関数は使用できません。

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- filemtime()
- fileinode()
- date()

---

filectime

(PHP 4, PHP 5)

filectime — ファイルの inode 変更時刻を取得する

説明

int **filectime** (string `$filename`)

ファイルの inode 変更時刻を取得します。

パラメータ

`filename`

ファイルへのパス。

返り値

ファイルの最終更新時刻を返し、エラーの場合は **FALSE** を返します。 時間は Unix タイムスタンプとして返されます。

例

Example#1 filectime() の例

```
<?php

// 出力例 somefile.txt was last changed: December 29 2002 22:16:23.

$filename = 'somefile.txt';
if (file_exists($filename)) {
    echo "$filename was last changed: " . date("F d Y H:i:s.", filectime($filename));
}

?>
```

注意

注意: 注意: 多くの Unix ファイルシステムでは、i-ノードが変更された際、つまり、 パーミッション、所有者、グループ、または他のメタデータが書き込まれた際に、 ファイルが変更されたとみなされます。 filemtime() (この関数は、Web ページ上に "最終更新時刻" を表示させたい場合に使用するものです) および fileatime() も参照ください。

注意: いくつかの Unix では、ファイルの ctime はファイルの作成時間として参照されます。 これは間違っています。多くの Unix ファイルシステムでは Unix ファイルの作成時間は存在しません。

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- filemtime()

---

filegroup

(PHP 4, PHP 5)

filegroup — ファイルのグループを取得する

説明

int **filegroup** (string `$filename`)

ファイルのグループを取得します。返り値は、数値形式のグループ ID です。 グループ名を取得するには posix_getgrgid() を使用します。

パラメータ

filename

ファイルへのパス。

返り値

ファイルの所有者のグループ ID を返し、エラーの場合は **FALSE** を返します。 グループ ID は数値で返されます。グループ名に変換するには posix_getgrgid() を使用してください。 失敗すると **FALSE** を返します。

エラー / 例外

失敗時には **E_WARNING** レベルのエラーを発行します。

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- fileowner()
- safe_mode_gid

---

fileinode

(PHP 4, PHP 5)

fileinode — ファイルの inode を取得する

説明

int **fileinode** (string `$filename`)

ファイルの inode を取得します。

パラメータ

filename

ファイルへのパス。

返り値

ファイルの inode 番号を返し、エラーの場合は **FALSE** を返します。

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- stat()

---

filemtime

(PHP 4, PHP 5)

filemtime — ファイルの更新時刻を取得する

説明

int **filemtime** (string `$filename`)

この関数は、ファイルのブロックデータが書き込まれた時間を返します。 これは、ファイルの内容が変更された際の時間です。

パラメータ

`filename`

ファイルへのパス。

返り値

ファイルの最終更新時刻を返し、エラーの場合は **FALSE** を返します。時間は Unix タイムスタンプとして返されます。 この関数の結果は date() 等で使用できます。

例

Example#1 filemtime() の例

```
<?php
// 出力例 somefile.txt was last modified: December 29 2002 22:16:23.

$filename = 'somefile.txt';
if (file_exists($filename)) {
    echo "$filename was last modified: " . date ("F d Y H:i:s.", filemtime($filename));
}
?>
```

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- filectime()
- stat()
- touch()
- getlastmod()

---

fileowner

(PHP 4, PHP 5)

fileowner — ファイルの所有者を取得する

説明

int **fileowner** (string `$filename`)

ファイルの所有者を取得します。

パラメータ

`filename`

ファイルへのパス。

返り値

ファイルの所有者のユーザ ID を返し、エラーの場合は **FALSE** を返します。 ユーザ ID は数値で返されます。ユーザ名に変換するには posix_getpwuid() を使用してください。

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- stat()

---

fileperms

(PHP 4, PHP 5)

fileperms — ファイルのパーミッションを取得する

説明

int **fileperms** (string `$filename`)

指定したファイルのパーミッションを取得します。

パラメータ

`filename`

ファイルへのパス。

返り値

ファイルのパーミッション、あるいはエラー時に **FALSE** を返します。

例

Example#1 八進形式でのパーミッションの表示

```
<?php
echo substr(sprintf('%o', fileperms('/tmp')), -4);
echo substr(sprintf('%o', fileperms('/etc/passwd')), -4);
?>
```

上の例の出力は以下となります。

```
1777
0644
```

Example#2 完全なパーミッションの表示

```
<?php
$perms = fileperms('/etc/passwd');

if (($perms & 0xC000) == 0xC000) {
    // ソケット
    $info = 's';
} elseif (($perms & 0xA000) == 0xA000) {
    // シンボリックリンク
    $info = 'l';
} elseif (($perms & 0x8000) == 0x8000) {
    // 通常のファイル
    $info = '-';
} elseif (($perms & 0x6000) == 0x6000) {
    // ブロックスペシャルファイル
    $info = 'b';
} elseif (($perms & 0x4000) == 0x4000) {
    // ディレクトリ
    $info = 'd';
} elseif (($perms & 0x2000) == 0x2000) {
    // キャラクタスペシャルファイル
    $info = 'c';
} elseif (($perms & 0x1000) == 0x1000) {
    // FIFO パイプ
    $info = 'p';
} else {
    // 不明
```

```
    $info = 'u';
}

// 所有者
$info .= (($perms & 0x0100) ? 'r' : '-');
$info .= (($perms & 0x0080) ? 'w' : '-');
$info .= (($perms & 0x0040) ?
            (($perms & 0x0800) ? 's' : 'x' ) :
            (($perms & 0x0800) ? 'S' : '-'));

// グループ
$info .= (($perms & 0x0020) ? 'r' : '-');
$info .= (($perms & 0x0010) ? 'w' : '-');
$info .= (($perms & 0x0008) ?
            (($perms & 0x0400) ? 's' : 'x' ) :
            (($perms & 0x0400) ? 'S' : '-'));

// 全体
$info .= (($perms & 0x0004) ? 'r' : '-');
$info .= (($perms & 0x0002) ? 'w' : '-');
$info .= (($perms & 0x0001) ?
            (($perms & 0x0200) ? 't' : 'x' ) :
            (($perms & 0x0200) ? 'T' : '-'));

echo $info;
?>
```

上の例の出力は以下となります。

```
-rw-r--r--
```

注意

> **注意**: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- is_readable()
- stat()

---

filesize

(PHP 4, PHP 5)

filesize — ファイルのサイズを取得する

説明

int **filesize** (string `$filename`)

指定したファイルのサイズを取得します。

パラメータ

`filename`

> ファイルへのパス。

返り値

ファイルのサイズを返し、エラーの場合は **`FALSE`** を返します (また **`E_WARNING`** レベルのエラーを発生させます) 。

> **注意**: PHP の数値型は符号付整数であり、 多くのプラットフォームでは 32 ビットの整数を取るため、 **filesize()** は 2GB より大きなファイルについては期待とは違う値を返すことがあります。 2GB から 4GB のサイズのファイルについては *sprintf("%u", filesize($file))* を使うことで打開されます。

例

Example#1 filesize() の例

```
<?php
```

```
// 出力例 somefile.txt: 1024 bytes

$filename = 'somefile.txt';
echo $filename . ': ' . filesize($filename) . ' bytes';

?>
```

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- file_exists()

---

filetype

(PHP 4, PHP 5)

filetype — ファイルタイプを取得する

説明

string **filetype** (string `$filename`)

指定したファイルのタイプを返します。

パラメータ

`filename`

ファイルへのパス。

返り値

ファイルのタイプを返します。返される値は fifo、char、dir、 block、link、file、socket および unknown のいずれかです。

エラーが発生すると **FALSE** を返します。 また **filetype()** は stat コールに失敗したり、 未知のファイルタイプであったりした場合に **E_NOTICE** メッセージを発行します。

例

Example#1 filetype() の例

```
<?php

echo filetype('/etc/passwd');  // ファイル
echo filetype('/etc/');        // ディレクトリ

?>
```

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- is_dir()
- is_file()
- is_link()
- file_exists()
- stat()

- mime_content_type()

---

flock

(PHP 4, PHP 5)

flock — 汎用のファイルロックを行う

説明

bool **flock** (resource `$handle` , int `$operation` [, int `&$wouldblock`])

flock()により、(ほとんどの Unix や Windows さえ含む) ほとんど全てのプラットフォームで使用可能な簡易な読み手/書き手モデルが実現されます。

ロックの解放には fclose() でを使用します (これは、スクリプトが終了した場合にも自動的にコールされます)。

PHP は、恣意的にファイルをロックする汎用の手段を提供します (これは、アクセスする全プログラムが同一のロックの方法を使用する必要があり、 そうでない場合は動作しないことを意味します)。

パラメータ

`handle`

オープンしたファイルへのポインタ。

`operation`

`operation` は以下のいずれかとなります。

- 共有ロック(読み手)とするには **LOCK_SH** をセットします。
- 排他的ロック(書き手)とするには **LOCK_EX** をセットします。
- (共有または排他的)ロックを開放するには **LOCK_UN** をセットします。
- ロック中に **flock()** でブロックを行いたくない場合は、 **LOCK_NB** をセットします (Windows ではサポートされていません)。

`wouldblock`

ロックをブロックモードとする場合 (EWOULDBLOCK errno 条件) にオプションの 3 番目の引数に **TRUE** を設定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.0.1	定数 *LOCK_XXX* が追加されました。以前のバージョンでは **LOCK_SH** のかわりに 1、 **LOCK_EX** のかわりに 2、 **LOCK_UN** のかわりに 3、そして **LOCK_NB** のかわりに 4 を使用しなければなりません。

例

Example#1 flock() の例

```
<?php

$fp = fopen("/tmp/lock.txt", "w+");

if (flock($fp, LOCK_EX)) { //  排他ロックを行います
    fwrite($fp, "Write  something  here¥n");
    flock($fp, LOCK_UN); //  ロックを解放します
} else {
    echo "ファイルをロックできません!";
}

fclose($fp);

?>
```

注意

注意: **flock()** は、Windows 環境下では必ずロックを行います。

注意: **flock()**は、ファイルポインタを必要とするため、 (fopen()へ引数"w"または"w+"を指定して)書き込 みモードでオープンすることによ

り丸めるファイルにアクセス保護する 特別なロックファイルを使用する必要があるかもしれません。

警告

flock() は NFS 及び他の多くのネットワークファイルシステムでは動作しません。 詳細についてはオペレーティングシステムのドキュメントを確認ください。

いくつかのオーペレーティングシステムで**flock()** はプロセスレベルで実装されています。ISAPIのようなマルチスレッド 型のサーバーAPIを使用している場合、同じサーバーインスタンスの並 列スレッドで実行されている他のPHPスクリプトに対してファイルを保 護する際に **flock()**を使用することはできません!

flock()は*FAT*のような 旧式のファイルシステムではサポートされていないため、 そのような環境の場合は常に**FALSE**を返すことになります。 （これは特にWindows98ユーザーにとって常に真です）

---

fnmatch

(PHP 4 >= 4.3.0, PHP 5)

fnmatch — ファイル名がパターンにマッチするか調べる

説明

bool **fnmatch** (string `$pattern` , string `$string` [, int `$flags`])

fnmatch()は*string* で指定された文字列が *pattern* で指定されたシェルワイルドカードにマッチするかどうかチェックします。

パラメータ

pattern

シェルのワイルドカードパターン。

string

調べたい文字列。この機能は特にファイル名のマッチに便利ですが、 通常の文字列に関しても使用できます。

一般的なユーザにとって、シェルパターンやあるいは少なくとも '?'と '*'によるワイルドカードのほうが 慣れていると思われます。そのため、ereg() または preg_match() などの代わりに **fnmatch()** をフロントエンドの検索表現として使うことは、 プログラマではないユーザにとってより便利でしょう。

flags

フラグ名については Unix のマニュアル *fnmatch(3)* も参照ください (そのうちここにも説明が追加されるでしょう)。

返り値

マッチした場合に **TRUE**、それ以外の場合に **FALSE** を返します。

例

Example#1 シェルのワイルドカードパターンによる色の名前のチェック

```
<?php
if (fnmatch("*gr[ae]y", $color)) {
   echo "some form of gray ...";
}
?>
```

注意

警告

今のところ、この機能は Windows あるいは他の POSIX に準拠していないシステムで利用できません。

参考

- glob()
- ereg()
- preg_match()
- sscanf()
- printf()

- sprintf()

---

fopen

(PHP 4, PHP 5)

fopen — ファイルまたは URL をオープンする

説明

resource **fopen** (string $filename , string $mode [, bool $use_include_path [, resource $context]])

fopen() は、 *filename* で指定されたリソースをストリームに結び付けます。

パラメータ

filename

filename が "スキーム://..." の形式である場合、 それは URL とみなされ、PHP はそのプロトコルのハンドラ (ラッパーともいいます) を探します。 もしもそのプロトコルに対するラッパが登録されていない場合、 PHP はスクリプトに潜在的な問題があることを示す NOTICE を発行したうえで、 *filename* を通常のファイルとみなしてオープンすることを試みます。

PHP は、 *filename* がローカルのファイルを示しているとみなすと、 そのファイルへのストリームをオープンします。 そのファイルはPHPからアクセスできるものでなければなりません。 ファイルのパーミッションが (パラメータで指定された) アクセスを許可されているかどうか確認する必要があります。 セーフモード または open_basedir を有効にしている場合は更なるアクセス制限が加えられることがあります。

filename が登録されているプロトコルを示していると PHP が判断し、かつそのプロトコルがネットワーク URL として登録されていれば、PHP は allow_url_fopen が有効となっているかどうかチェックします。 もしこれがオフになっていると、PHP は warning を発行し fopen は失敗します。

> **注意**: サポートされているプロトコルのリストは サポートされるプロトコル/ラッパー にあります。 いくつかのプロトコル (*wrappers*にも関連する) は *context* かつ/または *php.ini* のオプションをサポートします。 使用するプロトコルについてセットされるオプションのリストについては、 それぞれのページを見てください (例えば、 *php.ini* 上の *user_agent* の値は *http* ラッパーが使用します)。

Windows 環境では、ファイルパスで用いる全てのバックスラッシュを エスケープするかフォワードスラッシュを使用することに注意してください。

```
<?php
$handle = fopen("c:¥¥data¥¥info.txt", "r");
?>
```

mode

パラメータ *mode* は、 そのストリームに要するアクセス形式を指定します。 この指定は、下表のうちのどれかとなります。

fopen() で使用可能な *mode* のリスト

mode	説明
'r'	読み込みのみでオープンします。ファイルポインタをファイルの先頭に置きます。
'r+'	読み込み／書き出し用にオープンします。 ファイルポインタをファイルの先頭に置きます。
'w'	書き出しのみでオープンします。ファイルポインタをファイルの先頭に置き、 ファイルサイズをゼロにします。ファイルが存在しない場合には、 作成を試みます。
'w+'	読み込み／書き出し用でオープンします。 ファイルポインタをファイルの先頭に置き、 ファイルサイズをゼロにします。 ファイルが存在しない場合には、作成を試みます。
'a'	書き出し用のみでオープンします。ファイルポインタをファイルの終端に置きます。 ファイルが存在しない場合には、作成を試みます。
'a+'	読み込み／書き出し用でオープンします。 ファイルポインタをファイルの終端に置きます。 ファイルが存在しない場合には、作成を試みます。
'x'	書き込みのみでオープンします。ファイルポインタをファイルの先頭に置きます。 ファイルが既に存在する場合には **fopen()** は失敗し、**E_WARNING** レベルのエラーを発行します。 ファイルが存在しない場合には新規作成を試みます。 これは *open(2)* システムコールにおける *O_EXCL¦O_CREAT* フラグの指定と等価です。 このオプションはPHP4.3.2以降でサポートされ、また、 ローカルファイルに対してのみ有効です。
'x+'	読み込み／書き出し用でオープンします。ファイルポインタをファイルの先頭に置きます。 ファイルが既に存在する場合には **fopen()** は失敗し、 **E_WARNING** レベルのエラーを発行します。 これは *open(2)* システムコールにおける *O_EXCL¦O_CREAT* フラグの指定と等価です。 このオプションは PHP 4.3.2 以降でサポートされ、また、 ローカルファイルに対してのみ有効です。

> **注意**: オペレーティングシステムファミリが異なると行末も異なります。 テキストファイルに書き出し、そこに改行を加えたいと

き、 オペレーティングシステムにあわせた正しい改行コードを使用する必要があります。 Unix ベースのシステムでは改行に *¥n* キャラクタを使用します。 Windows ベースのシステムでは *¥r¥n* を使用します。 マッキントッシュベースのシステムでは *¥r* を使用します。
間違った改行コードでファイルに書き込むと、 他のアプリケーション上でそのファイルを開いた際に変な風に見えてしまいます。
Windows上では、*¥n*を*¥r¥n*に透過的に変換する text-mode変換フラグ(*'t'*)が提供されます。 それに対し、*'b'*を使って強制的にバイナリモードにすることもできます。 その場合データの変換はされません。 このフラグを使用するには、*'b'* または *'t'*を *mode* 引数の最後に追加してください。
デフォルトの変換モードは SAPI と使用している PHP のバージョンによって異なります。 したがって、互換性の意味から、常に適切なフラグを指定することが推奨されます。 plain-text ファイルを使用する場合には *'t'* モードを指定すべきであり、 改行に *¥n* を使用すると、 メモ帳のようなアプリケーションで読めることを期待できます。 それ以外のケースでは *'b'* を使うべきです。
バイナリファイルを扱っている際に *'b'* フラグを指定しなかった場合、 画像ファイルが壊れたり、*¥r¥n* キャラクタがおかしくなる等の問題を抱えてしまうでしょう。

注意: 互換性維持のために、**fopen()** でファイルをオープンする際は 常に *'b'* フラグを指定することが強く推奨されます。

注意: 互換性維持のために、*'t'* モードを使用または依存しているコードを書き直し、 正しい改行コードと *'b'* モードを代わりに使用することが、 強く推奨されます。

use_include_path

オプションの３番目の引数*use_include_path* に'1'又は **TRUE** を設定することにより、include_path のファイルの検索も行うこともできます。

context

注意: コンテキストのサポートは、 PHP 5.0.0 で追加されました。*contexts* の説明に関しては、 ストリーム を参照してください。

返り値

成功した場合にファイルポインタリソース、エラー時に **FALSE** を返します。

エラー / 例外

オープンが失敗するとこの関数は **FALSE** を返し、 **E_WARNING** レベルのエラーを発行します。 @ を使ってこの warning を抑制することもできます。

変更履歴

バージョン	説明
4.3.2	PHP 4.3.2 以降では、バイナリモードとテキストモードを区別する全ての プラットフォームにおいて、デフォルトのモードはバイナリにセットされます。 アップグレード後にスクリプトに問題が起きた場合は、 以上に述べたスクリプトの互換性を確保するまでの次善策として、 *'t'* フラグを試してみてください。
4.3.2	*'x'* および *'x+'* が追加されました。

例

Example#1 fopen() の例

```
<?php
$handle  = fopen("/home/rasmus/file.txt", "r");
$handle  = fopen("/home/rasmus/file.gif", "wb");
$handle  = fopen("http://www.example.com/", "r");
$handle  = fopen("ftp://user:password@example.com/somefile.txt", "w");
?>
```

注意

警告

IIS のような、いくつかの標準に対応してない Web サーバは、PHP に警告を発生させるような手順でデータを送信します。このようなサーバを使用する場合は、 error_reporting を警告を発生しないレベルまで小さくする必要があります。 PHP 4.3.7 以降では、https:// ラッパーでストリームをオープンする際にバグがある IIS サーバソフトウエアを検出することができ、この警告を抑制することができます。あなたが ssl:// ソケットを作性するために fsockopen() を使用している場合、自らこの警告を検出し、抑制する必要があります。

注意: セーフモード が有効の場合、PHP は、操作を行うディレクトリが、実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

ファイルの読みこみ・書きこみ時に問題が発生し、 サーバーモジュール版のPHPを使用している場合、 使用するファイル・ディレクトリがサーバプロセスからアクセス可能かどうかを確認してください。

参考

---

fpassthru

(PHP 4, PHP 5)

fpassthru — ファイルポインタ上に残っているすべてのデータを出力する

説明

int **fpassthru** (resource `$handle`)

与えられたファイルポインタを EOF まで読み、結果を出力バッファに書き出します。

ファイルに既にデータを書き終えている場合で ファイルポインタをファイルの先頭にリセットするには rewind() をコールする必要があります。

ファイルを更新したり特定のオフセットを探すのではなく 内容を出力バッファにダンプしたいだけの場合、 readfile() を使用することが可能です。 この場合、fopen() コールは必要ありません。

パラメータ

handle

ファイルポインタは、有効なファイルポインタである必要があり、 fopen() または fsockopen() で正常にオープンされた（そしてまだ fclose() でクローズされていない）ファイルを指している必要があります。

返り値

エラーが起こった場合、**fpassthru()** は **FALSE** を返します。 それ以外の場合、**fpassthru()** は *handle* から読み込んだ文字の数を返し、出力へ渡します。

例

Example#1 バイナリファイルに対する fpassthru() の使用例

```php
<?php

// バイナリモードでファイルをオープンする
$name = './img/ok.png';
$fp = fopen($name, 'rb');

// 正しいヘッダを送出する
header("Content-Type: image/png");
header("Content-Length: " . filesize($name));

// 画像をダンプしスクリプトを終了する
fpassthru($fp);
exit;

?>
```

注意

注意: **fpassthru()** を Windows システムのバイナリファイルで 使用する場合、fopen() をコールする際に モードに *b* を追加してバイナリモードでファイルを オープンするようにしてください。
バイナリファイルを扱う場合は、必要でなくても *b* フラグを使用するようにしましょう。 それにより、スクリプトの可搬性がより高くなります。

参考

- readfile()

- fopen()
- popen()
- fsockopen()

---

fputcsv

(PHP 5 >= 5.1.0)

fputcsv — 行を CSV 形式にフォーマットし、ファイルポインタに書き込む

説明

int **fputcsv** (resource $handle , array $fields [, string $delimiter [, string $enclosure]])

fputcsv() は、行（*fields* 配列として渡されたもの）を CSV としてフォーマットし、それを *handle* で指定したファイルに書き込みます (いちばん最後に改行を追加します)。

パラメータ

handle

ファイルポインタは、有効なファイルポインタである必要があり、 fopen() または fsockopen() で正常にオープンされた（そしてまだ fclose() でクローズされていない）ファイルを指している必要があります。

fields

値の配列。

delimiter

オプションの *delimiter* はフィールド区切り文字 (一文字だけ) を指定します。デフォルトはカンマ (,) です。

enclosure

オプションの *enclosure* はフィールドを囲む文字 (一文字だけ) を指定します。デフォルトは二重引用符 (") です。

返り値

書き込んだ文字列の長さを返します。失敗した場合は **FALSE** を返します。

例

Example#1 fputcsv() の例

```
<?php

$list = array (
    'aaa,bbb,ccc,dddd',
    '123,456,789',
    '"aaa","bbb"'
);

$fp = fopen('file.csv', 'w');

foreach ($list as $line) {
    fputcsv($fp, split(',', $line));
}

fclose($fp);
?>
```

注意

注意: マッキントッシュコンピュータ上で作成されたファイルを読み込む際に、 *PHP* が行末を認識できないという問題が発生した場合、実行時の設定オプションauto_detect_line_endings を有効にする必要が生じるかもしれません。

参考

- fgetcsv()

---

fputs

(PHP 4, PHP 5)

fputs — fwrite() のエイリアス

説明

この関数は次の関数のエイリアスです。 fwrite().

---

fread

(PHP 4, PHP 5)

fread — バイナリセーフなファイルの読み込み

説明

string **fread** (resource $handle , int $length)

fread() は、*handle* が指すファイルポインタから最高 *length* バイト読み込みます。 以下のいずれかの条件を満たしたら、読み込みを終了します。

- *length* バイトぶん読み込んだ
- EOF (ファイルの終端) に達した
- パケットが利用可能になった (ネットワークストリームの場合)
- 8192 バイトぶん読み込んだ (ユーザ定義ストリームをオープンした後)

パラメータ

handle

fopen() を使用して作成したファイルシステムポインタリソース。

length

最大 *length* バイトまで読み込む。

返り値

読み込んだ文字列、またはエラー時には **FALSE** を返します。

例

Example#1 シンプルな fread() の例

```
<?php
// ファイルの中身を読んで文字列に格納する
$filename = "/usr/local/something.txt";
$handle = fopen($filename, "r");
$contents = fread($handle, filesize($filename));
fclose($handle);
?>
```

Example#2 バイナリの fread() の例

警告

バイナリとテキストファイルの形式が異なるシステム(すなわち Windows)では、fopen()の mode パラメータに 'b' を指定してファイルをオープンする必要があります。

```
<?php
$filename = "c:¥¥files¥¥somepic.gif";
$handle = fopen($filename, "rb");
$contents = fread($handle, filesize($filename));
fclose($handle);
?>
```

Example#3 リモートファイルの fread() の例

警告

通常のローカルファイル以外のもの、例えば リモートファイル や popen()、fsockopen() が返す ストリームを読み込んでいる場合には、 パケットが有効になった後に読み込みはストップします。 つまり以下の例のように分割されたデータを結合すべきであるということです。

```
<?php
// PHP 5 以降での例
$handle = fopen("http://www.example.com/", "rb");
```

```
$contents = stream_get_contents($handle);
fclose($handle);
?>
<?php
$handle = fopen("http://www.example.com/", "rb");
$contents = '';
while (!feof($handle)) {
  $contents .= fread($handle, 8192);
}
fclose($handle);
?>
```

注意

注意: 文字列にファイルを読み込みたいだけならば、file_get_contents() を使うほうが上記の例よりも効率的です。

参考

- fwrite()
- fopen()
- fsockopen()
- popen()
- fgets()
- fgetss()
- fscanf()
- file()
- fpassthru()

---

fscanf

(PHP 4 >= 4.0.1, PHP 5)

fscanf — フォーマットに基づきファイルからの入力を処理する

説明

mixed **fscanf** (resource `$handle` , string `$format` [, mixed `&$...`])

関数**fscanf()** は sscanf() に似ていますが、*handle* が指すファイルから入力を取得し、 指定したフォーマット *format* に基づき解釈を行います。 フォーマットについては sprintf() に解説されています。

フォーマット文字列におけるあらゆる空白は 入力ストリームのあらゆる空白にマッチします。 これはつまりフォーマット文字列の¥t（タブ）すらも 入力ストリームの空白1個にマッチしてしまうことを意味します。

パラメータ

handle

fopen() を使用して作成したファイルシステムポインタリソース。

format

sprintf() のドキュメントに説明されているフォーマット。

...

オプションで代入する値。

返り値

この関数のパラメータが二つだけの場合、処理された値は配列として返されます。 他方、オプションのパラメータが指定された場合、 この関数は、代入された値の数を返します。 オプション引数は参照渡しとする必要があります。

変更履歴

バージョン	説明
4.3.0	このバージョンまでは、ファイルから読み込む文字の最大数は512 （または最初に\nが現れるまで）でした。 これ以降では任意の長い行を読みスキャンすることができます。

例

Example#1 fscanf() の例

```
<?php
$handle  = fopen("users.txt", "r");
while  ($userinfo  = fscanf($handle, "%s¥t%s¥t%s¥n")) {
     list ($name, $profession, $countrycode) = $userinfo;
    //... これらの値を使用して何か行う
}
fclose($handle);
?>
```

Example#2 users.txt の内容

```
javier   argonaut          pe
hiroshi  sculptor          jp
robert    slacker  us
luigi     florist  it
```

参考

- fread()
- fgets()
- fgetss()
- sscanf()
- printf()
- sprintf()

---

fseek

(PHP 4, PHP 5)

fseek — ファイルポインタを移動する

説明

int **fseek** (resource $handle , int $offset [, int $whence])

handle が指しているファイルのファイル位置識別子を ファイル・ストリーム中の *offset* バイト目に セットします。新規位置は、ファイルの先頭からのバイト数で 測られます。これは *whence* で指定した位置に *offset* を追加することにより得られます。

パラメータ

handle

fopen() を使用して作成したファイルシステムポインタリソース。

offset

オフセット。

ファイルの終端から数えた位置に移動するには、負の値を *offset* に渡す必要があります。

whence

whence の値は以下のようになります。

- **SEEK_SET** - 位置を *offset* バイト目に設定する
- **SEEK_CUR** - 現在の位置に *offset* を加えた位置に設定する
- **SEEK_END** - ファイル終端に *offset* を加えた位置に設定する

whence を指定しない場合、 **SEEK_SET** が指定されたと仮定します。

返り値

成功すると 0 を返し、そうでなければ -1 を返します。EOF より先の位置にシークしてもエラーとはならないので注意してください。

例

Example#1 fseek() の例

```
<?php

$fp = fopen('somefile.txt', 'r');

// データを読み込む
```

```php
$data = fgets($fp, 4096);

// ファイルの先頭に移動する。
// rewind($fp); と等価。
fseek($fp, 0);

?>
```

注意

注意: 追加モード ("a" or "a+") でファイルをオープンした場合、 ファイル位置によらず、ファイルに書き込むあらゆるデータが追加されます。

注意: "http://" または "ftp://" フォーマット指定の **fopen ()** により返されたファイルポインタに対しては使わないでください。 **fseek()** は、("a" フラグ付きでオープンされた) 追加のみ可能なストリームに対する結果も未定義です。

参考

- ftell()
- rewind()

---

fstat

(PHP 4, PHP 5)

fstat — オープンしたファイルポインタからファイルに関する情報を取得する

説明

array **fstat** (resource `$handle`)

ファイルポインタ*handle* によりオープンされたファイルの統計情報を取得します。 この関数は関数stat()に似ていますが、 ファイル名の代わりにオープンされたファイルポインタを指定するところが異なります。

パラメータ

handle

fopen() を使用して作成したファイルシステムポインタリソース。

返り値

ファイルの統計情報の配列を返します。配列のフォーマットについては stat()のマニュアルをご覧ください。

例

Example#1 fstat() の例

```php
<?php

// ファイルをオープンする
$fp = fopen("/etc/passwd", "r");

// 統計情報を収集する
$fstat = fstat($fp);

// ファイルをクローズする
fclose($fp);

// 連想配列部のみ表示する
print_r(array_slice($fstat, 13));

?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [dev] => 771
     [ino] => 488704
     [mode] => 33188
     [nlink] => 1
     [uid] => 0
     [gid] => 0
     [rdev] => 0
     [size] => 1114
     [atime] => 1061067181
     [mtime] => 1056136526
     [ctime] => 1056136526
     [blksize] => 4096
```

```
    [blocks] => 8
)
```

注意

> **注意**: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

---

ftell

(PHP 4, PHP 5)

ftell — ファイルポインタから読み書きの位置を取得する

説明

int **ftell** (resource `$handle`)

ファイルポインタから読み書きの位置を取得します。

パラメータ

`handle`

> ファイルポインタは有効なものでなければならず、また fopen()、popen() で正常にオープンされたファイルを指している必要があります。**ftell()** は、("a" フラグ付きでオープンされた) 追加のみ可能なストリームに対する結果も未定義です。

返り値

`handle` が示すファイルポインタの位置、 すなわちファイル・ストリーム上のオフセットを返します。

エラーが起こった場合 **FALSE** を返します。

例

Example#1 ftell() の例

```php
<?php

// ファイルをオープンし、データを読み込む
$fp = fopen("/etc/passwd", "r");
$data = fgets($fp, 12);

// どこにいるんだ ?
echo ftell($fp); // 11

fclose($fp);

?>
```

参考

- fopen()
- popen()
- fseek()
- rewind()

---

ftruncate

(PHP 4, PHP 5)

ftruncate — ファイルを指定した長さに丸める

説明

bool **ftruncate** (resource `$handle` , int `$size`)

ファイルポインタ*`handle`* を引数とし、 ファイルを指定した長さ、サイズに丸めます。

パラメータ

handle

ファイルポインタ。

注意: *handle* は書き込みモードでオープンする必要があります。

size

丸める大きさ。

注意: *size* がファイルのサイズより大きい場合は、 null バイトを用いてファイルを拡大します。
size がファイルのサイズより小さい場合は、 余分なデータは失われます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
PHP 4.3.3	これより前のバージョンでは、**ftruncate()** が成功した場合の値は boolean **TRUE** ではなく integer 型の 1 でした。

注意

注意: ファイルポインタは変更 *されません*。

参考

- fopen()
- fseek()

---

fwrite

(PHP 4, PHP 5)

fwrite — バイナリセーフなファイル書き込み処理

説明

int **fwrite** (resource `$handle` , string `$string` [, int `$length`])

fwrite()は*string* の内容を *handle* が指しているファイル・ストリームに書き込みます。

パラメータ

handle

fopen() を使用して作成したファイルシステムポインタリソース。

string

書き込む文字列。

length

length パラメータが与えられている場合、 *length* バイト数分の書き込みが完了したか、 *string* が終わりに達したかのいずれか早い方の 事象により書き込みは中止されます。

length パラメータが指定されている場合、 magic_quotes_runtime 構成オプションは無視され、*string* からの スラッシュ文字の取り除きは行われないことに注意してください。

返り値

fwrite() は、 書き込んだバイト数、またはエラー時に **FALSE** を返します。

注意

注意: (Windowsのように)バイナリとテキストファイルの形式が異なるシステムにおいては、ファイルをオープンする際に fopen()の mode パラメータに 'b' を指定する必要があります。

注意: fopen() を使用して追記モードでオープンした *handle* の場合、 **fwrite()** はアトミックになります (ただし、一部のプラットフォームにおいて *string* がファイルシステムのブロックサイズを超えない場合、 そしてローカルファイルシステム上のファイルである場合に限ります)。 アトミックであるとは、つまり **fwrite()** をコールする前にリソースを flock() する必要がないということです。データの書き込みが中断されることはありません。

例

Example#1 簡単な fwrite() の例

```
<?php
$filename = 'test.txt';
$somecontent = "Add this to the file¥n";

// ファイルが存在しかつ書き込み可能かどうか確認します
if (is_writable($filename)) {

    // この例では$filenameを追加モードでオープンします。
    // ファイルポインタはファイルの終端になりますので
    // そこがfwrite()で$somecontentが追加される位置になります。
    if (!$handle = fopen($filename, 'a')) {
         echo "Cannot open file ($filename)";
         exit;
    }

    // オープンしたファイルに$somecontentを書き込みます
    if (fwrite($handle, $somecontent) === FALSE) {
        echo "Cannot write to file ($filename)";
        exit;
    }

    echo "Success, wrote ($somecontent) to file ($filename)";

    fclose($handle);

} else {
    echo "The file $filename is not writable";
}
?>
```

参考

- fread()
- fopen()
- fsockopen()
- popen()
- file_get_contents()

---

glob

(PHP 4 >= 4.3.0, PHP 5)

glob — パターンにマッチするパス名を探す

説明

array **glob** (string `$pattern` [, int `$flags`])

glob() 関数は libc の glob() 関数で使われるルールに基づいて *pattern* にマッチする全てのパス名を検索します。 ルールは、一般のシェルで使われるルールと似ています。

パラメータ

pattern

パターン。チルダの展開やパラメータ置換は行いません。

flags

有効なフラグは次のとおりです。

- **GLOB_MARK** - 各戻り値にスラッシュを追加します
- **GLOB_NOSORT** - ディレクトリに存在するファイルを返します (ソートはされません)
- **GLOB_NOCHECK** - 検索パターンにマッチするファイルが見つからない場合に、 検索パターン自身を返します
- **GLOB_NOESCAPE** - バックスラッシュによるメタ文字のクォートを行いません

- **GLOB_BRACE** - {a,b,c} を展開し「a」、「b」あるいは「c」のいずれかにマッチさせます
- **GLOB_ONLYDIR** - パターンにマッチするディレクトリのみを返します
- **GLOB_ERR** - (ディレクトリが読めないなどの) 読み込みエラー時に停止します。デフォルトではエラーは無視されます。

返り値

マッチするファイル／ディレクトリを含む配列を返します。 マッチするファイルがなかった場合には空の配列、 そして失敗した場合には **FALSE** を返します。

変更履歴

バージョン	説明
5.1.0	**GLOB_ERR** が追加されました。
4.3.3	PHP 4.3.3 以前では、**GLOB_ONLYDIR** が、 Windows やその他の GNU C ライブラリを使用しないシステムでも利用できるようになりました。

例

Example#1 glob() が opendir() と関連する関数群の代替策になるかを示す簡便な方法

```
<?php
foreach (glob("*.txt") as $filename) {
    echo "$filename size " . filesize($filename) . "¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
funclist.txt size 44686
funcsummary.txt size 267625
quickref.txt size 137820
```

注意

注意: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

注意: この関数が使用できないシステムも存在します (例: 昔の Sun OS など)。

注意: **GLOB_BRACE** フラグは、Solaris などの非 GNU システムでは動作しないことがあります。

参考

- opendir()
- readdir()
- closedir()
- fnmatch()

---

is_dir

(PHP 4, PHP 5)

is_dir — ファイルがディレクトリかどうかを調べる

説明

bool **is_dir** (string `$filename`)

指定したファイルがディレクトリかどうかを調べます。

パラメータ

`filename`

ファイルへのパス。*`filename`* が相対パスの場合は、現在の作業ディレクトリからの相対パスとして処理します。

返り値

ファイルがが存在して、かつそれがディレクトリであれば **TRUE**、それ以外の場合は **FALSE** を返します。

例

Example#1 is_dir() の例

```
<?php
var_dump(is_dir('a_file.txt')) . "¥n";
var_dump(is_dir('bogus_dir/abc')) . "¥n";

var_dump(is_dir('..')); //一つ上のディレクトリ
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(false)
bool(true)
```

注意

> **注意**: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- chdir()
- dir
- opendir()
- is_file()
- is_link()

---

is_executable

(PHP 4, PHP 5)

is_executable — ファイルが実行可能かどうかを調べる

説明

bool **is_executable** (string `$filename`)

ファイルが実行可能かどうかを調べます。

パラメータ

`filename`

> ファイルへのパス。

返り値

ファイルが存在し、かつそれが実行可能な場合に **TRUE**、 エラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	**is_executable()** は Windows でも使用可能となりました。

例

Example#1 is_executable() の例

```
<?php

$file = '/home/vincent/somefile.sh';
```

```
if (is_executable($file)) {
    echo $file.'  は実行可能です';
} else {
    echo $file.'  は実行可能ではありません';
}

?>
```

注意

> **注意**: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- is_file()
- is_link()

---

is_file

(PHP 4, PHP 5)

is_file — 通常ファイルかどうかを調べる

説明

bool **is_file** (string `$filename`)

指定したファイルが通常のファイルかどうかを調べます。

パラメータ

`filename`

> ファイルへのパス。

返り値

ファイルが存在し、かつそれが通常のファイルである場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

例

Example#1 is_file() の例

```
<?php
var_dump(is_file('a_file.txt')) . "¥n";
var_dump(is_file('/usr/bin/')) . "¥n";
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(false)
```

注意

> **注意**: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- is_dir()

- is_link()

---

is_link

(PHP 4, PHP 5)

is_link — ファイルがシンボリックリンクかどうかを調べる

説明

bool **is_link** (string `$filename`)

指定したファイルがシンボリックリンクかどうかを調べます。

パラメータ

`filename`

ファイルへのパス。

返り値

filename が存在し、かつシンボリックリンクであれば **TRUE**、 それ以外の場合に **FALSE** を返します。

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- is_dir()
- is_file()
- readlink()

---

is_readable

(PHP 4, PHP 5)

is_readable — ファイルが読み込み可能かどうかを知る

説明

bool **is_readable** (string `$filename`)

ファイルが読み込み可能かどうかを調べます。

パラメータ

`filename`

ファイルへのパス。

返り値

`filename` で指定したファイルあるいはディレクトリが存在し、 それが読み込み可能であれば **TRUE**、それ以外の場合に **FALSE** を返します。

例

Example#1 is_readable() の例

```
<?php
$filename = 'test.txt';
if (is_readable($filename)) {
```

```
    echo 'このファイルは読み込み可能です';
} else {
    echo 'このファイルは読み込み可能ではありません';
}
?>
```

注意

PHP は、Web サーバを実行しているユーザ ID (たいていは 'nobody') でファイルにアクセスすることを覚えておいてください。 PHP 5.1.5 より前のバージョンでは、セーフモードの制限は働きません。

> **注意**: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

> **注意**: チェックは、実効ユーザではなく実ユーザの UID/GID で行います。

参考

- is_writable()
- file_exists()
- fgets()

---

is_uploaded_file

(PHP 4 >= 4.0.3, PHP 5)

is_uploaded_file — HTTP POST でアップロードされたファイルかどうかを調べる

説明

bool **is_uploaded_file** (string `$filename`)

filename という名前のファイルが HTTP POST によりアップロードされたものである場合に **TRUE** を返します。 悪意のあるユーザがスクリプトをだまして、 本来見られてはいけないはずのファイル (*/etc/passwd* など) にアクセスすることを防止したい場合に、この関数は有用です。

この種の確認は、アップロードされたファイルに関して何でもできる場合には、 その内容をユーザ、または同じシステム上の他のユーザにさえ 暴かれる可能性があるため、特に重要です。

適切に動作させるため、関数 **is_uploaded_file()** は $_FILES['userfile']['tmp_name'] のような引数を必要とします。 アップロードされたファイルのクライアントマシン上での名前 $_FILES['userfile']['name'] では動作しません。

パラメータ

`filename`

> 調べたいファイル名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 is_uploaded_file() の例

```
<?php

if (is_uploaded_file($_FILES['userfile']['tmp_name'])) {
   echo "ファイル  ". $_FILES['userfile']['name'] ."  のアップロードに成功しました。\n";
   echo "その中身を表示します\n";
   readfile($_FILES['userfile']['tmp_name']);
} else {
   echo "おそらく何らかの攻撃を受けました。";
    echo "ファイル名  '". $_FILES['userfile']['tmp_name'] . "'.";
}

?>
```

Example#2 PHP 4 < 4.0.3 用の is_uploaded_file() の例

以下のサンプルは PHP 4.0.2 以降の PHP 4 では*動きません*。 これはこのバージョン以降で PHP の内部処理が変更されたためです。

```
<?php
/* アップロードされたファイルのテスト */
function is_uploaded_file_4_0_2($filename)
{
    if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
        $tmp_file = dirname(tempnam('', ''));
    }
    $tmp_file .= '/' . basename($filename);
    /* ユーザは php.ini で最後にスラッシュを付けているかも知れない... */
    return (ereg_replace('/+', '/', $tmp_file) == $filename);
}

/* 以下はこの関数の使用方法。これら古いバージョンには
 * move_uploaded_file() もない。 */
if (is_uploaded_file_4_0_2($HTTP_POST_FILES['userfile'])) {
    copy($HTTP_POST_FILES['userfile'], "/place/to/put/uploaded/file");
} else {
    echo "ファイルアップロード攻撃を受けた可能性あり： ファイル名 '$HTTP_POST_FILES[userfile]'";
}
?>
```

参考

- move_uploaded_file()
- 簡単な使用例については ファイルアップロードの処理

---

is_writable

(PHP 4, PHP 5)

is_writable — ファイルが書き込み可能かどうかを調べる

説明

bool **is_writable** (string `$filename`)

filename が存在して、かつそれが書き込み可能であれば **TRUE**を返します。引数filenameはディレクトリ名とすることができ、 ディレクトリが書き込み可能であることを調べることが可能です。

PHP は、Web サーバが実行されているユーザ ID('nobody' が多い) でファイルにアクセスすることを覚えておいてください。 セーフモードの制限は働きません。

パラメータ

filename

調べたいファイル名。

返り値

filename が存在して書き込み可能な場合に **TRUE** を返します。

例

Example#1 is_writable() の例

```
<?php
$filename = 'test.txt';
if (is_writable($filename)) {
    echo 'このファイルは書き込み可能です';
} else {
    echo 'このファイルは書き込みできません';
}
?>
```

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- is_readable()
- file_exists()
- fwrite()

---

is_writeable

(PHP 4, PHP 5)

is_writeable — is_writable() のエイリアス

説明

この関数は次の関数のエイリアスです。 is_writable().

---

lchgrp

(PHP 5 >= 5.1.2)

lchgrp — シンボリックリンクのグループ所有権を変更する

説明

bool **lchgrp** (string `$filename` , mixed `$group`)

シンボリックリンク *`filename`* のグループを *`group`* に変更しようと試みます。

スーパーユーザは、 シンボリックリンクのグループを任意のものに変更できます。 その他のユーザは、自分自身がメンバーの一員となっているグループにのみ変更できます。

パラメータ

`filename`

シンボリックリンクへのパス。

`group`

グループ名あるいはグループ番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

注意: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- chgrp()
- lchown()
- chown()
- chmod()

---

lchown

(PHP 5 >= 5.1.2)

lchown — シンボリックリンクの所有者を変更する

説明

bool **lchown** (string `$filename` , mixed `$user`)

シンボリックリンク *`filename`* の所有者を *`user`* に変更しようと試みます。

シンボリックリンクの所有者を変更できるのは、スーパーユーザのみです。

パラメータ

`filename`

ファイルへのパス。

`user`

ユーザ名あるいはユーザ番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

注意: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- chgrp()
- lchgrp()
- chgrp()
- chmod()

---

link

(PHP 4, PHP 5)

link — ハードリンクを作成する

説明

bool **link** (string `$target` , string `$link`)

link() はハードリンクを作成します。

パラメータ

`target`

リンクの対象。

`link`

リンクの名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数では、 リモートファイル を使用することはできません。これは、処理されるファイルがサーバのファイルシステムによりアクセスできる必要があるためです。

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- symlink()
- readlink()
- linkinfo()

---

linkinfo

(PHP 4, PHP 5)

linkinfo — リンクに関する情報を取得する

説明

int **linkinfo** (string $path)

リンクに関する情報を取得します。

この関数を使用して (*path* が指している) リンクが実際に存在するかどうかを、 (*stat.h* で定義されている S_ISLNK マクロと同じ方法で) チェックします。

パラメータ

path

リンクへのパス。

返り値

linkinfo()は、*lstat* システムコールで返された Unix C 言語の stat 構造体の *st_dev* フィールドを返します。 0 を返し、エラーの場合に **FALSE** を返します。

例

Example#1 linkinfo() の例

```
<?php
echo linkinfo('/vmlinuz'); // 835
?>
```

注意

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- symlink()
- link()
- readlink()

---

lstat

(PHP 4, PHP 5)

lstat — ファイルあるいはシンボリックリンクの情報を取得する

説明

array **lstat** (string $filename)

filename という名前のファイル、 またはシンボリックリンクの情報を得ます。

パラメータ

filename

ファイルあるいはシンボリックリンクへのパス。

返り値

lstat() が返す配列の内容については stat() のマニュアルをご覧ください。 この関数の動作は stat() 関数と同じですが、 パラメータ *filename* がシンボリックリンクであった場合に、 シンボリック先のファイルのステータスではなく、 シンボリックリンクのステータスが返されるところが異なります。

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが stat() ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- stat()

---

mkdir

(PHP 4, PHP 5)

mkdir — ディレクトリを作る

説明

bool **mkdir** (string `$pathname` [, int `$mode` [, bool `$recursive` [, resource `$context`]]])

指定したディレクトリを作成します。

パラメータ

pathname

ディレクトリのパス。

mode

モードは 0777 がデフォルトです。 これは最も緩やかなアクセス制限を意味します。 モードに関する詳細は chmod() をご覧ください。

注意: Windows では *mode* は無視されます。

モードを八進数で指定したくなることもあるかもしれません。 その場合は先頭にゼロをつける必要があります。 また、モードは、現在設定されている umask の影響も受けます。 umask を変更するには umask() を使用します。

recursive

context

注意: コンテキストのサポートは、 PHP 5.0.0 で追加されました。*contexts* の説明に関しては、 ストリーム を参照してください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	*recursive* パラメータが追加されました。
5.0.0	PHP 5.0.0 以降、**mkdir()** は いくつかの URL ラッパーを併用することが可能です。 **mkdir()** をサポートしているラッパーの一覧については、 サポートされるプロトコル/ラッパー を参照ください。
4.2.0	*mode* パラメータがオプションとなりました。

例

Example#1 mkdir() の例

```
<?php
mkdir("/path/to/my/dir", 0700);
?>
```

注意

> **注意**: セーフモード が有効の場合、PHP は、操作を行うディレクトリが、実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- rmdir()

---

move_uploaded_file

(PHP 4 >= 4.0.3, PHP 5)

move_uploaded_file — アップロードされたファイルを新しい位置に移動する

説明

bool **move_uploaded_file** (string `$filename` , string `$destination`)

この関数は、*filename* で指定されたファイルが (PHP の HTTP POST アップロード機構によりアップロードされたという意味で) 有効なアップロードファイルであるかどうかを確認します。 そのファイルが有効な場合、*destination* で指定したファイル名に移動されます。

この種の確認は、アップロードされたファイルに関して何でもできる場 合には、その内容をユーザ、または同じシステム上の他のユーザにさえ 暴かれる可能性があるため、特に重要です。

パラメータ

filename

> アップロードしたファイルのファイル名。

destination

> ファイルの移動先。

返り値

filename が有効なアップロードファイルでない場合、 処理は行われず、**move_uploaded_file()** は **FALSE** を返します。

filename が有効なアップロードファイルであるが、 何らかの理由により、移動できない場合、処理は行われず、 **move_uploaded_file()** は **FALSE** を返します。加えて、警告が出力されます。

注意

> **注意**: **move_uploaded_file()** は セーフモード と open_basedir の両者を考慮しています。 しかしながら、アップロードされたファイルを移動する *destination* パスのみ制限が設けられます。 そこでは *filename* がそれらの制限に抵触する可能性があるためです。 **move_uploaded_file()** は PHP を通じてアップロードされたファイルのみを移動できるようにすることで この操作の安全性を保証しています。

警告

コピー先のファイルが既に存在する場合、上書きされます。

参考

- is_uploaded_file()
- 簡単な使用例については ファイルアップロードの処理

---

parse_ini_file

(PHP 4, PHP 5)

parse_ini_file — 設定ファイルをパースする

説明

array **parse_ini_file** (string $filename [, bool $process_sections])

parse_ini_file() は、 *filename* で指定した ini ファイルをロードし、 連想配列としてその設定値を返します。

初期ファイルの構造は、*php.ini* の構造と同じです。

パラメータ

filename

パースしたい ini ファイルのファイル名。

process_sections

直近の *process_sections* パラメータに **TRUE** を設定することにより、セクション名と設定が含まれた多次元の配列を 得ることができます。デフォルトでは、 *process_sections* は **FALSE** です。

返り値

設定を連装配列形式で返します。

変更履歴

バージョン	説明
5.2.4	数字から始まるキーおよびセクション名は、PHP では 整数 として評価されます。 よって、0 で始まる数字は 8 進数として評価され、0x で始まる数字は 16 進数として評価されます。
5.0.0	値をダブルクォートで囲めばその中で改行を使用することもできます。
4.2.1	この関数は セーフモード と open_basedir の影響を受けます。

例

Example#1 *sample.ini* の内容

```
; これは設定ファイルのサンプルです。
; php.ini と同様、';' で始まる行はコメントです。

[first_section]
one = 1
five = 5
animal = BIRD

[second_section]
path = "/usr/local/bin"
URL = "http://www.example.com/~username"
```

Example#2 parse_ini_file() の例

定数も ini ファイル上でパースされます。 そのため、**parse_ini_file()** をコールする前に ini ファイル上の値として定数を定義した場合、返り値に統合されます。 ini ファイル上の値だけが評価されます。以下は例です：

```
<?php

define('BIRD', 'Dodo bird');

// セクションを無視してパースします。
$ini_array = parse_ini_file("sample.ini");
print_r($ini_array);

// セクションを意識してパースします。
$ini_array = parse_ini_file("sample.ini", true);
print_r($ini_array);

?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [one] => 1
    [five] => 5
    [animal] => Dodo bird
    [path] => /usr/local/bin
    [URL] => http://www.example.com/~username
)
Array
(
```

```
    [first_section] => Array
       (
            [one] => 1
            [five] => 5
             [animal] = Dodo bird
       )

    [second_section] => Array
       (
             [path] => /usr/local/bin
              [URL] => http://www.example.com/~username
       )

)
```

注意

注意: この関数は、*php.ini* ファイルには何もしません。 このファイルはスクリプトを実行している時には既に処理されています。 この関数は、アプリケーション個有の設定ファイルを読み込む際に使用可能です。

注意: ini ファイル上の値に英数字ではないものがある場合、 ダブルクォート(")で囲う必要があります。

注意: ini ファイル上でキーとして使ってはいけない単語があります。 それらは null, yes, no, true, false などが含まれます。 null, no および false は "" となり、yes および true は "1" となります。 次の文字 *{}|&~![()"* は、キーで使ってはいけません。 また、値の中で特別な意味を持ちます。

---

pathinfo

(PHP 4 >= 4.0.3, PHP 5)

pathinfo — ファイルパスに関する情報を返す

説明

mixed **pathinfo** (string $path [, int $options])

pathinfo() は、*path* に関する情報を有する連想配列を返します。

パラメータ

path

調べたいパス。

options

どの要素を返すのかをオプションのパラメータ *options* で指定します。これは **PATHINFO_DIRNAME**、 **PATHINFO_BASENAME**、 **PATHINFO_EXTENSION** および **PATHINFO_FILENAME** の組み合わせとなります。 デフォルトではすべての要素を返します。

返り値

以下の要素を含む連装配列を返します。 *dirname* (ディレクトリ名)、*basename* (ファイル名) そして、もし存在すれば *extension* (拡張子)。

options を使用すると、 すべての要素を選択しない限りこの関数の返り値は文字列となります。

変更履歴

バージョン	説明
5.2.0	定数 **PATHINFO_FILENAME** が追加されました。

例

Example#1 pathinfo() の例

```php
<?php
$path_parts = pathinfo('/www/htdocs/index.html');

echo $path_parts['dirname'], "¥n";
echo $path_parts['basename'], "¥n";
echo $path_parts['extension'], "¥n";
echo $path_parts['filename'], "¥n"; // PHP 5.2.0 以降
?>
```

上の例の出力は以下となります。

```
/www/htdocs
index.html
html
index
```

注意

注意: カレントのパスに関する情報を取得するには、 定義済みの変数 のセクションをご覧ください。

参考

- dirname()
- basename()
- parse_url()
- realpath()

---

pclose

(PHP 4, PHP 5)

pclose — プロセスのファイルポインタをクローズする

説明

int **pclose** (resource `$handle`)

popen() でオープンしたパイプへの ファイルポインタをクローズします。

パラメータ

`handle`

ファイルポインタは有効なものでなければならず、また popen() で正常にオープンされたものである必要があります。

返り値

実行していたプロセスの終了ステータスを返します。

参考

- popen()

---

popen

(PHP 4, PHP 5)

popen — プロセスへのファイルポインタをオープンする

説明

resource **popen** (string `$command` , string `$mode`)

command で指定したコマンドのフォークによってできたプロセスへのパイプをオープンします。

パラメータ

`command`

コマンド。

`mode`

モード。

返り値

fopen() により返されたファイルポインタと同様のものを返しますが、 それは(読み書きのいずれか一方でのみ使われる)片方向ストリームであり、 pclose() によりクローズされなければならないところが異なります。 このポインタは、fgets()、fgetss() および fwrite() のいずれかで使うことができます。

エラーが発生した場合は **FALSE** を返します。

例

Example#1 popen() の例

```
<?php
$handle = popen("/bin/ls", "r");
?>
```

実行すべきコマンドが見つからない場合には、正常なリソースが返されます。 おかしなことと思われるかもしれませんが、これには意味があります。 これによってシェルから返されたエラーメッセージにアクセスすることができるのです。

Example#2 popen() の例

```
<?php
error_reporting(E_ALL);

/* リダイレクトにより、標準エラー出力を取得します */
$handle = popen('/path/to/spooge 2>&1', 'r');
echo "'$handle'; " . gettype($handle) . "¥n";
$read = fread($handle, 2096);
echo $read;
pclose($handle);
?>
```

注意

> **注意**: 双方向(two-way)のサポートを求めているのなら、 proc_open() を使用してください。
>
> **注意**: セーフモード が有効な場合、 safe_mode_exec_dirの中にある実行プログラムのみ実行可能です。実際的な理由により、現在、実行プログラムへのパスに ..を含めることはできません。

警告

セーフモードが有効な場合、コマンド文字列は escapeshellcmd() でエスケープされます。つまり、*echo y ¦ echo x* は、*echo y ¥¦ echo x* となります。

参考

- pclose()
- fopen()
- proc_open()

---

readfile

(PHP 4, PHP 5)

readfile — ファイルを出力する

説明

int **readfile** (string `$filename` [, bool `$use_include_path` [, resource `$context`]])

ファイルを読んで標準出力に書き出します。

パラメータ

`filename`

> 読み込もうとするファイルの名前。

`use_include_path`

> オプションの2番目の引数を使用して、これに**TRUE**を設定することにより、 include_path のファイルの検索も行うことができます。

`context`

コンテキストストリームリソース。

返り値

ファイルから読み込んだバイト数を返します。エラーが起こると **FALSE**を返し、また@**readfile()**という名前でコールされない限り、 エラーメッセージが表示されます。

注意

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

注意: コンテキストのサポートは、 PHP 5.0.0 で追加されました。*contexts* の説明に関しては、 ストリーム を参照してください。

参考

- fpassthru()
- file()
- fopen()
- include()
- require()
- virtual()
- file_get_contents()
- サポートされるプロトコル/ラッパー

---

readlink

(PHP 4, PHP 5)

readlink — シンボリックリンク先を返す

説明

string **readlink** (string `$path`)

readlink() は同名の C 関数と同じ動作をします。

パラメータ

path

シンボリックリンクのパス。

返り値

シンボリックリンク・パスの内容を返します。エラーの場合は **FALSE** を返します。

例

Example#1 readlink() の例

```
<?php

//  出力例  /boot/vmlinux-2.4.20-xfs
echo readlink('/vmlinuz');

?>
```

注意

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- is_link()
- symlink()
- linkinfo()

---

realpath

(PHP 4, PHP 5)

realpath — 絶対パス名を返す

説明

string **realpath** (string $path)

realpath() は、 入力 *path* のシンボリックリンクをすべて展開し、 「/./」「/../」「/」などの参照をすべて解決することにより、正規化した絶対パスを返します。

パラメータ

path

調べたいパス。

返り値

成功した場合は正規化した絶対パス名を返します。 返されるパスはシンボリックリンクや「/./」「/../」要素を含みません。

realpath() は、 たとえばファイルが存在しないなどの失敗時に **FALSE** を返します。 BSD システムでは **realpath()** は最後の *path* コンポーネントのみが存在しない場合には 失敗となりません。一方、他のシステムではそのような場合にも **FALSE** を返します。

例

Example#1 realpath() の例

```
<?php
chdir('/var/www/');
echo realpath('./../../etc/passwd');
?>
```

上の例の出力は以下となります。

```
/etc/passwd
```

参考

- basename()
- dirname()
- pathinfo()

---

rename

(PHP 4, PHP 5, PECL zip:1.1.0-1.4.1)

rename — ファイルをリネームする

説明

bool **rename** (string $oldname , string $newname [, resource $context])

oldname を *newname* にリネームしようと試みます。

パラメータ

oldname

注意: 変更前の名前。 *oldname* で使用されるラッパーは、 *newname* で使用するラッパーと適合 *している必要があります* 。

newname

変更後の名前。

context

注意: コンテキストのサポートは、 PHP 5.0.0 で追加されました。*contexts* の説明に関しては、 ストリーム を参照してください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	**rename()** は いくつかの URL ラッパーを併用することができるようになりました。 **rename()** をサポートしているラッパーの一覧については、 サポートされるプロトコル/ラッパー を参照ください。
4.3.3	**rename()** は *nix ベースシステムで パーティション越しにファイル名を変更できるようになりました。

例

Example#1 rename() の例

```
<?php
rename("/tmp/tmp_file.txt", "/home/user/login/docs/my_file.txt");
?>
```

参考

- copy()
- unlink()
- move_uploaded_file()

---

rewind

(PHP 4, PHP 5)

rewind — ファイルポインタの位置を先頭に戻す

説明

bool **rewind** (resource $handle)

handle のファイル位置指示子を、 ファイルストリームの先頭にセットします。

注意: 追記モード ("a" もしくは "a+") でファイルをオープンした場合、 ファイル位置によらずファイルに書き込まれるデータは常に追加されます。

パラメータ

handle

ファイルポインタは有効なものでなければならず、 また fopen() で正常にオープンされたファイルを指している必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fseek()
- ftell()

---

rmdir

(PHP 4, PHP 5)

rmdir — ディレクトリを削除する

説明

bool **rmdir** (string `$dirname` [, resource `$context`])

dirname で指定されたディレクトリを 削除しようと試みます。ディレクトリは空でなくてはならず、また 適切なパーミッションが設定されていなければなりません。

パラメータ

dirname

ディレクトリへのパス。

context

注意: コンテキストのサポートは、 PHP 5.0.0 で追加されました。*contexts* の説明に関しては、 ストリーム を参照してください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	PHP 5.0.0 以降、**rmdir()** は いくつかの URL ラッパーを併用することが可能です。 **rmdir()** をサポートしているラッパーの一覧については、 サポートされるプロトコル/ラッパー を参照ください。

注意

注意: セーフモード が有効の場合、PHP は、操作を行うディレクトリが、実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- mkdir()
- unlink()

---

set_file_buffer

(PHP 4, PHP 5)

set_file_buffer — stream_set_write_buffer() のエイリアス

説明

この関数は次の関数のエイリアスです。 stream_set_write_buffer().

---

stat

(PHP 4, PHP 5, PECL maxdb:7.5.00.24-7.6.00.38)

stat — ファイルに関する情報を取得する

説明

array **stat** (string `$filename`)

filename で指定されたファイルに ついての統計情報を取得します。 *filename* がシンボリックリンクの場合、 シンボリックリンクではなくファイルの実体の統計情報が返されます。

lstat() はシンボリックリンクの統計情報を返すという違いを除いて **stat()** と等価です。

パラメータ

filename

ファイルへのパス。

返り値

stat() と fstat() の結果のフォーマット

数値	連想 (PHP 4.0.6 以上)	説明
0	dev	デバイス番号
1	ino	inode 番号
2	mode	inode プロテクトモード
3	nlink	リンク数
4	uid	所有者のユーザ ID
5	gid	所有者のグループ ID
6	rdev	inode デバイス の場合、デバイスの種類(*)
7	size	バイト単位のサイズ
8	atime	最終アクセス時間 (Unix タイムスタンプ)
9	mtime	最終修正時間 (Unix タイムスタンプ)
10	ctime	最終 inode 変更時間 (Unix タイムスタンプ)
11	blksize	ファイル IO のブロックサイズ(*)
12	blocks	ブロックの確保数(*)

* st_blksize タイプをサポートするシステムでのみ有効です。 その他のシステム(例えば Windows)では -1 を返します。

stat() はエラーの場合 **FALSE** を返します。

エラー / 例外

失敗した場合は **E_WARNING** が発生します。

変更履歴

バージョン	説明
4.0.6	数値添字の配列に加えて、 各パラメータ毎の連想配列としてもアクセスできるようになりました。

注意

注意: この関数の結果はキャッシュされます。詳細は、clearstatcache() を参照してください。

ヒント

PHP 5.0.0 以降、この関数は、 *何らかの* URL ラッパーと組合せて使用することができます。どのラッパーが **stat()** ファミリーをサポートしているかのリストについては、サポートされるプロトコル/ラッパー を参照してください。

参考

- lstat()
- fstat()
- filemtime()
- filegroup()

---

symlink

(PHP 4, PHP 5)

symlink — シンボリックリンクを作成する

説明

bool **symlink** (string `$target` , string `$link`)

symlink()は、指定されたリンク名 *link* で既存のファイル *target* へのシンボリックリンクを作成します。

パラメータ

target

リンクの対象。

link

リンクの名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- ハードリンクを作るには link()
- readlink() や linkinfo()

---

tempnam

(PHP 4, PHP 5)

tempnam — 一意なファイル名を生成する

説明

string **tempnam** (string `$dir` , string `$prefix`)

一意なテンポラリファイル名を、パーミッションを 0600 に設定し、 指定したディレクトリに作成します。 指定したディレクトリが存在しない場合、 **tempnam()** はシステムのテンポラリディレクトリにあるファイル名を生成し、 その名前を返します。

パラメータ

dir

テンポラリファイルを作成したいディレクトリ。

prefix

作成されるテンポラリファイルのプレフィックス。

返り値

新しいテンポラリファイル名を返し、失敗した場合には **FALSE** を返します。

変更履歴

バージョン	説明
4.0.6	PHP 4.0.6 より前では、関数 **tempnam()** の動作はシステムに依存していました。Windows において TMP 環境変数は *dir* パラメータを上書きします。Linux においては、 TMPDIR 環境変数が優先されます。一方 SVR4 においては、 指定したディレクトリが存在する場合は、常に *dir* パラメータを使用します。疑問がある場合は、tempnam(3) に関するシステムドキュメントを参照ください。
4.0.3	この関数の動作は 4.0.3 で変更されました。文字列が生成された時間の間や スクリプトがファイルの作成にとりかかる前にファイルシステムに現れ る可能性がある場合のファイルとの競合を回避するためにもテンポラリ ファイルは作成されます。このファイルは自動的に削除されないため、 不要となった場合にはこのファイルを削除する必要があることに注意してください。

例

Example#1 tempnam() の例

```
<?php
$tmpfname = tempnam("/tmp", "FOO");

$handle = fopen($tmpfname, "w");
```

```
fwrite($handle, "writing to tempfile");
fclose($handle);

// ここで何か行う

unlink($tmpfname);
?>
```

注意

注意: PHP が指定されたパラメータ *dir* にファイルを生成することができない場合、 システム標準のフォールバックが実行されます。

参考

- tmpfile()
- sys_get_temp_dir()
- unlink()

---

tmpfile

(PHP 4, PHP 5)

tmpfile — テンポラリファイルを作成する

説明

resource **tmpfile** (void)

書き込み可のモード (w+) でユニークな名前を有するテンポラリファイルを作成し、 ファイルハンドルを返します。

ファイルが (fclose() を用いて) クローズされた時、 またはスクリプトが終了された際に自動的にファイルが削除されます。

詳細については、関数 *tmpfile(3)* のシステムドキュメント、およびヘッダファイル *stdio.h* を参照ください。

返り値

ファイルのハンドルを返します。 これは、 fopen() により返されるハンドルと同じものです。 失敗した場合には **FALSE** を返します。

例

Example#1 tmpfile() の例

```
<?php
$temp = tmpfile();
fwrite($temp, "writing to tempfile");
fseek($temp, 0);
echo fread($temp, 1024);
fclose($temp); // ファイルを削除します
?>
```

上の例の出力は以下となります。

```
writing to tempfile
```

参考

- tempnam()
- sys_get_temp_dir()

---

touch

(PHP 4, PHP 5)

touch — ファイルの最終アクセス時刻および最終更新日をセットする

説明

bool **touch** (string `$filename` [, int `$time` [, int `$atime`]])

filename で指定されたファイルの最終更新日を、 *time* で指定された値に セットしようと試みます。 パラメータの数にかかわらず、アクセス時刻は常に変更されることに注意しましょう。

ファイルが存在しない場合、ファイルが生成されます。

パラメータ

filename

処理したいファイルの名前。

time

設定する時刻。 *time* を省略した場合は、 現在時刻を使用します。

atime

指定されたファイルの最終アクセス時刻が *atime* にセットされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 touch() の例

```
<?php
if (touch($FileName)) {
    echo "$FileName modification time has been changed to present time";
} else {
    echo "Sorry, could not change modification time of $FileName";
}
?>
```

注意

警告

現在は、Windows 環境のもとでは この関数によってディレクトリの最終更新日を変更することはできません。

---

umask

(PHP 4, PHP 5)

umask — 現在の umask を変更する

説明

int **umask** ([int $mask])

umask() は PHP の umask を *mask* & 0777 にセットし、 元の umask 値を返します。 PHP がサーバ・モジュールとして動作中の場合、 各リクエストが終了するたびに umask は元の値に戻されます。

パラメータ

mask

新しい umask。

返り値

umask()を引数無しで実行すると、単に現在の umask値を返します。

例

Example#1 umask() の例

```
<?php
$old = umask(0);
chmod("/path/some_dir/some_file.txt", 0755);
umask($old);

// チェック
```

```
if ($old != umask()) {
    die('An error occured while changing back the umask');
}
?>
```

注意

> **注意**: マルチスレッドな Web サーバでこの関数を使用することは避けてください。 ファイルを生成後、chmod() を使用してファイル権限を 変更するのがより良い方法です。全て同じ umask が使用されるので、 **umask()** の使用は、スクリプトを同時に実行する場合や Web サーバ自身の予期しない動作を引き起こす原因になる可能性があります。

---

unlink

(PHP 4, PHP 5)

unlink — ファイルを削除する

説明

bool **unlink** (string `$filename` [, resource `$context`])

`filename` を削除します。 Unix C 言語の関数 unlink() と動作は同じです。

パラメータ

`filename`

ファイルへのパス。

`context`

> **注意**: コンテキストのサポートは、 PHP 5.0.0 で追加されました。*contexts* の説明に関しては、 ストリーム を参照してください。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

変更履歴

バージョン	説明
5.0.0	PHP 5.0.0 以降、**unlink()**は いくつかのURL ラッパと共に使用することもできます。 **unlink()**をサポートするラッパのリストについては サポートされるプロトコル/ラッパーをご覧ください。

参考

- ディレクトリを削除する場合は rmdir()

目次

- fgetcsv — ファイルポインタから行を取得し、CSVフィールドを処理する
- fgets — ファイルポインタから 1 行取得する
- fgetss — ファイルポインタから 1 行取り出し、HTML タグを取り除く
- file_exists — ファイルまたはディレクトリが存在するかどうか調べる
- file_get_contents — ファイルの内容を全て文字列に読み込む
- file_put_contents — 文字列をファイルに書き込む
- file — ファイル全体を読み込んで配列に格納する
- fileatime — ファイルの最終アクセス時刻を取得する
- filectime — ファイルの inode 変更時刻を取得する
- filegroup — ファイルのグループを取得する
- fileinode — ファイルの inode を取得する
- filemtime — ファイルの更新時刻を取得する
- fileowner — ファイルの所有者を取得する
- fileperms — ファイルのパーミッションを取得する
- filesize — ファイルのサイズを取得する
- filetype — ファイルタイプを取得する
- flock — 汎用のファイルロックを行う
- fnmatch — ファイル名がパターンにマッチするか調べる
- fopen — ファイルまたは URL をオープンする
- fpassthru — ファイルポインタ上に残っているすべてのデータを出力する
- fputcsv — 行を CSV 形式にフォーマットし、ファイルポインタに書き込む
- fputs — fwrite のエイリアス
- fread — バイナリセーフなファイルの読み込み
- fscanf — フォーマットに基づきファイルからの入力を処理する
- fseek — ファイルポインタを移動する
- fstat — オープンしたファイルポインタからファイルに関する情報を取得する
- ftell — ファイルポインタから読み書きの位置を取得する
- ftruncate — ファイルを指定した長さに丸める
- fwrite — バイナリセーフなファイル書き込み処理
- glob — パターンにマッチするパス名を探す
- is_dir — ファイルがディレクトリかどうかを調べる
- is_executable — ファイルが実行可能かどうかを調べる
- is_file — 通常ファイルかどうかを調べる
- is_link — ファイルがシンボリックリンクかどうかを調べる
- is_readable — ファイルが読み込み可能かどうかを知る
- is_uploaded_file — HTTP POST でアップロードされたファイルかどうかを調べる
- is_writable — ファイルが書き込み可能かどうかを調べる
- is_writeable — is_writable のエイリアス
- lchgrp — シンボリックリンクのグループ所有権を変更する
- lchown — シンボリックリンクの所有者を変更する
- link — ハードリンクを作成する
- linkinfo — リンクに関する情報を取得する
- lstat — ファイルあるいはシンボリックリンクの情報を取得する
- mkdir — ディレクトリを作る
- move_uploaded_file — アップロードされたファイルを新しい位置に移動する
- parse_ini_file — 設定ファイルをパースする
- pathinfo — ファイルパスに関する情報を返す
- pclose — プロセスのファイルポインタをクローズする
- popen — プロセスへのファイルポインタをオープンする
- readfile — ファイルを出力する
- readlink — シンボリックリンク先を返す
- realpath — 絶対パス名を返す
- rename — ファイルをリネームする
- rewind — ファイルポインタの位置を先頭に戻す
- rmdir — ディレクトリを削除する
- set_file_buffer — stream_set_write_buffer のエイリアス
- stat — ファイルに関する情報を取得する
- symlink — シンボリックリンクを作成する
- tempnam — 一意なファイル名を生成する
- tmpfile — テンポラリファイルを作成する
- touch — ファイルの最終アクセス時刻および最終更新日をセットする
- umask — 現在の umask を変更する
- unlink — ファイルを削除する

フィルタ関数

導入

この拡張モジュールは、 ユーザ入力などのセキュアでないデータの検証およびフィルタリングを行います。

現在存在するのは、以下のフィルタです。各定数の振る舞いについての説明は フィルタ定数 の節を参照ください。

存在するフィルタ

ID	名前	オプション	フラグ	説明
FILTER_VALIDATE_INT	"int"	*min_range*, *max_range*	**FILTER_FLAG_ALLOW_OCTAL**, **FILTER_FLAG_ALLOW_HEX**	値が整数であるかどうか、オプションで指定した範囲内にあるかどうかを検証します。
FILTER_VALIDATE_BOOLEAN	"boolean"		**FILTER_NULL_ON_FAILURE**	"1"、"true"、"on" および "yes" の場合に **TRUE**、 それ以外の場合に **FALSE** を返します。 **FILTER_NULL_ON_FAILURE** が設定されている場合は、 **FALSE** が返されるのは "0"、"false"、"off"、"no" および "" の場合のみとなります。 boolean 以外の値については **NULL** を返します。
FILTER_VALIDATE_FLOAT	"float"	*decimal*	**FILTER_FLAG_ALLOW_THOUSAND**	値が float であるかどうかを検証します。
FILTER_VALIDATE_REGEXP	"validate_regexp"	*regexp*		値が、Perl 互換の 正規表現 *regexp* に一致するかどうかを検証します。
FILTER_VALIDATE_URL	"validate_url"		**FILTER_FLAG_PATH_REQUIRED**, **FILTER_FLAG_QUERY_REQUIRED**	値が URL 形式であるかどうか、 オプションで指定したコンポーネントが含まれているかどうかを検証します。
FILTER_VALIDATE_EMAIL	"validate_email"			値が e-mail 形式であるかどうかを検証します。
FILTER_VALIDATE_IP	"validate_ip"		**FILTER_FLAG_IPV4**, **FILTER_FLAG_IPV6**, **FILTER_FLAG_NO_PRIV_RANGE**, **FILTER_FLAG_NO_RES_RANGE**	値が IP アドレスであるかどうかを検証します。 オプションで IPv4 あるいは IPv6 のみの指定、 プライベートアドレスや予約済みアドレスではないことの指定もできます。
FILTER_SANITIZE_STRING	"string"		**FILTER_FLAG_NO_ENCODE_QUOTES**, **FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_ENCODE_LOW**, **FILTER_FLAG_ENCODE_HIGH**, **FILTER_FLAG_ENCODE_AMP**	タグを取り除きます。オプションで、特殊文字を取り除いたりエンコードしたりします。
FILTER_SANITIZE_STRIPPED	"stripped"			"string" フィルタのエイリアス。
FILTER_SANITIZE_ENCODED	"encoded"		**FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_ENCODE_LOW**, **FILTER_FLAG_ENCODE_HIGH**	文字列を URL エンコードします。オプションで、 特殊文字を取り除いたりエンコードしたりします。
FILTER_SANITIZE_SPECIAL_CHARS	"special_chars"		**FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_ENCODE_HIGH**	*'"<>&* および ASCII 値が 32 未満の文字を HTML エスケープします。オプションで、 特殊文字を取り除いたりエンコードしたりします。
FILTER_UNSAFE_RAW	"unsafe_raw"		**FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_ENCODE_LOW**, **FILTER_FLAG_ENCODE_HIGH**, **FILTER_FLAG_ENCODE_AMP**	何もせず、オプションで特殊文字を取り除いたりエンコードしたりします。
FILTER_SANITIZE_EMAIL	"email"			英字、数字および *!#$%&'*+-/=?^_`{|}~@.[]* 以外のすべての文字を取り除きます。
FILTER_SANITIZE_URL	"url"			英字、数字および *$-_.+!*'(),{}|¥¥^~[]`<>#%";/?:@&=* 以外のすべての文字を取り除きます。
FILTER_SANITIZE_NUMBER_INT	"number_int"			数字、プラス記号、マイナス記号 以外のすべての文字を取り除きます。
FILTER_SANITIZE_NUMBER_FLOAT	"number_float"		**FILTER_FLAG_ALLOW_FRACTION**, **FILTER_FLAG_ALLOW_THOUSAND**, **FILTER_FLAG_ALLOW_SCIENTIFIC**	数字、*+-* および オプションで *.,eE* 以外のすべての文字を取り除きます。
FILTER_SANITIZE_MAGIC_QUOTES	"magic_quotes"			addslashes() を適用します。

ID	名前	オプション	フラグ	説明

要件

外部ライブラリを必要としません。

インストール手順

簡単なインストールメモ。コンソールで、ただ単に

```
$ pecl install filter
```

とタイプします。

実行時設定

php.ini の設定により動作が変化します。

フィルタ設定オプション

名前	デフォルト	変更の可否	変更履歴
filter.default	"unsafe_raw"	PHP_INI_PERDIR	filter <= 0.9.4 では PHP_INI_ALL。PHP 5.2.0 以降で使用可能です。
filter.default_flags	NULL	PHP_INI_PERDIR	filter <= 0.9.4 では PHP_INI_ALL。PHP 5.2.0 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

`filter.default` string

このフィルタで $_GET, $_POST, $_COOKIE および $_REQUEST のすべてのデータをフィルタリングします。 もとのデータへは filter_input() でアクセスが可能です。

デフォルトで使用するフィルタの名前を指定することができます。 既存のフィルタの一覧を参照し、フィルタ名を指定してください。

`filter.default_flags` integer

デフォルトのフラグ。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

`INPUT_POST` (integer)
POST 変数。
`INPUT_GET` (integer)
GET 変数。
`INPUT_COOKIE` (integer)
COOKIE 変数。
`INPUT_ENV` (integer)
ENV 変数。
`INPUT_SERVER` (integer)
SERVER 変数。
`INPUT_SESSION` (integer)
SESSION 変数 (まだ実装されていません)。
`INPUT_REQUEST` (integer)
REQUEST 変数 (まだ実装されていません)。
`FILTER_FLAG_NONE` (integer)
フィルタしない。
`FILTER_REQUIRE_SCALAR` (integer)
入力値としてスカラーを要求するために使用するフラグ。

FILTER_REQUIRE_ARRAY (integer)
入力として配列を要求します。
FILTER_FORCE_ARRAY (integer)
常に配列として返します。
FILTER_NULL_ON_FAILURE (integer)
失敗した場合に FALSE ではなく NULL を使用します。
FILTER_VALIDATE_INT (integer)
"int" フィルタの ID。
FILTER_VALIDATE_BOOLEAN (integer)
"boolean" フィルタの ID。
FILTER_VALIDATE_FLOAT (integer)
"float" フィルタの ID。
FILTER_VALIDATE_REGEXP (integer)
"validate_regexp" フィルタの ID。
FILTER_VALIDATE_URL (integer)
"validate_url" フィルタの ID。
FILTER_VALIDATE_EMAIL (integer)
"validate_email" フィルタの ID。
FILTER_VALIDATE_IP (integer)
"validate_ip" フィルタの ID。
FILTER_DEFAULT (integer)
デフォルト ("string") フィルタの ID。
FILTER_UNSAFE_RAW (integer)
"unsafe_raw" フィルタの ID。
FILTER_SANITIZE_STRING (integer)
"string" フィルタの ID。
FILTER_SANITIZE_STRIPPED (integer)
"stripped" フィルタの ID。
FILTER_SANITIZE_ENCODED (integer)
"encoded" フィルタの ID。
FILTER_SANITIZE_SPECIAL_CHARS (integer)
"special_chars" フィルタの ID。
FILTER_SANITIZE_EMAIL (integer)
"email" フィルタの ID。
FILTER_SANITIZE_URL (integer)
"url" フィルタの ID。
FILTER_SANITIZE_NUMBER_INT (integer)
"number_int" フィルタの ID。
FILTER_SANITIZE_NUMBER_FLOAT (integer)
"number_float" フィルタの ID。
FILTER_SANITIZE_MAGIC_QUOTES (integer)
"magic_quotes" フィルタの ID。
FILTER_CALLBACK (integer)
"callback" フィルタの ID。
FILTER_FLAG_ALLOW_OCTAL (integer)
"int" フィルタで 8 進表記 (*0[0-7]+*) を許可します。
FILTER_FLAG_ALLOW_HEX (integer)
"int" フィルタで 16 進表記 (*0x[0-9a-fA-F]+*) を許可します。
FILTER_FLAG_STRIP_LOW (integer)
ASCII 値が 32 未満の文字を取り除きます。
FILTER_FLAG_STRIP_HIGH (integer)
ASCII 値が 127 より大きい文字を取り除きます。
FILTER_FLAG_ENCODE_LOW (integer)
ASCII 値が 32 未満の文字をエンコードします。
FILTER_FLAG_ENCODE_HIGH (integer)
ASCII 値が 127 より大きい文字をエンコードします。
FILTER_FLAG_ENCODE_AMP (integer)
& をエンコードします。
FILTER_FLAG_NO_ENCODE_QUOTES (integer)
' および *"* をエンコードしません。
FILTER_FLAG_EMPTY_STRING_NULL (integer)
(現在は使用されていません)
FILTER_FLAG_ALLOW_FRACTION (integer)
"number_float" フィルタで小数を許可します。

FILTER_FLAG_ALLOW_THOUSAND (integer)
"number_float" フィルタで桁区切り文字 (,) を許可します。
FILTER_FLAG_ALLOW_SCIENTIFIC (integer)
"number_float" フィルタで科学記法 (*e*, *E*) を許可します。
FILTER_FLAG_SCHEME_REQUIRED (integer)
"validate_url" フィルタでスキームを必須とします。
FILTER_FLAG_HOST_REQUIRED (integer)
"validate_url" フィルタでホスト名を必須とします。
FILTER_FLAG_PATH_REQUIRED (integer)
"validate_url" フィルタでパスを必須とします。
FILTER_FLAG_QUERY_REQUIRED (integer)
"validate_url" フィルタでクエリ文字列を必須とします。
FILTER_FLAG_IPV4 (integer)
"validate_ip" フィルタで IPv4 アドレスのみを許可します。
FILTER_FLAG_IPV6 (integer)
"validate_ip" フィルタで IPv6 アドレスのみを許可します。
FILTER_FLAG_NO_RES_RANGE (integer)
"validate_ip" フィルタで予約済みアドレスを拒否します。
FILTER_FLAG_NO_PRIV_RANGE (integer)
"validate_ip" フィルタでプライベートアドレスを拒否します。

---

filter_has_var

(PHP 5 >= 5.2.0, PECL filter:0.11.0)

filter_has_var — 指定した型の変数が存在するかどうかを調べる

説明

bool **filter_has_var** (int `$type` , string `$variable_name`)

パラメータ

`type`

INPUT_GET、**INPUT_POST**、 **INPUT_COOKIE**、**INPUT_SERVER**、 **INPUT_ENV** のいずれか。

`variable_name`

調べたい変数の名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

filter_id

(PHP 5 >= 5.2.0, PECL filter:0.11.0)

filter_id — フィルタの名前からフィルタ ID を返す

説明

int **filter_id** (string `$filtername`)

パラメータ

`filtername`

取得したいフィルタの名前。

返り値

フィルタの ID を返します。フィルタが存在しない場合は **NULL** を返します。

参考

- filter_list()

---

filter_input_array

(PHP 5 >= 5.2.0, PECL filter:0.11.0)

filter_input_array — PHP の外部から複数の変数を受け取り、オプションでそれらをフィルタリングする

説明

mixed **filter_input_array** (int $type [, mixed $definition])

この関数を使用すると、大量のデータを取得する際に filter_input() を繰り返しコールする必要がなくなるので便利です。

パラメータ

type

INPUT_GET、**INPUT_POST**、**INPUT_COOKIE**、**INPUT_SERVER**、**INPUT_ENV**、**INPUT_SESSION** あるいは **INPUT_REQUEST** のいずれか。

definition

引数を定義する配列。配列のキーとして使用できるのは 変数名を string で表したものです。 対応する値に使用できるのは、 フィルタの型か配列 (フィルタ・フラグ・オプションを指定したもの) です。 配列の値として配列を使用する場合に使用できるキーは、 *filter* (フィルタの型)、 *flags* (フィルタに適用するフラグ) および *options* (フィルタに適用するオプション) です。理解を深めるために、以下の例を参照ください。

このパラメータには、フィルタ定数 を表す整数値を指定することもできます。 こうすると、入力配列のすべての値がそのフィルタで処理されます。

返り値

成功した場合は要求された変数の値を含む配列、 あるいは失敗した場合に **FALSE** を返します。 配列の値は、フィルタリングに失敗した場合には **FALSE**、 変数が設定されていない場合は **NULL** となります。 フラグ **FILTER_NULL_ON_FAILURE** が指定されている場合は、変数が設定されていないときに **FALSE**、 フィルタリングに失敗した場合に **NULL** となります。

例

Example#1 filter_input_array() の例

```
<?php
error_reporting(E_ALL | E_STRICT);
/* データは、実際には POST リクエストでやってきます
$_POST = array(
      'product_id'      => 'libgd<script>',
     'component'       => '10',
     'versions'        => '2.0.33',
      'testscalar'      => array('2', '23', '10', '12'),
     'testarray'       => '2',
);
*/

$args = array(
    'product_id'    => FILTER_SANITIZE_ENCODED,
    'component'     => array('filter'     => FILTER_VALIDATE_INT,
                             'flags'      => FILTER_REQUIRE_ARRAY,
                             'options'    => array('min_range' => 1, 'max_range' => 10)
                            ),
    'versions'      => FILTER_SANITIZE_ENCODED,
    'doesnotexist'  => FILTER_VALIDATE_INT,
    'testscalar'    => array(
                             'filter' => FILTER_VALIDATE_INT,
                             'flags'  => FILTER_REQUIRE_SCALAR,
                            ),
    'testarray'     => array(
                             'filter' => FILTER_VALIDATE_INT,
                             'flags'  => FILTER_REQUIRE_ARRAY,
                            )

);

$myinputs = filter_input_array(INPUT_POST, $args);

var_dump($myinputs);
echo "¥n";
?>
```

上の例の出力は以下となります。

```
array(6) {
   ["product_id"]=>
   array(1) {
     [0]=>
      string(17) "libgd%3Cscript%3E"
  }
   ["component"]=>
   array(1) {
     [0]=>
     int(10)
  }
   ["versions"]=>
   array(1) {
     [0]=>
      string(6) "2.0.33"
  }
   ["doesnotexist"]=>
   NULL
   ["testscalar"]=>
   bool(false)
   ["testarray"]=>
   array(1) {
     [0]=>
     int(2)
  }
}
```

参考

- filter_input()
- filter_var_array()

---

filter_input

(PHP 5 >= 5.2.0, PECL filter:0.11.0)

filter_input — PHP の外部から変数を受け取り、オプションでそれをフィルタリングする

説明

mixed **filter_input** (int `$type` , string `$variable_name` [, int `$filter` [, mixed `$options`]])

パラメータ

type

INPUT_GET、**INPUT_POST**、 **INPUT_COOKIE**、**INPUT_SERVER**、 **INPUT_ENV**、**INPUT_SESSION** (まだ実装されていません) および **INPUT_REQUEST** (まだ実装されていません) のいずれか。

variable_name

取得する変数の名前。

filter

適用するフィルタ。デフォルトは **FILTER_DEFAULT** です。

options

オプションあるいはフラグの論理和の連想配列。 オプションを指定可能なフィルタの場合、この配列の "flags" フィールドにフラグを指定します。

返り値

成功した場合は要求された変数の値、フィルタリングに失敗した場合に **FALSE**、 あるいは変数 *variable_name* が設定されていない場合に **NULL** を返します。フラグ **FILTER_NULL_ON_FAILURE** が指定されている場合は、変数が設定されていなければ **FALSE**、 フィルタリングに失敗したら **NULL** を返します。

例

Example#1 filter_input() の例

```
<?php
$search_html  = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_SPECIAL_CHARS);
$search_url  = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_ENCODED);
echo "You have searched for $search_html.¥n";
```

```
echo "<a  href='?search=$search_url'>Search  again.</a>";
?>
```

上の例の出力は、たとえば以下のようになります。

```
You  have  searched  for  Me  &#38;  son.
<a  href='?search=Me%20%26%20son'>Search  again.</a>
```

参考

- filter_var()
- filter_input_array()
- filter_var_array()

---

filter_list

(PHP 5 >= 5.2.0, PECL filter:0.11.0)

filter_list — サポートされるフィルタの一覧を返す

説明

array **filter_list** (void)

返り値

サポートされる全フィルタの名前の配列を返します。 フィルタが存在しない場合は空の配列を返します。 この配列のインデックスはフィルタの ID ではありません。 ID を取得するには filter_id() にフィルタ名を渡します。

例

Example#1 filter_list() の例

```
<?php
print_r(filter_list());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => int
     [1] => boolean
     [2] => float
      [3] => validate_regexp
      [4] => validate_url
      [5] => validate_email
      [6] => validate_ip
     [7] => string
     [8] => stripped
     [9] => encoded
      [10] => special_chars
      [11] => unsafe_raw
     [12] => email
     [13] => url
      [14] => number_int
      [15] => number_float
      [16] => magic_quotes
     [17] => callback
)
```

---

filter_var_array

(PHP 5 >= 5.2.0, PECL filter:0.11.0)

filter_var_array — 複数の変数を受け取り、オプションでそれらをフィルタリングする

説明

mixed **filter_var_array** (array `$data` [, mixed `$definition`])

この関数を使用すると、大量の変数を取得する際に filter_var() を繰り返しコールする必要がなくなるので便利です。

パラメータ

data

文字列キーの配列。フィルタリングするデータを保持します。

definition

引数を定義する配列。配列のキーとして使用できるのは 変数名を string で表したものです。 対応する値に使用できるのは、 フィルタの型か配列 (フィルタ・フラグ・オプションを指定したもの) です。 配列の値として配列を使用する場合に使用できるキーは、 *filter* (フィルタの型)、 *flags* (フィルタに適用するフラグ) および *options* (フィルタに適用するオプション) です。理解を深めるために、以下の例を参照ください。

このパラメータには、フィルタ定数 を表す整数値を指定することもできます。 こうすると、入力配列のすべての値がそのフィルタで処理されます。

返り値

成功した場合は要求された変数の値を含む配列、 あるいは失敗した場合に **FALSE** を返します。 配列の値は、フィルタリングに失敗した場合には **FALSE**、 変数が設定されていない場合は **NULL** となります。

例

Example#1 filter_var_array() の例

```
<?php
error_reporting(E_ALL | E_STRICT);
$data = array(
    'product_id'     => 'libgd<script>',
    'component'      => '10',
    'versions'       => '2.0.33',
    'testscalar'     => array('2', '23', '10', '12'),
    'testarray'      => '2',
);

$args = array(
    'product_id'    => FILTER_SANITIZE_ENCODED,
    'component'     => array('filter'     => FILTER_VALIDATE_INT,
                             'flags'      => FILTER_FORCE_ARRAY,
                             'options'    => array('min_range' => 1, 'max_range' => 10)
                            ),
    'versions'      => FILTER_SANITIZE_ENCODED,
    'doesnotexist'  => FILTER_VALIDATE_INT,
    'testscalar'    => array(
                             'filter'  => FILTER_VALIDATE_INT,
                             'flags'   => FILTER_REQUIRE_SCALAR,
                            ),
    'testarray'     => array(
                             'filter'  => FILTER_VALIDATE_INT,
                             'flags'   => FILTER_FORCE_ARRAY,
                            )

);

$myinputs = filter_var_array($data, $args);

var_dump($myinputs);
echo "¥n";
?>
```

上の例の出力は以下となります。

```
array(6) {
   ["product_id"]=>
   array(1) {
     [0]=>
      string(17) "libgd%3Cscript%3E"
  }
   ["component"]=>
   array(1) {
     [0]=>
     int(10)
  }
   ["versions"]=>
   array(1) {
     [0]=>
      string(6) "2.0.33"
  }
   ["doesnotexist"]=>
   NULL
   ["testscalar"]=>
   bool(false)
   ["testarray"]=>
   array(1) {
     [0]=>
     int(2)
  }
}
```

参考

- filter_input_array()
- filter_var()
- filter_input()

---

filter_var

(PHP 5 >= 5.2.0, PECL filter:0.11.0)

filter_var — 指定したフィルタでデータをフィルタリングする

説明

mixed **filter_var** (mixed `$variable` [, int `$filter` [, mixed `$options`]])

パラメータ

`variable`

フィルタリングする値。

`filter`

使用するフィルタの ID。 デフォルトは **`FILTER_SANITIZE_STRING`** です。

`options`

オプションあるいはフラグの論理和の連想配列。 オプションを指定可能なフィルタの場合、この配列の "flags" フィールドにフラグを指定します。 "callback" フィルタの場合は、callback 型を渡さなければなりません。

返り値

フィルタリングされたデータ、あるいは処理に失敗した場合に **`FALSE`** を返します。

例

Example#1 filter_var() の例

```
<?php
var_dump(filter_var('bob@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('example.com', FILTER_VALIDATE_URL, FILTER_FLAG_SCHEME_REQUIRED));
?>
```

上の例の出力は以下となります。

```
string(15)  "bob@example.com"
bool(false)
```

参考

- filter_var_array()
- filter_input()
- filter_input_array()
- callback 型に関する情報

目次

- filter_var — 指定したフィルタでデータをフィルタリングする

Firebird/InterBase 関数

導入

Firebird/InterBase は ANSI SQL-92 の多くの機能をサポートする リレーショナルデータベースで、Linux・Windows その他多くの Unix プラットフォーム上で動作します。Firebird/InterBase は 優れた同時実行性・パフォーマンス、そしてストアドプロシージャや トリガでの強力な言語サポートを提供します。1981 年以降、このデータベースは さまざまな名前の製品として使用されています。

InterBase はこの RDBMS のクローズドソース版の名前で、Borland/Inprise によって開発されました。InterBase についての詳細な情報は » http://www.borland.com/interbase/ にあります。

Firebird は C・C++プログラムやテクニカルアドバイザやサポーターたちによる 商業的に独立したプロジェクトです。Inprise Corp（現在は Borland Software Corp という名で知られています）が 2000 年 7 月 25 日に InterBase Public License v.1.0 の下で公開したソースコードをもとにして、マルチプラットフォームなリレーショナルデータベースの開発と機能拡張を 行っています。Firebird についての詳細な情報は » http://www.firebirdsql.org/ にあります。

> **注意**: この拡張モジュールは、InterBase のバージョン 5 以降とすべてのバージョンの Firebird をサポートします。InterBase バージョン 5.x のサポートは、PHP 5 で削除されました。
> このデータベースは、シングルクオート(')文字をエスケープに使用します。 この動作は Sybase データベースに似ており、以下のディレクティブを *php.ini* に追加してください。

```
magic_quotes_sybase  =  On
```

要件

インストール手順

PHP で InterBase サポートを有効にするには、 --with-interbase[=DIR] を指定して 設定を行います。ただし、DIR は InterBase のベースインストールディレクトリで、 デフォルトは */usr/interbase* です。

> **注意**: **Win32 ユーザへの注意** この拡張モジュールを動作させるには、 Windows システムの *PATH* が通った場所に DLL ファイルが存在する必要があります。 FAQ の "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" で、その方法を説明しています。 DLL ファイルを PHP のフォルダから Windows のシステムディレクトリにコピーしても動作します (システムディレクトリは、デフォルトで *PATH* に含まれるからです) が、これは推奨しません。 *この拡張モジュールを使用するには、以下のファイルが PATH の通った場所にある必要があります。 gds32.dll*
> InterBase データベースサーバを PHP と同じマシンにインストールしている場合は、 すでに DLL が存在するはずです。その場合は特に何も考える必要はありません。 というのも *gds32.dll* はすでに *PATH* の通った場所に存在するからです。

実行時設定

php.ini の設定により動作が変化します。

InterBase設定オプション

名前	デフォルト	変更の可否	変更履歴
ibase.allow_persistent	"1"	PHP_INI_SYSTEM	
ibase.max_persistent	"-1"	PHP_INI_SYSTEM	
ibase.max_links	"-1"	PHP_INI_SYSTEM	
ibase.default_db	NULL	PHP_INI_SYSTEM	PHP 5.0.0 以降で使用可能です。
ibase.default_user	NULL	PHP_INI_ALL	
ibase.default_password	NULL	PHP_INI_ALL	
ibase.default_charset	NULL	PHP_INI_ALL	PHP 5.0.0 以降で使用可能です。
ibase.timestampformat	"%Y-%m-%d %H:%M:%S"	PHP_INI_ALL	
ibase.dateformat	"%Y-%m-%d"	PHP_INI_ALL	
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

ibase.allow_persistent boolean

Firebird/InterBase 持続的な接続 を許可するかどうか。

ibase.max_persistent integer

プロセスごとの、Firebird/InterBase の持続的接続の最大数。 この数を超えた場合、ibase_pconnect() による新しい接続は 持続的ではない接続になります。

ibase.max_links integer

プロセスごとの Firebird/InterBase の接続の最大数。持続的な接続の数も 含みます。

ibase.default_db string

データベース名を指定せずに ibase_[p]connect() がコールされた場合に 接続するデフォルトのデータベース。この値が設定されており、かつ SQL セーフモードが有効な場合、このデータベース以外への接続は一切許可されません。

ibase.default_user string

ユーザ名を指定せずにデータベースに接続した際に使用されるユーザ名。

ibase.default_password string

パスワードを指定せずにデータベースに接続した際に使用されるパスワード。

ibase.default_charset string

文字セットを指定せずにデータベースに接続した際に使用される文字セット。

ibase.timestampformat string

ibase.dateformat string

ibase.timeformat string

これらのディレクティブは日付や時間のフォーマットを指定します。 結果セットの中の日付や時間のデータ、そしてパラメータとしてバインドする 日付や時間のデータに適用されます。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

トランザクションの挙動を設定するため、以下の定数を ibase_trans() に渡すことが可能です。

Firebird/InterBase トランザクションフラグ

定数	説明
IBASE_DEFAULT	デフォルトのトランザクション設定が使用されます。デフォルト設定は クライアントライブラリによって決定され、たいていの場合は IBASE_WRITE¦IBASE_CONCURRENCY¦IBASE_WAIT です。
IBASE_READ	読み込み専用のトランザクションを開始します。
IBASE_WRITE	読み書き可能なトランザクションを開始します。
IBASE_CONSISTENCY	分離レベルを 'consistency' にしてトランザクションを開始します。 他のトランザクションによって変更中のテーブルを読み込むことは できません。
IBASE_CONCURRENCY	分離レベルを 'concurrency' (あるいは 'snapshot') にして トランザクションを開始します。すべてのテーブルへのアクセスが 可能ですが、トランザクションの開始以降に別のトランザクションに よってコミットされた内容を見ることはできません。
IBASE_COMMITTED	分離レベルを 'read committed' にしてトランザクションを開始します。 このフラグは **IBASE_REC_VERSION** あるいは **IBASE_REC_NO_VERSION** とともに使用する必要があります。 トランザクションの開始以降にコミットされた内容についてもアクセス可能と なります。**IBASE_REC_NO_VERSION** が指定された場合、 最新バージョンのデータのみが読み込み可能となります。 **IBASE_REC_VERSION** が指定された場合、他の トランザクションで処理が完了していない変更についても読み込むことが できます。
IBASE_WAIT	衝突が発生した場合に、トランザクションが処理を再試行することを示します。
IBASE_NOWAIT	衝突が発生した場合に、トランザクションがその場で処理を中断して 処理が失敗することを示します。

データの取得方法を指定するために、以下の定数を ibase_fetch_row()、 ibase_fetch_assoc() あるいは ibase_fetch_object() に渡すことが可能で

す。

Firebird/InterBase フェッチフラグ

定数	説明
IBASE_FETCH_BLOBS	過去との互換性を保持するため、**IBASE_TEXT** としても使用可能です。BLOB データを読み込む際に、BLOB ID ではなく直接データそのものを読み込みます。
IBASE_FETCH_ARRAYS	配列をインラインで読み込みます。指定しなかった場合は、配列の ID を返します。配列 ID は INSERT 操作への引数としてのみ使用可能で、 配列 ID を処理できる関数は現在存在しません。
IBASE_UNIXTIME	日付や時刻のフィールド値を、文字列ではなく UNIX タイムスタンプ (1-Jan-1970 0:00 UTC からの経過秒数)で返します。 1970 年より前の日付に対してこれを使用すると、環境によっては 問題を引き起こす可能性があります。

要求内容やオプションを指定するために、以下の定数を サービス API 関数(ibase_server_info(), ibase_db_info(), ibase_backup(), ibase_restore() および ibase_maintain_db())に渡すことが可能です。 これらのオプションの意味については Firebird/InterBase のマニュアルを 参照ください。

IBASE_BKP_IGNORE_CHECKSUMS
IBASE_BKP_IGNORE_LIMBO
IBASE_BKP_METADATA_ONLY
IBASE_BKP_NO_GARBAGE_COLLECT
IBASE_BKP_OLD_DESCRIPTIONS
IBASE_BKP_NON_TRANSPORTABLE
IBASE_BKP_CONVERT
ibase_backup() へのオプション
IBASE_RES_DEACTIVATE_IDX
IBASE_RES_NO_SHADOW
IBASE_RES_NO_VALIDITY
IBASE_RES_ONE_AT_A_TIME
IBASE_RES_REPLACE
IBASE_RES_CREATE
IBASE_RES_USE_ALL_SPACE
ibase_restore() へのオプション
IBASE_PRP_PAGE_BUFFERS
IBASE_PRP_SWEEP_INTERVAL
IBASE_PRP_SHUTDOWN_DB
IBASE_PRP_DENY_NEW_TRANSACTIONS
IBASE_PRP_DENY_NEW_ATTACHMENTS
IBASE_PRP_RESERVE_SPACE
IBASE_PRP_RES_USE_FULL
IBASE_PRP_RES
IBASE_PRP_WRITE_MODE
IBASE_PRP_WM_ASYNC
IBASE_PRP_WM_SYNC
IBASE_PRP_ACCESS_MODE
IBASE_PRP_AM_READONLY
IBASE_PRP_AM_READWRITE
IBASE_PRP_SET_SQL_DIALECT
IBASE_PRP_ACTIVATE
IBASE_PRP_DB_ONLINE
IBASE_RPR_CHECK_DB
IBASE_RPR_IGNORE_CHECKSUM
IBASE_RPR_KILL_SHADOWS
IBASE_RPR_MEND_DB
IBASE_RPR_VALIDATE_DB
IBASE_RPR_FULL
IBASE_RPR_SWEEP_DB
ibase_maintain_db() へのオプション
IBASE_STS_DATA_PAGES
IBASE_STS_DB_LOG
IBASE_STS_HDR_PAGES
IBASE_STS_IDX_PAGES
IBASE_STS_SYS_RELATIONS
ibase_db_info() へのオプション
IBASE_SVC_SERVER_VERSION
IBASE_SVC_IMPLEMENTATION
IBASE_SVC_GET_ENV

IBASE_SVC_GET_ENV_LOCK
IBASE_SVC_GET_ENV_MSG
IBASE_SVC_USER_DBPATH
IBASE_SVC_SVR_DB_INFO
IBASE_SVC_GET_USERS
ibase_server_info() へのオプション

---

ibase_add_user

(PHP 4 >= 4.2.0, PHP 5)

ibase_add_user — セキュリティデータベースにユーザを追加する(IB6 以降のみ)

説明

bool **ibase_add_user** (resource `$service_handle` , string `$user_name` , string `$password` [, string `$first_name` [, string `$middle_name` [, string `$last_name`]]])

PHP 4 では、*service_handle* パラメータの かわりに *server* 、 *dba_user_name* および *dba_user_password* を使用します。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_modify_user()
- ibase_delete_user()

---

ibase_affected_rows

(PHP 5)

ibase_affected_rows — 直近のクエリで変更された行の数を返す

説明

int **ibase_affected_rows** ([resource `$link_identifier`])

この関数は、*link_identifier* で指定された トランザクション内で実行された直近のクエリ(INSERT、UPDATE あるいは DELETE)によって変更された行の数を返します。

パラメータ

link_identifier

トランザクションのコンテキスト。 *link_identifier* が接続リソースの場合、 そのデフォルトのトランザクションが使用されます。

返り値

変更された行数を整数値で返します。

参考

- ibase_query()
- ibase_execute()

---

ibase_backup

(PHP 5)

ibase_backup — サービスマネージャのバックアップタスクを起動し、すぐに結果を返す

説明

mixed **ibase_backup** (resource `$service_handle` , string `$source_db` , string `$dest_file` [, int `$options` [, bool `$verbose`]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ibase_blob_add

(PHP 4, PHP 5)

ibase_blob_add — 生成された blob にデータを追加する

説明

void **ibase_blob_add** (resource `$blob_handle` , string `$data`)

ibase_blob_add() は、 ibase_blob_create() で作成した blob にデータを追加します。

パラメータ

`blob_handle`

ibase_blob_create() でオープンした blob ハンドル。

`data`

追加するデータ。

返り値

値を返しません。

参考

- ibase_blob_cancel()
- ibase_blob_close()
- ibase_blob_create()
- ibase_blob_import()

---

ibase_blob_cancel

(PHP 4, PHP 5)

ibase_blob_cancel — blob の生成を取り消す

説明

bool **ibase_blob_cancel** (resource `$blob_handle`)

この関数は、まだ ibase_blob_close() で閉じられていない場合に BLOB を捨てます。

パラメータ

`blob_handle`

ibase_blob_create() でオープンした blob ハンドル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_blob_close()
- ibase_blob_create()
- ibase_blob_import()

---

ibase_blob_close

(PHP 4, PHP 5)

ibase_blob_close — blob を閉じる

説明

mixed **ibase_blob_close** (resource `$blob_handle`)

この関数は、読み込み用に **ibase_open_blob()** でオープンされたかあるいは書き込み用に **ibase_create_blob()** でオープンされた BLOB を閉じます。

パラメータ

`blob_handle`

ibase_blob_create() あるいは **ibase_open_blob()** でオープンした blob ハンドル。

返り値

BLOB が読み込み用であった場合、 この関数は成功時に **TRUE** を返します。一方 BLOB が書き込み用であった場合、 データベースによって割り当てられた BLOB ID を文字列で返します。 失敗した場合は、この関数は **FALSE** を返します。

参考

- ibase_blob_cancel()
- ibase_blob_open()

---

ibase_blob_create

(PHP 4, PHP 5)

ibase_blob_create — データを追加するために blob を生成する

説明

resource **ibase_blob_create** ([resource `$link_identifier`])

ibase_blob_create() は、 データを書き込むための新しい BLOB を生成します。

パラメータ

`link_identifier`

InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

返り値

ibase_blob_add() で使用するための BLOB ハンドルを返します。失敗した場合は **FALSE** を返します。

参考

- ibase_blob_add()
- ibase_blob_cancel()
- ibase_blob_close()
- ibase_blob_import()

---

ibase_blob_echo

(PHP 4, PHP 5)

ibase_blob_echo — ブラウザに blob の内容を出力する

説明

bool **ibase_blob_echo** ([resource `$link_identifier`], string `$blob_id`)

この関数は読み込み用に BLOB をオープンし、直接その内容を標準出力 (たいていの場合はブラウザ) に送信します。

パラメータ

`link_identifier`

InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

`blob_id`

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_blob_open()
- ibase_blob_close()
- ibase_blob_get()

---

ibase_blob_get

(PHP 4, PHP 5)

ibase_blob_get — オープンした blob から len バイト分のデータを取得する

説明

string **ibase_blob_get** (resource `$blob_handle` , int `$len`)

この関数は、ibase_blob_open() によって読み込み用に オープンした BLOB から最大 *`len`* バイトを返します。

注意: ibase_blob_create() で書き込み用にオープンした BLOB から読み込むことはできません。

パラメータ

`blob_handle`

ibase_blob_open() でオープンした BLOB ハンドル。

`len`

返されるデータのサイズ。

返り値

BLOB から最大 *`len`* バイトを返します。 失敗した場合は **FALSE** を返します。

例

Example#1 ibase_blob_get() の例

```
<?php
$result     = ibase_query("SELECT  blob_value  FROM  table");
$data       = ibase_fetch_object($result);
$blob_data  = ibase_blob_info($data->BLOB_VALUE);
$blob_hndl  = ibase_blob_open($data->BLOB_VALUE);
echo          ibase_blob_get($blob_hndl, $blob_data[0]);
?>
```

この例では 'ibase_blob_echo($data->BLOB_VALUE)' が行う以上のことは 行っていませんが、結果を変数に代入して操作する方法を示しています。

参考

- ibase_blob_open()

- ibase_blob_close()
- ibase_blob_echo()

---

ibase_blob_import

(PHP 4, PHP 5)

ibase_blob_import — blob を生成し、ファイルをコピーし、閉じる

説明

string **ibase_blob_import** (resource `$link_identifier` , resource `$file_handle`)
string **ibase_blob_import** (resource `$file_handle`)

この関数は BLOB を作成し、その中にファイルのすべての内容を読み込み、 それを閉じ、関連付けられた BLOB ID を返します。

パラメータ

`link_identifier`

InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

`file_handle`

fopen() が返すファイルハンドル。

返り値

成功した場合に BLOB の ID、エラー時に **FALSE** を返します。

例

Example#1 ibase_blob_import() の例

```
<?php
$dbh = ibase_connect($host, $username, $password);
$filename = '/tmp/bar';

$fd = fopen($filename, 'r');
if ($fd) {

    $blob = ibase_blob_import($dbh, $fd);
    fclose($fd);

    if (!is_string($blob)) {
        // インポートに失敗しました
    } else {
        $query = "INSERT INTO foo (name, data) VALUES ('$filename', ?)";
        $prepared = ibase_prepare($dbh, $query);
        if (!ibase_execute($prepared, $blob)) {
            // レコードの挿入に失敗しました
        }
    }
} else {
    // データファイルをオープンできませんでした
}
?>
```

参考

- ibase_blob_add()
- ibase_blob_cancel()
- ibase_blob_close()
- ibase_blob_create()

---

ibase_blob_info

(PHP 4, PHP 5)

ibase_blob_info — blob の長さと他の便利な情報を返す

説明

array **ibase_blob_info** (resource `$link_identifier` , string `$blob_id`)
array **ibase_blob_info** (string `$blob_id`)

BLOB の長さや他の便利な情報を返します。

パラメータ

`link_identifier`

InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

`blob_id`

BLOB の ID。

返り値

BLOB についての情報を含む配列を返します。返される情報には BLOB の長さ・含んでいるセグメントの数・最大のセグメントのサイズ・ ストリーム BLOB とセグメント BLOB のどちらであるかなどがあります。

---

ibase_blob_open

(PHP 4, PHP 5)

ibase_blob_open — データの一部を取得するために blob をオープンする

説明

resource **ibase_blob_open** (resource `$link_identifier` , string `$blob_id`)
resource **ibase_blob_open** (string `$blob_id`)

既存の BLOB を読み込み用にオープンします。

パラメータ

`link_identifier`

InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

`blob_id`

BLOB の ID。

返り値

後で ibase_blob_get() とともに使用する BLOB ハンドルを返します。失敗した場合には **`FALSE`** を返します。

参考

- ibase_blob_close()
- ibase_blob_echo()
- ibase_blob_get()

---

ibase_close

(PHP 4, PHP 5)

ibase_close — InterBase データベースへの接続を閉じる

説明

bool **ibase_close** ([resource `$connection_id`])

ibase_connect() から返された接続 ID が指す InterBase データベースへのリンクを閉じます。 接続 ID が省略された場合、最後にオープンされたリンクが仮定されます。 リンクにおけるデフォルトのトランザクションがコミットされ、 他のトランザクションはロールバックされます。

パラメータ

connection_id

ibase_connect() が返す InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_connect()
- ibase_pconnect()

---

ibase_commit_ret

(PHP 5)

ibase_commit_ret — トランザクションを閉じずにコミットする

説明

bool **ibase_commit_ret** ([resource `$link_or_trans_identifier`])

トランザクションを閉じずにコミットします。

パラメータ

link_or_trans_identifier

引数なしでコールされた場合、この関数はデフォルトリンクのデフォルトの トランザクションをコミットします。引数が接続 ID だった場合、対応する 接続のデフォルトのトランザクションをコミットします。引数が トランザクション ID だった場合、対応するトランザクションをコミットします。 トランザクションコンテキストはそのまま維持され、このトランザクション内で 実行された文は取り消されません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ibase_commit

(PHP 4, PHP 5)

ibase_commit — トランザクションをコミットする

説明

bool **ibase_commit** ([resource `$link_or_trans_identifier`])

トランザクションをコミットします。

パラメータ

link_or_trans_identifier

引数なしでコールされた場合、この関数はデフォルトリンクの デフォルトのトランザクションをコミットします。引数が接続 ID であった場合は、対応する接続のデフォルトのトランザクションを コミットします。引数がトランザクション ID であった場合は、 対応するトランザクションがコミットされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ibase_connect

(PHP 4, PHP 5)

ibase_connect — InterBase データベースへの接続をオープンする

説明

resource **ibase_connect** ([string $database [, string $username [, string $password [, string $charset [, int $buffers [, int $dialect [, string $role [, int $sync]]]]]]])

InterBaseサーバへの接続を確立します。

同じ引数で **ibase_connect()** が再度コールされた場合、 新規のリンクは作成されず、代わりに既にオープンされているリンクの リンク ID が返されます。 ibase_close() を明示的にコールしない限り、 サーバへのリンクはスクリプトの実行終了時に閉じられます。

パラメータ

database

database は接続するサーバ上のデータベース ファイルへの正しいパスである必要があります。ローカルなサーバへの 接続でない場合、使用する接続プロトコルに応じてこの引数の前に' hostname:' (TCP/IP)、'//hostname/' (NetBEUI)、'hostname@' (IPX/SPX)のどれかをつける必要があります。

username

ユーザ名。*php.ini* ディレクティブ *ibase.default_user* で設定します。

password

username のパスワード。 *php.ini* ディレクティブ *ibase.default_password* で設定します。

charset

charset はデータベースに関するデフォルトの文字セットです。

buffers

buffers はサーバ側のキャッシュに確保されるデータベースバッファの数です。0 または省略された場合、サーバーはデフォルト値を用います。

dialect

dialect は、接続時に実行される全ての命令に 関する SQL 方言のデフォルト値を選択し、デフォルトではクライアント ライブラリでサポートされる方言のうち、最高位のものになります。 InterBase 6 以降でのみ有効です。

role

InterBase 5 以降でのみ有効です。

sync

返り値

成功した場合に InterBase リンク ID、エラー時に **FALSE** を返します。

エラー / 例外

この関数を使用して ibase_query() をコールした後に "arithmetic exception, numeric overflow, or string truncation. Cannot transliterate character between character sets" のようなエラーが発生した場合 (たとえばアクセント記号つきの文字を使用した場合などに発生します)、 文字セットを指定する必要があります (例: ISO8859_1 あるいは現在の文字セット)。

変更履歴

バージョン	説明
4.0.0	*buffers* 、*dialect* および *role* が追加されました。

例

Example#1 ibase_connect() の例

```
<?php
$host = 'localhost:/path/to/your.gdb';

$dbh = ibase_connect($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$sth = ibase_query($dbh, $stmt);
while ($row = ibase_fetch_object($sth)) {
    echo $row->email, "¥n";
}
```

```
ibase_free_result($sth);
ibase_close($dbh);
?>
```

参考

- ibase_pconnect()
- ibase_close()

---

ibase_db_info

(PHP 5)

ibase_db_info — データベースについての統計情報を要求する

説明

string **ibase_db_info** (resource $service_handle , string $db , int $action [, int $argument])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ibase_delete_user

(PHP 4 >= 4.2.0, PHP 5)

ibase_delete_user — セキュリティデータベースからユーザを削除する(IB6 以降のみ)

説明

bool **ibase_delete_user** (resource $service_handle , string $user_name)

PHP 4 では、*service_handle* パラメータの かわりに *server* 、 *dba_user_name* および *dba_user_password* を使用します。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_add_user()
- ibase_modify_user()

---

ibase_drop_db

(PHP 5)

ibase_drop_db — データベースを削除する

説明

bool **ibase_drop_db** ([resource $connection])

この関数は、ibase_connect() あるいは ibase_pconnect() のいずれかによってオープンされたデータベースを削除します。 データベースがクローズされ、サーバから削除されます。

パラメータ

connection

InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_connect()
- ibase_pconnect()

---

ibase_errcode

(PHP 5)

ibase_errcode — エラーコードを返す

説明

int **ibase_errcode** (void)

直近の InterBase 関数のコールによって得られたエラーコードを返します。

返り値

エラーコードを表す整数、あるいはエラーが発生しなかった場合は **FALSE** を返します。

参考

- ibase_errmsg()

---

ibase_errmsg

(PHP 4, PHP 5)

ibase_errmsg — エラーメッセージを返す

説明

string **ibase_errmsg** (void)

直近の InterBase 関数コールの結果として得られるエラーメッセージを返します。

返り値

エラーメッセージを表す文字列、あるいはエラーが発生しなかった場合は **FALSE** を返します。

参考

- ibase_errcode()

---

ibase_execute

(PHP 4, PHP 5)

ibase_execute — 準備されたクエリを実行する

説明

resource **ibase_execute** (resource `$query` [, mixed `$bind_arg` [, mixed `$...`]])

ibase_prepare() で準備したクエリを実行します。

いくつかのパラメータが変わるだけで同じクエリを 複数回実行する場合には、この関数は ibase_query() を使用するよりもずっと効率的です。

パラメータ

query

ibase_prepare() で準備した InterBase クエリ。

bind_arg

...

返り値

クエリがエラーを発生させた場合は **FALSE** を返します。 クエリが成功し、結果セット(空のセットとなる可能性もありえます)を 返した場合(SELECT クエリなど)は結果 ID を返します。クエリが成功し、 結果セットが返されなかった場合は **TRUE** を返します。

> **注意**: PHP 5.0.0 以降では、この関数はクエリによって変更された行の数を 返します(行を変更するクエリで、結果が 0 より大きかった場合)。 クエリは成功したが、どの行も変更されなかった場合(例: 存在しない 行に対する UPDATE)は **TRUE** を返します。

例

Example#1 ibase_execute() の例

```
<?php

$dbh = ibase_connect($host, $username, $password);

$updates = array(
    1 => 'Eric',
    5 => 'Filip',
    7 => 'Larry'
);

$query = ibase_prepare($dbh, "UPDATE FOO SET BAR = ? WHERE BAZ = ?");

foreach ($updates as $baz => $bar) {
    ibase_execute($query, $bar, $baz);
}

?>
```

参考

- ibase_query()

---

ibase_fetch_assoc

(PHP 4 >= 4.3.0, PHP 5)

ibase_fetch_assoc — クエリの結果から、行を連想配列として取得する

説明

array **ibase_fetch_assoc** (resource `$result` [, int `$fetch_flag`])

クエリの結果の行を連想配列で返します。

ibase_fetch_assoc() は、 *result* から結果を 1 行取得します。 同じフィールド名のカラムが 2 つ以上存在する場合、最後のカラムが 優先されます。同名のその他のカラムにアクセスするには、 ibase_fetch_row() を使用して数値添字を用いるか あるいはクエリ中でカラムに別名をつけます。

パラメータ

result

結果ハンドル。

fetch_flag

fetch_flag は、定数 **IBASE_TEXT** および **IBASE_UNIXTIME** を論理和で指定します。**IBASE_TEXT** を渡すと、BLOB ID のかわりに BLOB の内容自体を返します。**IBASE_UNIXTIME** を渡すと、日付/時刻の値を文字列ではなく Unix タイムスタンプで返します。

返り値

ibase_fetch_assoc() は、取得した行に対応する連想配列を返します。 続けてコールすると、結果セットの次の行を返し、 行がもうない場合には **FALSE** を返します。

参考

- ibase_fetch_row()
- ibase_fetch_object()

---

ibase_fetch_object

(PHP 4, PHP 5)

ibase_fetch_object — InterBase データベースからオブジェクトを得る

説明

object **ibase_fetch_object** (resource $result_id [, int $fetch_flag])

指定した結果 ID から、行を疑似オブジェクトとして取得します。

ibase_fetch_object() を続けてコールすると、 結果セットの次の行を返します。

パラメータ

result_id

ibase_query() あるいは ibase_execute() で取得した InterBase 結果 ID。

fetch_flag

fetch_flag は、定数 **IBASE_TEXT** および **IBASE_UNIXTIME** を論理和で指定します。**IBASE_TEXT** を渡すと、BLOB ID のかわりに BLOB の内容自体を返します。**IBASE_UNIXTIME** を渡すと、日付/時刻の値を文字列ではなく Unix タイムスタンプで返します。

返り値

次の行の情報を含むオブジェクト、 あるいは行がもうない場合には **FALSE** を返します。

例

Example#1 ibase_fetch_object() の例

```
<?php
$dbh = ibase_connect($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$sth = ibase_query($dbh, $stmt);
while ($row = ibase_fetch_object($sth)) {
    echo $row->email . "¥n";
}
ibase_close($dbh);
?>
```

参考

- ibase_fetch_row()
- ibase_fetch_assoc()

---

ibase_fetch_row

(PHP 4, PHP 5)

ibase_fetch_row — InterBase データベースから 1 行分の結果を取得する

説明

array **ibase_fetch_row** (resource $result_identifier [, int $fetch_flag])

ibase_fetch_row() は、指定した *result_identifier* に関連付けられた結果からデータを 1 行取得します。

ibase_fetch_row() を続けてコールすると、 結果セットの次の行を返します。行がもうない場合には **FALSE** を返します。

パラメータ

result_identifier

InterBase 結果 ID。

fetch_flag

fetch_flag は、定数 **IBASE_TEXT** および **IBASE_UNIXTIME** を論理和で指定します。**IBASE_TEXT** を渡すと、BLOB ID のかわりに BLOB の内容自体を返します。**IBASE_UNIXTIME** を渡すと、日付/時刻の値を文字列ではなく Unix タイムスタンプで返します。

返り値

取得した行に対応する配列を返します。行がもうない場合には **FALSE** を返します。 結果の各行は配列のオフセットに格納され、 このオフセットは 0 から始まります。

参考

- ibase_fetch_assoc()
- ibase_fetch_object()

---

ibase_field_info

(PHP 4, PHP 5)

ibase_field_info — フィールドに関する情報を得る

説明

array **ibase_field_info** (resource `$result` , int `$field_number`)

select クエリが実行された後、フィールドに関する情報を配列として返します。

パラメータ

result

InterBase 結果 ID。

field_number

フィールドのオフセット。

返り値

配列を返します。配列に含まれるキーは *name*、 *alias*、*relation*、 *length* そして *type* です。

例

Example#1 ibase_field_info() の例

```
<?php
$rs = ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i = 0; $i < $coln; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "name: ". $col_info['name']. "¥n";
    echo "alias: ". $col_info['alias']. "¥n";
    echo "relation: ". $col_info['relation']. "¥n";
    echo "length: ". $col_info['length']. "¥n";
    echo "type: ". $col_info['type']. "¥n";
}
?>
```

参考

- ibase_num_fields()

---

ibase_free_event_handler

(PHP 5)

ibase_free_event_handler — 登録済みのイベントハンドラをキャンセルする

説明

bool **ibase_free_event_handler** (resource $event)

この関数は、*event* で指定した登録済みの イベントハンドラをキャンセルします。イベントに対応して登録されていた コールバック関数は、もはや コールされません。

パラメータ

event

ibase_set_event_handler() で作成したイベントリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_set_event_handler()

---

ibase_free_query

(PHP 4, PHP 5)

ibase_free_query — プリペアドクエリにより確保されたメモリを解放する

説明

bool **ibase_free_query** (resource $query)

プリペアドクエリを解放します。

パラメータ

query

ibase_prepare() で準備したクエリ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ibase_free_result

(PHP 4, PHP 5)

ibase_free_result — 結果セットを解放する

説明

bool **ibase_free_result** (resource $result_identifier)

結果セットを解放します。

パラメータ

result_identifier

ibase_query() あるいは ibase_execute() で作成した結果セット。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ibase_gen_id

(PHP 5)

ibase_gen_id — 指定した名前のジェネレータをひとつ加算し、その新しい値を返す

説明

mixed **ibase_gen_id** (string `$generator` [, int `$increment` [, resource `$link_identifier`]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

新しいジェネレータの値を整数で返します。値が大きくなりすぎた場合は文字列で返します。

---

ibase_maintain_db

(PHP 5)

ibase_maintain_db — データベースサーバでメンテナンスコマンドを実行する

説明

bool **ibase_maintain_db** (resource `$service_handle` , string `$db` , int `$action` [, int `$argument`])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ibase_modify_user

(PHP 4 >= 4.2.0, PHP 5)

ibase_modify_user — セキュリティデータベースのユーザを変更する(IB6 以降のみ)

説明

bool **ibase_modify_user** (resource `$service_handle` , string `$user_name` , string `$password` [, string `$first_name` [, string `$middle_name` [, string `$last_name`]]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

PHP 4 では、*service_handle* パラメータのかわりに *server* 、 *dba_user_name* および *dba_user_password* を使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ibase_add_user()
- ibase_delete_user()

---

ibase_name_result

(PHP 5)

ibase_name_result — 結果セットに名前を割り当てる

説明

bool **ibase_name_result** (resource $result , string $name)

この関数は、結果セットに新しい名前を割り当てます。この名前は、後で UPDATE¦DELETE ... WHERE CURRENT OF *name* 文で使用します。

パラメータ

result

InterBase 結果セット。

name

割り当てる名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ibase_name_result() の例

```
<?php
$result = ibase_query("SELECT field1,field2 FROM table FOR UPDATE");
ibase_name_result($result, "my_cursor");

$updateqry = ibase_prepare("UPDATE table SET field2 = ? WHERE CURRENT OF my_cursor");

for ($i = 0; ibase_fetch_row($result); ++$i) {
    ibase_execute($updateqry, $i);
}
?>
```

参考

- ibase_prepare()
- ibase_execute()

---

ibase_num_fields

(PHP 4, PHP 5)

ibase_num_fields — 結果セットにおけるフィールド数を得る

説明

int **ibase_num_fields** (resource $result_id)

結果セットにおけるフィールド数を整数として返します。

パラメータ

result_id

InterBase 結果 ID。

返り値

フィールドの数を整数値で返します。

例

Example#1 ibase_num_fields() の例

```
<?php
$rs = ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i = 0; $i < $coln; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "name: " . $col_info['name'] . "¥n";
    echo "alias: " . $col_info['alias'] . "¥n";
    echo "relation: " . $col_info['relation'] . "¥n";
    echo "length: " . $col_info['length'] . "¥n";
    echo "type: " . $col_info['type'] . "¥n";
}
?>
```

参考

- ibase_field_info()

---

ibase_num_params

(PHP 5)

ibase_num_params — プリペアドクエリのパラメータ数を返す

説明

int **ibase_num_params** (resource $query)

この関数は、*query* で指定した プリペアドクエリのパラメータの数を返します。 ibase_execute() をコールする際に、 この数と同じだけのバインド引数が割り当てられている必要があります。

パラメータ

query

プリペアドクエリのハンドル。

返り値

パラメータの数を整数値で返します。

参考

- ibase_prepare()
- ibase_param_info()

---

ibase_param_info

(PHP 5)

ibase_param_info — プリペアドクエリのパラメータに関する情報を返す

説明

array **ibase_param_info** (resource $query , int $param_number)

クエリが準備された後に、パラメータについての情報を配列で返します。

パラメータ

query

InterBase プリペアドクエリのハンドル。

param_number

パラメータのオフセット。

返り値

配列を返します。配列に含まれるキーは *name*、 *alias*、*relation*、 *length* および *type* となります。

参考

- ibase_field_info()
- ibase_num_params()

---

ibase_pconnect

(PHP 4, PHP 5)

ibase_pconnect — InterBase データベースへの持続的接続をオープンする

説明

resource **ibase_pconnect** ([string `$database` [, string `$username` [, string `$password` [, string `$charset` [, int `$buffers` [, int `$dialect` [, string `$role` [, int `$sync`]]]]]]])

InterBase データベースへの持続的な接続をオープンします。

ibase_pconnect() の動作は ibase_connect() と非常に似ていますが、 大きな違いが二つあります。

まず、この関数は接続時に同じパラメータで既にオープンされている (持続的)リンクを探します。見つかった場合、新規接続をオープンする 代わりにそのリンクの ID が返されます。

2 番目の違いとしては、InterBase サーバへの接続は スクリプト終了時にも閉じられないということです。 代わりに、そのリンクは今後使用するためにオープンされたままとなります (ibase_close() は **ibase_pconnect()** によりオープンされたリンクを閉じません)。 このため、この型のリンクは'持続的(persistent)'と呼ばれます。

パラメータ

`database`

`database` は接続するサーバ上のデータベース ファイルへの正しいパスである必要があります。ローカルなサーバへの 接続でない場合、使用する接続プロトコルに応じてこの引数の前に' hostname:' (TCP/IP)、'//hostname/' (NetBEUI)、'hostname@' (IPX/SPX)のどれかをつける必要があります。

`username`

ユーザ名。*php.ini* ディレクティブ *ibase.default_user* で設定します。

`password`

`username` のパスワード。 *php.ini* ディレクティブ *ibase.default_password* で設定します。

`charset`

`charset` はデータベースに関するデフォルトの文字セットです。

`buffers`

`buffers` はサーバ側のキャッシュに確保されるデータベースバッファの数です。0 または省略された場合、サーバーはデフォルト値を用います。

`dialect`

`dialect` は、接続時に実行される全ての命令に 関する SQL 方言のデフォルト値を選択し、デフォルトではクライアント ライブラリでサポートされる方言のうち、最高位のものになります。 InterBase 6 以降でのみ有効です。

`role`

InterBase 5 以降でのみ有効です。

`sync`

返り値

成功した場合に InterBase リンク ID、エラー時に **FALSE** を返します。

変更履歴

バージョン	説明
4.0.0	*`buffers`* 、*`dialect`* および *`role`* が追加されました。

参考

- ibase_close()
- ibase_connect()

---

ibase_prepare

(PHP 4, PHP 5)

ibase_prepare — 後でパラメータのバインド及び実行を行うためにクエリを準備する

説明

resource **ibase_prepare** (string `$query`)
resource **ibase_prepare** (resource `$link_identifier` , string `$query`)
resource **ibase_prepare** (resource `$link_identifier` , string `$trans` , string `$query`)

対応するパラメータのバインドや (ibase_execute() による) 実行をするためのクエリを準備します。

パラメータ

query

InterBase クエリ。

返り値

プリペアドクエリのハンドル、あるいはエラー時に **FALSE** を返します。

---

ibase_query

(PHP 4, PHP 5)

ibase_query — InterBase データベースでクエリを実行する

説明

resource **ibase_query** ([resource `$link_identifier`], string `$query` [, int `$bind_args`])

InterBase データベース上でクエリを実行します。

パラメータ

link_identifier

InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

query

InterBase クエリ。

bind_args

返り値

クエリが失敗した場合、**FALSE** を返します。成功した場合、 (SELECT クエリのような) 結果の行がある場合、結果 ID を返します。 クエリが成功し、結果がない場合は **TRUE** を返します。

> **注意**: PHP 5.0.0 以降では、INSERT・UPDATE・DELETE 文に対してはこの関数は 変更された行数を返します。後方互換性を確保するため、これらの文で クエリが成功したものの 1 行も更新されなかった場合には **TRUE** を返します。

エラー / 例外

ibase_query() の実行後に "arithmetic exception, numeric overflow, or string truncation. Cannot transliterate character between character sets" のようなエラーに遭遇した場合 (アクセント記号つきの文字を使用した場合などに起こります)、 文字セットを(ISO8859_1 あるいは現在の文字セットに)設定する必要があります。

例

Example#1 ibase_query() の例

```
<?php

$host = 'localhost:/path/to/your.gdb';

$dbh = ibase_connect($host, $username, $password);
$stmt = 'SELECT * FROM tblname';

$sth = ibase_query($dbh, $stmt) or die(ibase_errmsg());
```

```
?>
```

参考

- ibase_errmsg()
- ibase_fetch_row()
- ibase_fetch_object()
- ibase_free_result()

---

ibase_restore

(PHP 5)

ibase_restore — サービスマネージャのリストアタスクを起動し、すぐに結果を返す

説明

mixed **ibase_restore** (resource `$service_handle` , string `$source_file` , string `$dest_db` [, int `$options` [, bool `$verbose`]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ibase_rollback_ret

(PHP 5)

ibase_rollback_ret — トランザクションを閉じずにロールバックする

説明

bool **ibase_rollback_ret** ([resource `$link_or_trans_identifier`])

トランザクションを閉じずにロールバックします。

パラメータ

`link_or_trans_identifier`

引数なしでコールされた場合、この関数はデフォルトリンクのデフォルトの トランザクションをロールバックします。引数が接続 ID だった場合、対応する 接続のデフォルトのトランザクションをロールバックします。引数が トランザクション ID だった場合、対応するトランザクションをロールバックします。 トランザクションコンテキストはそのまま維持され、このトランザクション内で 実行された文は取り消されません。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

---

ibase_rollback

(PHP 4, PHP 5)

ibase_rollback — トランザクションをロールバックする

説明

bool **ibase_rollback** ([resource `$link_or_trans_identifier`])

トランザクションをロールバックします。

パラメータ

`link_or_trans_identifier`

引数なしでコールされた場合、この関数はデフォルトリンクのデフォルトの トランザクションをロールバックします。引数として接続 ID が渡さ

れた場合、 対応する接続のデフォルトのトランザクションをロールバックします。 引数としてトランザクション ID が渡された場合、対応するトランザクションを ロールバックします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ibase_server_info

(PHP 5)

ibase_server_info — データベースサーバについての情報を要求する

説明

string **ibase_server_info** (resource `$service_handle` , int `$action`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ibase_service_attach

(PHP 5)

ibase_service_attach — サービスマネージャに接続する

説明

resource **ibase_service_attach** (string `$host` , string `$dba_username` , string `$dba_password`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ibase_service_detach

(PHP 5)

ibase_service_detach — サービスマネージャとの接続を切断する

説明

bool **ibase_service_detach** (resource `$service_handle`)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ibase_set_event_handler

(PHP 5)

ibase_set_event_handler — イベントが発生した際にコールされるコールバック関数を登録する

説明

resource **ibase_set_event_handler** (callback `$event_handler` , string `$event_name1` [, string `$event_name2` [, string `$...`]])
resource **ibase_set_event_handler** (resource `$connection` , callback `$event_handler` , string `$event_name1` [, string `$event_name2` [, string `$...`]])

この関数は、指定したイベントのハンドラとして PHP ユーザ関数を登録します。

パラメータ

event_handler

コールバックは、データベースから特定のイベントが送信された場合に、 イベント名とリンクリソースを引数としてコールされます。

イベントハンドラがキャンセルされた場合は、 コールバックは **FALSE** を返す必要があります。 その他の返り値は無視されます。 この関数は、最大 15 のイベントを引数として受け取ります。

event_name1

イベントの名前。

event_name2

...

返り値

返される値はイベントのリソースです。このリソースは、 ibase_free_event_handler() でイベントハンドラを開放する際に使用可能です。

例

Example#1 ibase_set_event_handler() の例

```
<?php

function event_handler($event_name, $link)
{
    if ($event_name == "NEW ORDER") {
       // 新しい注文を処理します
        ibase_query($link, "UPDATE orders SET status='handled'");
    } else if ($event_name == "DB_SHUTDOWN") {
       // イベントハンドラを解放します
        return false;
    }
}

ibase_set_event_handler($link, "event_handler", "NEW_ORDER", "DB_SHUTDOWN");
?>
```

参考

- ibase_free_event_handler()
- ibase_wait_event()

---

ibase_timefmt

(PHP 4)

ibase_timefmt — クエリから返される timestamp、data、time 型カラムのフォーマットを設定する

説明

bool **ibase_timefmt** (string `$format` [, int `$columntype`])

クエリから返される timestamp、data、time 型カラムのフォーマットを設定します。

PHP 設定ディレクティブ *ibase.timestampformat*、 *ibase.dateformat* および *ibase.timeformat* によりこれらのフォーマットのデフォルト値を設定することが可能です。

注意: この関数は PHP 5 で削除されました。かわりに ini_set() を使用してください。

パラメータ

format

内部的にこれらのカラムは C 言語の関数 strftime() により フォーマットされます。このため、文字列のフォーマットについては この C 言語の関数ドキュメントを参照ください。

columntype

columntype は **IBASE_TIMESTAMP**、**IBASE_DATE** および **IBASE_TIME** のいずれかとなります。 省略した場合は、下位互換性を保つために **IBASE_TIMESTAMP** を使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ibase_timefmt() の例

```
<?php
/* InterBase 6 TIME型カラムが
 * '05 hours 37 minutes'の形式で返されます */
ibase_timefmt("%H hours %M minutes", IBASE_TIME);
?>
```

---

ibase_trans

(PHP 4, PHP 5)

ibase_trans — トランザクションを開始する

説明

resource **ibase_trans** ([int $trans_args [, resource $link_identifier]])

トランザクションを開始します。

> **注意**: この関数の挙動は PHP 5.0.0 で変更されました。The first call **ibase_trans()** を最初にコールした際には、接続の デフォルトのトランザクションを返しません。**ibase_trans()** によって開始されたすべてのトランザクションは、ibase_commit() や ibase_rollback() を使用してコミットあるいは ロールバックを明示的に行わない限り、スクリプトの終了時に自動的に ロールバックされます。
>
> **注意**: PHP 5.0.0 以降では、この関数は複数の *trans_args* および *link_identifier* を指定することが可能です。これにより、 複数のデータベース接続にまたがるトランザクションを扱えるようになり、 2 フェーズコミット機能を使用したコミットが可能になります。つまり、複数の データベースの更新内容が成功したか失敗したかによる判断ができるということです。 これは、ひとつのクエリで異なるデータベースのテーブルを同時に使用できるという ことではありません!
> 複数データベースにまたがるトランザクションを使用する場合、 ibase_query() および ibase_prepare() をコールする際には *link_id* および *transaction_id* の両方を指定する必要があります。

パラメータ

trans_args

> *trans_args* は、以下の **IBASE_READ**、 **IBASE_WRITE**、 **IBASE_COMMITTED**、 **IBASE_CONSISTENCY**、 **IBASE_CONCURRENCY**、 **IBASE_REC_VERSION**、 **IBASE_REC_NO_VERSION**、 **IBASE_WAIT** および **IBASE_NOWAIT** の組み合わせとなります。

link_identifier

> InterBase リンク ID。省略した場合は、 最後にオープンしたリンクを使用します。

返り値

トランザクションハンドル、あるいはエラー時に **FALSE** を返します。

---

ibase_wait_event

(PHP 5)

ibase_wait_event — データベースでイベントが発生するのを待つ

説明

string **ibase_wait_event** (string $event_name1 [, string $event_name2 [, string $...]])
string **ibase_wait_event** (resource $connection , string $event_name1 [, string $event_name2 [, string $...]])

この関数は、指定したイベントのうちのひとつがデータベースで発生するまで スクリプトの実行を停止します。発生したイベントの名前を返します。
この関数は、最大 15 のイベントを指定可能です。

パラメータ

event_name1

イベントの名前。

event_name2

...

返り値

発生したイベントの名前を返します。

参考

- ibase_set_event_handler()
- ibase_free_event_handler()

目次

- ibase_service_detach — サービスマネージャとの接続を切断する
- ibase_set_event_handler — イベントが発生した際にコールされるコールバック関数を登録する
- ibase_timefmt — クエリから返される timestamp、data、time 型カラムのフォーマットを設定する
- ibase_trans — トランザクションを開始する
- ibase_wait_event — データベースでイベントが発生するのを待つ

Firebird/Interbase 関数 (PDO_FIREBIRD)

導入

警告

この拡張モジュールは、 *実験的* なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

PDO_FIREBIRD は、PHP から Firebird および Interbase データベースへのアクセスを可能にするための PHP Data Objects (PDO) インターフェース を実装したドライバです。

PDO_FIREBIRD DSN

(No version information available, might be only in CVS)

PDO_FIREBIRD DSN — Firebird および Interbase データベースに接続する

説明

PDO_FIREBIRD データソース名 (DSN) は以下の要素で構成されます。

DSN 接頭辞

DSN 接頭辞は **`firebird:`** です。

`DataSource`

データベースサーバが存在するホスト名を指定します。

`Port`

データベースサーバが起動しているポートを指定します。

`Database`

データベース名を指定します。

`User`

データベースに接続するユーザ名を指定します。

`Password`

ユーザのパスワードを指定します。

例

Example#1 PDO_FIREBIRD DSN の例

以下の例は、Firebird および Interbase データベースに接続するための PDO_FIREBIRD DSN を表します。

```
firebird:User=john;Password=mypass;Database=DATABASE.GDE;DataSource=localhost;Port=3050
```

FriBiDi 関数

導入

FriBiDiは、» Unicode Bidirectional Algorithm のフリーの実装です。

要件

» FriBiDiパッケージ を ダウンロードし、インストールする必要があります。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/fribidi.

これらの関数を使用するには、 --with-fribidi[=DIR] を指定し、 Fribidi のサポートを有効にして PHP をコンパイルしてください。

Windows ユーザは、*php.ini* の内部で *php_fribidi.dll* を有効にすることで、これらの関数を使用可能です。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

FRIBIDI_CHARSET_UTF8 (integer)
Unicode
FRIBIDI_CHARSET_8859_6 (integer)
アラビア語
FRIBIDI_CHARSET_8859_8 (integer)
ヘブライ語
FRIBIDI_CHARSET_CP1255 (integer)
ヘブライ語/イディッシュ語
FRIBIDI_CHARSET_CP1256 (integer)
アラビア語
FRIBIDI_CHARSET_ISIRI_3342 (integer)
ペルシア語
FRIBIDI_CHARSET_CAP_RTL (integer)
テスト用。CAPS を非英文字として扱います
FRIBIDI_RTL (integer)
右から左へ
FRIBIDI_LTR (integer)
左から右へ
FRIBIDI_AUTO (integer)
方向の自動検出

---

fribidi_log2vis

(PHP 4 >= 4.0.4, PECL fribidi:1.0)

fribidi_log2vis — 論理表記を物理表記に変換する

説明

string **fribidi_log2vis** (string `$str` , string `$direction` , int `$charset`)

論理表記を物理表記に変換します。

パラメータ

`str`

論理表記の文字列。

direction

FRIBIDI_RTL、 **FRIBIDI_LTR** あるいは **FRIBIDI_AUTO** のいずれかひとつ。

charset

FRIBIDI_CHARSET_XXX のいずれかひとつ。

返り値

成功した場合に物理表記の文字列、失敗した場合に **FALSE** を返します。

FrontBase 関数

導入

このモジュールの関数により、FrontBase データベースサーバに アクセスすることが可能になります。FrontBase に関する詳細については、» http://www.frontbase.com/ で入手可能です。

FrontBase に関するドキュメントは、» http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation から入手可能です。

Frontbase のサポートは、PHP 4.0.6 で追加されました。

要件

この関数を使用するには、FrontBase データベースサーバまたは少なくとも fbsql クライアントライブラリをインストールする必要があります。 » http://www.frontbase.com/ から FrontBase を 取得することができます。

インストール手順

これらの関数を使用するには、オプション --with-fbsql[=DIR] を使用し、fbsql のサポートを有効にして PHP をコンパイルする必要があります。fbsql へのパスを指定せずにこのオプションを 使用した場合、PHP は fbsql クライアントライブラリをそのプラットフォームの デフォルトのインストール位置で探します。FrontBase を標準以外の 場所にインストールしているユーザは、必ず次のように fbsql へのパスを 指定する必要があります。 --with-fbsql=/path/to/fbsql これにより、PHP は、間違いなく FrontBase によりインストールされた クライアントライブラリを探すことができるようになります。

実行時設定

php.ini の設定により動作が変化します。

FrontBase 設定オプション

名前	デフォルト	変更の可否	変更履歴
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM	PHP 4.2.0 以降で使用可能です。
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.autocommit	"1"	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.max_links	"128"	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.max_connections	"128"	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.max_results	"128"	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.batchSize	"1000"	PHP_INI_SYSTEM	PHP 4.2.0 以降で使用可能です。PHP 5.1.0 で削除されました。
fbsql.default_host	NULL	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.default_password	""	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.default_database	""	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。
fbsql.default_database_password	""	PHP_INI_SYSTEM	PHP 4.0.6 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

FBSQL_ASSOC (integer)
FBSQL_NUM (integer)
FBSQL_BOTH (integer)
FBSQL_LOCK_DEFERRED (integer)
FBSQL_LOCK_OPTIMISTIC (integer)
FBSQL_LOCK_PESSIMISTIC (integer)
FBSQL_ISO_READ_UNCOMMITTED (integer)
FBSQL_ISO_READ_COMMITTED (integer)
FBSQL_ISO_REPEATABLE_READ (integer)
FBSQL_ISO_SERIALIZABLE (integer)
FBSQL_ISO_VERSIONED (integer)
FBSQL_UNKNOWN (integer)
FBSQL_STOPPED (integer)
FBSQL_STARTING (integer)
FBSQL_RUNNING (integer)
FBSQL_STOPPING (integer)
FBSQL_NOEXEC (integer)
FBSQL_LOB_DIRECT (integer)
FBSQL_LOB_HANDLE (integer)

---

fbsql_affected_rows

(PHP 4 >= 4.0.6, PHP 5)

fbsql_affected_rows — 直近の FrontBase 操作により変更されたレコードの数を得る

説明

int **fbsql_affected_rows** ([resource `$link_identifier`])

fbsql_affected_rows() は、 *`link_identifier`* が指す接続において 直近の INSERT、UPDATE、DELETE クエリで変更されたレコードの数を返します。

> **注意**: トランザクションを使用している場合、コミットの後ではなくINSERT、 UPDATE、DELETE クエリの後で **fbsql_affected_rows()** をコールする必要があります。

直近のクエリが WHERE 句のない DELETE クエリの場合、全てのレコードが テーブルから削除されますが、この関数はゼロを返します。

> **注意**: UPDATE を使用する場合、FrontBase は新しい値が古い値と同じ場合には カラムを更新しません。このため、**fbsql_affected_rows()** は、実際にはマッチした 行の数と一致しない可能性があり、クエリにより実際に変更された行の 数だけとなります。

パラメータ

`link_identifier`

> fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。
>
> 指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

直近のクエリが失敗した場合、この関数は -1 を返します。

参考

- fbsql_num_rows()

---

fbsql_autocommit

(PHP 4 >= 4.0.6, PHP 5)

fbsql_autocommit — autocommit を有効または無効にする

説明

bool **fbsql_autocommit** (resource `$link_identifier` [, bool `$OnOff`])

autocommit の状態を返します。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

`OnOff`

このオプションのパラメータを指定すると、 オートコミットステータスは変更されます。

`OnOff` を **`TRUE`** に設定すると、 エラーがない場合に各命令が自動的にコミットされます。

OnOff を **`FALSE`** に設定すると、ユーザは fbsql_commit() あるいは fbsql_rollback() によりコミットまたはロールバックを行う必要があります。

返り値

現在の autocommit の状態を boolean 値で返します。

参考

- fbsql_commit()
- fbsql_rollback()

---

fbsql_blob_size

(PHP 4 >= 4.2.0, PHP 5)

fbsql_blob_size — BLOB の大きさを取得する

説明

int **fbsql_blob_size** (string `$blob_handle` [, resource `$link_identifier`])

指定した BLOB の大きさを返します。

パラメータ

`blob_handle`

fbsql_create_blob() が返す BLOB ハンドル。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

BLOB の大きさを整数値で返します。エラー時には **`FALSE`** を返します。

参考

- fbsql_clob_size()

---

fbsql_change_user

(PHP 4 >= 4.0.6, PHP 5)

fbsql_change_user — アクティブな接続にログインしているユーザを変更する

説明

bool **fbsql_change_user** (string `$user` , string `$password` [, string `$database` [, resource `$link_identifier`]])

fbsql_change_user() は、指定した接続にログインするユーザを変更します。 ユーザおよびパスワードの認証に失敗した場合は、現在の接続ユーザがアクティブなままとなります。

パラメータ

`user`

新しいユーザ名。

`password`

新しいユーザのパスワード。

`database`

指定した場合、これがユーザ変更後のデフォルトまたはカレントデータベースとなります。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

---

fbsql_clob_size

(PHP 4 >= 4.2.0, PHP 5)

fbsql_clob_size — CLOB の大きさを取得する

説明

int **fbsql_clob_size** (string `$clob_handle` [, resource `$link_identifier`])

指定した CLOB の大きさを取得します。

パラメータ

`clob_handle`

fbsql_create_clob() が返す CLOB ハンドル。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

CLOB の大きさを整数値で返します。エラー時には **`FALSE`** を返します。

参考

- fbsql_blob_size()

---

fbsql_close

(PHP 4 >= 4.0.6, PHP 5)

fbsql_close — FrontBase 接続を閉じる

説明

bool **fbsql_close** ([resource `$link_identifier`])

指定したリンク ID に関連する FrontBase サーバへの接続を閉じます。

持続的でないオープンされたリンクは、スクリプト実行終了時に自動的に クローズされるため、**fbsql_close()** は通常は不要です。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 fbsql_close() の例

```
<?php
$link = fbsql_connect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
echo "接続に成功しました";
fbsql_close($link);
?>
```

参考

- fbsql_connect()
- fbsql_pconnect()

---

fbsql_commit

(PHP 4 >= 4.0.6, PHP 5)

fbsql_commit — データベースへのトランザクションをコミットする

説明

bool **fbsql_commit** ([resource `$link_identifier`])

ディスクへの insert、update、 delete を全て書き込んでトランザクションを終了し、トランザクションにより 保持された全ての行及びテーブルのロックを解除します。このコマンドは、 autocommit が false に設定されている場合のみ必要です。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fbsql_autocommit()
- fbsql_rollback()

---

fbsql_connect

(PHP 4 >= 4.0.6, PHP 5)

fbsql_connect — FrontBase サーバへの接続をオープンする

説明

resource **fbsql_connect** ([string $hostname [, string $username [, string $password]]])

fbsql_connect() は、FrontBase サーバへの接続を確立します。

同じ引数で **fbsql_connect()** が 2 度目に コールされた場合、新規のリンクは確立されず、替わりに既に オープンされているリンク ID が返されます。

fbsql_close() をコールすることにより事前に 明示的にクローズされていない限り、サーバへのリンクはスクリプトの 実行終了時にクローズされます。

パラメータ

hostname

サーバのホスト名。デフォルトは '**NULL**'。

username

接続用のユーザ名。デフォルトは *_SYSTEM*。

password

接続用のパスワード。デフォルトは空文字列。

返り値

成功時に正の FrontBase リンク ID、エラー時に **FALSE** を返します。

例

Example#1 fbsql_connect() の例

```
<?php

$link = fbsql_connect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
echo "接続に成功しました";
fbsql_close($link);

?>
```

参考

- fbsql_pconnect()
- fbsql_close()

---

fbsql_create_blob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_create_blob — BLOB を作成する

説明

string **fbsql_create_blob** (string `$blob_data` [, resource `$link_identifier`])

指定したデータから BLOB を作成します。

パラメータ

`blob_data`

BLOB データ。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

新しく作成した BLOB のリソースハンドルを返します。 これを insert や update コマンドで使用すると、 データベースに BLOB を保存することができます。

例

Example#1 fbsql_create_blob() の例

```
<?php
$link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
$filename = "blobfile.bin";
$fp = fopen($filename, "rb");
$blobdata = fread($fp, filesize($filename));
fclose($fp);

$blobHandle = fbsql_create_blob($blobdata, $link);

$sql = "INSERT INTO BLOB_TABLE (BLOB_COLUMN) VALUES ($blobHandle);";
$rs = fbsql_query($sql, $link);
?>
```

参考

- fbsql_create_clob()
- fbsql_read_blob()
- fbsql_read_clob()
- fbsql_set_lob_mode()

---

fbsql_create_clob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_create_clob — CLOB を作成する

説明

string **fbsql_create_clob** (string `$clob_data` [, resource `$link_identifier`])

指定したデータから CLOB を作成します。

パラメータ

`clob_data`

CLOB データ。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

新しく作成した CLOB のリソースハンドルを返します。 これを insert や update コマンドで使用すると、 データベースに CLOB を保存することができます。

例

Example#1 fbsql_create_clob() の例

```
<?php
$link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
$filename = "clob_file.txt";
$fp = fopen($filename, "rb");
$clobdata = fread($fp, filesize($filename));
fclose($fp);

$clobHandle = fbsql_create_clob($clobdata, $link);

$sql = "INSERT INTO CLOB_TABLE (CLOB_COLUMN) VALUES ($clobHandle);";
$rs = fbsql_query($sql, $link);
?>
```

参考

- fbsql_create_blob()
- fbsql_read_blob()
- fbsql_read_clob()
- fbsql_set_lob_mode()

---

fbsql_create_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_create_db — FrontBase データベースを作成する

説明

bool **fbsql_create_db** (string `$database_name` [, resource `$link_identifier` [, string `$database_options`]])

指定したサーバ上で新しいデータベースの作成を試みます。

パラメータ

database_name

データベース名を表す文字列。

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

database_options

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 fbsql_create_db() の例

```
<?php
$link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
if (fbsql_create_db("my_db")) {
    echo "データベースの作成に成功しました\n";
} else {
    printf("データベース作成エラー:  %s\n", fbsql_error());
}
?>
```

参考

- fbsql_drop_db()

fbsql_data_seek

(PHP 4 >= 4.0.6, PHP 5)

fbsql_data_seek — 内部結果ポインタを移動する

説明

bool **fbsql_data_seek** (resource $result , int $row_number)

指定した結果 ID が指す FrontBase 結果の内部行ポインタを指定した行番号に移動します。

これ以降に fbsql_fetch_row() をコールすると、 その行が返されます。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

row_number

行番号。0 からはじまります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 fbsql_data_seek() の例

```
<?php
$link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
    or die("接続できません");

fbsql_select_db("samp_db")
    or die("データベースを選択できません");

$query = "SELECT last_name, first_name FROM friends;";
$result = fbsql_query($query)
    or die("クエリに失敗しました");

// 行を逆順に取得します

for ($i = fbsql_num_rows($result) - 1; $i >=0; $i--) {
    if (!fbsql_data_seek($result, $i)) {
        printf("行 %d\ に移動できませんn", $i);
          continue;
    }

    if (!($row = fbsql_fetch_object($result)))
         continue;

    echo $row->last_name . $row->first_name . "<br />¥n";
}

fbsql_free_result($result);
?>
```

fbsql_database_password

(PHP 4 >= 4.0.6, PHP 5)

fbsql_database_password — FrontBase データベースのパスワードを設定または取得する

説明

string **fbsql_database_password** (resource $link_identifier [, string $database_password])

現在の接続で 使用されているデータベースのパスワードを設定または取得します。 データベースがパスワードで保護されている場合は、
fbsql_select_db() の前にこの関数をコールする必要があります。

リンクがオープンされていない場合、この関数は fbsql_connect() がコールされた場合と同様にリンクを確立し、使用します。

この関数はデータベース内のデータベースパスワードを変更しませんし、 またデータベースのデータベースパスワードを取得することもしません。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

database_password

データベースのパスワードを表す文字列。 指定したリンク ID が指すサーバのデータベースパスワードを設定します。

返り値

リンク ID により表されるデータベースのデータベースパスワードを返します。

例

Example#1 fbsql_create_clob() の例

```
<?php
$link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
fbsql_database_password($link, "secret  db  password");
fbsql_select_db($database, $link);
?>
```

参考

- fbsql_connect()
- fbsql_pconnect()
- fbsql_select_db()

---

fbsql_database

(PHP 4 >= 4.0.6, PHP 5)

fbsql_database — 接続で使用するデータベース名を取得あるいは設定する

説明

string **fbsql_database** (resource `$link_identifier` [, string `$database`])

接続で使用するデータベース名を取得あるいは設定します。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

database

データベース名。指定すると、接続で使用するデフォルトのデータベースが *database* に変わります。

返り値

この接続で使用するデータベースの名前を返します。

---

fbsql_db_query

(PHP 4 >= 4.0.6, PHP 5)

fbsql_db_query — FrontBase クエリを送信する

説明

resource **fbsql_db_query** (string `$database` , string `$query` [, resource `$link_identifier`])

データベースを選択し、そこでクエリを実行します。

パラメータ

`database`

選択したデータベース。

`query`

実行する SQL クエリ。

注意: クエリ文字列の最後には、セミコロンをつけなければなりません。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

クエリ結果への正の FrontBase 結果 ID、またはエラー時には **`FALSE`** を返します。

参考

- fbsql_query()
- fbsql_connect()

---

fbsql_db_status

(PHP 4 >= 4.0.7, PHP 5)

fbsql_db_status — 指定したデータベースの状態を取得する

説明

int **fbsql_db_status** (string `$database_name` [, resource `$link_identifier`])

指定したデータベースの現在の状態を取得します。

パラメータ

`database_name`

データベース名。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

現在の状態を表す整数値を返します。これは、以下の定数のいずれかです。

- **`FALSE`** - host の exec ハンドラが無効です。このエラーは、 *`link_identifier`* がポート番号を用いてデータベースへ直接接続する場合に発生します。 FBExec はサーバで利用可能ですが、接続が行われていません。
- **`FBSQL_UNKNOWN`** - 状態は不明です。
- **`FBSQL_STOPPED`** - FBSQL_STOPPED - データベースは実行されていません。 データベースを開始するには、fbsql_start_db() を使用してください。

- **FBSQL_STARTING** - データベースは起動中です。
- **FBSQL_RUNNING** - データベースは実行中で、SQL 処理を実行可能です。
- **FBSQL_STOPPING** - データベースは停止中です。
- **FBSQL_NOEXEC** - FBExec がサーバで実行されておらず、 データベースの状態を取得することはできません。

参考

- fbsql_start_db()
- fbsql_stop_db()

---

fbsql_drop_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_drop_db — FrontBase データベースを破棄(削除)する

説明

bool **fbsql_drop_db** (string `$database_name` [, resource `$link_identifier`])

fbsql_drop_db() は、指定したリンク ID が指す サーバからデータベース全体を破棄(削除)します。

パラメータ

`database_name`

データベース名を表す文字列。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fbsql_create_db()

---

fbsql_errno

(PHP 4 >= 4.0.6, PHP 5)

fbsql_errno — 前の FrontBase 操作によるエラーメッセージの数値を返す

説明

int **fbsql_errno** ([resource `$link_identifier`])

直近の FrontBase 操作からのエラーメッセージを表す数値を返します。

fbsql データベースのバックエンドから返されるエラーは、警告を発生しません。 その代わりに、**fbsql_errno()** を使用してエラーコードを 取得します。この関数は、直近に実行した fbsql 関数 (fbsql_error() および **fbsql_errno()** は除く) のエラーコードのみを返すことに注意しましょう。 もしこれを使用するつもりなら、他の fbsql 関数をコールする前に値を調べる必要があります。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指

定せずにコールしたときと同様にして作成します。

返り値

直近の fbsql 関数のエラー番号、あるいはエラーが発生しなかった場合に *0* (ゼロ) を返します。

例

Example#1 fbsql_errno() の例

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
fbsql_select_db("nonexistentdb");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
?>
```

参考

- fbsql_error()
- fbsql_warnings()

---

fbsql_error

(PHP 4 >= 4.0.6, PHP 5)

fbsql_error — 前の FrontBase 操作によるエラーメッセージの文字列を返す

説明

string **fbsql_error** ([resource `$link_identifier`])

直近の FrontBase 操作のエラー文字列を返します。

fbsql データベースのバックエンドから返されるエラーは、警告を発生しません。 その代わりに、**fbsql_error()** を使用してエラー文字列を 取得します。この関数は、直近に実行した fbsql 関数 (**fbsql_error()** および fbsql_errno() は除く)のエラーコードのみを返すことに注意しましょう。もしこれを使用 するつもりなら、他の fbsql 関数をコールする前に値を調べる必要があります。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

直近の fbsql 関数のエラー文字列を返します。エラーが発生していない場合は '' (空の文字列) を返します。

例

Example#1 fbsql_error() の例

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
fbsql_select_db("nonexistentdb");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno() . ": " . fbsql_error() . "<br />";
?>
```

参考

- fbsql_errno()
- fbsql_warnings()

---

fbsql_fetch_array

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_array — 連想配列、数値配列、またはその両方として結果レコードを取得する

説明

array **fbsql_fetch_array** (resource `$result` [, int `$result_type`])

fbsql_fetch_array() は fbsql_fetch_row() と fbsql_fetch_assoc() を組み合わせたものです。

注意してほしいのは、 **fbsql_fetch_array()** は fbsql_fetch_row() と比べてそれほど遅くはないのに その追加機能がとても優れているということです。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

result_type

FBSQL_ASSOC、**FBSQL_NUM** あるいは **FBSQL_BOTH** のいずれかの定数値。

FBSQL_BOTH を使用すると、 数値添字の結果配列にデータを格納するだけでなく、フィールド名を キーとする連想配列にもデータを格納します。

返り値

取得した行に対応する配列を返します。行がもうない場合には **FALSE** を返します。

2 つ以上のカラムが同じ名前を持っている場合、最後にあらわれたカラムが 優先されます。他のカラムにアクセスするには、カラムの数値インデックスを 使用するか、カラムに別名をつける必要があります。

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

例

Example#1 fbsql_fetch_array() の例

```
<?php
fbsql_connect($host, $user, $password);
$result = fbsql_db_query("database", "select user_id, fullname from table");
while ($row = fbsql_fetch_array($result)) {
    echo "user_id: " . $row["user_id"] . "<br />¥n";
    echo "user_id: " . $row[0] . "<br />¥n";
    echo "fullname: " . $row["fullname"] . "<br />¥n";
    echo "fullname: " . $row[1] . "<br />¥n";
}
fbsql_free_result($result);
?>
```

参考

- fbsql_fetch_row()
- fbsql_fetch_assoc()
- fbsql_fetch_object()

---

fbsql_fetch_assoc

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_assoc — 連想配列として結果レコードを取得する

説明

array **fbsql_fetch_assoc** (resource `$result`)

fbsql_fetch_assoc() は、 fbsql_fetch_array() のオプションの第 2 引数に FBSQL_ASSOC を指定してコールするのと等価です。この関数は 連想配列のみを返します。

fbsql_fetch_array() も、当初はこのように動作していました。もし連想配列だけでなく 数値添字の配列も必要な場合は fbsql_fetch_array() を使用してください。

注意してほしいのは、 **fbsql_fetch_assoc()** は fbsql_fetch_row() と比べてそれほど遅くはないのに その追加機能がとても優れているということです。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

取得した行に対応する連想配列を返します。行がもうない場合には **FALSE** を返します。

2 つ以上のカラムが同じ名前を持っている場合、最後にあらわれたカラムが 優先されます。他のカラムにアクセスするには、 fbsql_fetch_array() を使用して数値添字の配列を 取得する必要があります。

例

Example#1 fbsql_fetch_assoc() の例

```
<?php
fbsql_connect($host, $user, $password);
$result = fbsql_db_query("database", "select * from table");
while ($row = fbsql_fetch_assoc($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result($result);
?>
```

参考

- fbsql_fetch_row()
- fbsql_fetch_array()
- fbsql_fetch_object()

---

fbsql_fetch_field

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_field — 結果からカラム情報を得て、オブジェクトとして返す

説明

object **fbsql_fetch_field** (resource `$result` [, int `$field_offset`])

クエリ結果のフィールドについての情報を取得するために使用されます。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

field_offset

フィールドの数値オフセット。フィールドのインデックスは 0 からはじまります。 省略した場合は、まだ **fbsql_fetch_field()** で取得されていない フィールドのうちで最小のものが取得されます。

返り値

フィールド情報を含むオブジェクト、あるいはエラー時に **FALSE** を返します。

オブジェクトのプロパティは以下のとおりです。

- name - カラム名。
- table - カラムが属するテーブルの名前。
- max_length - カラムの最大長。
- not_null - カラムが **NULL** にならない場合に 1。

- type - カラムの型。

例

Example#1 fbsql_fetch_field() の例

```
<?php
fbsql_connect($host, $user, $password)
    or die("接続できません");
$result = fbsql_db_query("database", "select * from table")
    or die("クエリに失敗");
# get column metadata
$i = 0;
while ($i < fbsql_num_fields($result)) {
    echo "カラム $i の情報:<br />\n";
    $meta = fbsql_fetch_field($result);
    if (!$meta) {
        echo "使用可能な情報がありません<br />\n";
    }
    echo "<pre>
max_length:     $meta->max_length
name:           $meta->name
not_null:       $meta->not_null
table:          $meta->table
type:           $meta->type
</pre>";
    $i++;
}
fbsql_free_result($result);
?>
```

参考

- fbsql_field_seek()

---

fbsql_fetch_lengths

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_lengths — 結果の各出力の長さを得る

説明

array **fbsql_fetch_lengths** (resource `$result`)

fbsql_fetch_row()、 fbsql_fetch_array() および fbsql_fetch_object() によって返された直近の行について 各カラムの長さを格納した配列を返します。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

オフセット 0 から始まる配列を返します。各要素は、 fbsql_fetch_row() で取得した行の各フィールドの長さを表します。 エラー時には **FALSE** を返します。

参考

- fbsql_fetch_row()

---

fbsql_fetch_object

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_object — オブジェクトとして結果レコードを取得する

説明

object **fbsql_fetch_object** (resource `$result`)

fbsql_fetch_object() は fbsql_fetch_array() と似ていますが 1 点だけ違いが あります。それは、返されるのが配列ではなくオブジェクトであるという点です。 つまり、データへのアクセスはフィールド名によってのみ可能で、 そのオフセットではアクセスできない(数値はプロパティ名として使用できません) ということです。

速度面では、この関数は fbsql_fetch_array() とまったく同等で、fbsql_fetch_row() と比べても ほぼ同じです (その差はごくわずかです)。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

取得した行に対応するプロパティをもつオブジェクトを返します。 行がもうない場合には **FALSE** を返します。

例

Example#1 fbsql_fetch_object() の例

```
<?php
fbsql_connect($host, $user, $password);
$result = fbsql_db_query("database", "select * from table");
while ($row = fbsql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result($result);
?>
```

参考

- fbsql_fetch_array()
- fbsql_fetch_row()
- fbsql_fetch_assoc()

---

fbsql_fetch_row

(PHP 4 >= 4.0.6, PHP 5)

fbsql_fetch_row — 数値配列として結果レコードを得る

説明

array **fbsql_fetch_row** (resource `$result`)

fbsql_fetch_row() は、指定した結果 ID に 関連付けられた結果から、1 行分のデータを取得します。

fbsql_fetch_row() を続けてコールすると、 結果セットの次の行を返します。行がもうない場合には **FALSE** を返します。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

取得した行は配列として返されます。各カラムは、配列の 0 からはじまる オフセットに格納されます。行がもうない場合には **FALSE** を返します。

参考

- fbsql_fetch_array()
- fbsql_fetch_assoc()
- fbsql_fetch_object()
- fbsql_data_seek()
- fbsql_fetch_lengths()
- fbsql_result()

---

fbsql_field_flags

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_flags — クエリ結果において指定したフィールドに関するフラグを得る

説明

string **fbsql_field_flags** (resource `$result` [, int `$field_offset`])

結果の指定したフィールドに関連付けられたフラグを取得します。

パラメータ

`result`

fbsql_list_fields() が返す結果ポインタ。

`field_offset`

フィールドの数値オフセット。フィールドのインデックスは 0 からはじまります。

返り値

指定したフィールドの フィールドフラグを返します。フラグは、単語ごとに空白 1 文字で 区切られた形式で報告されます。そのため、explode() を使用して返り値を分割することが可能です。

---

fbsql_field_len

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_len — 指定したフィールドの長さを返す

説明

int **fbsql_field_len** (resource `$result` [, int `$field_offset`])

指定したフィールドの長さを返します。

パラメータ

`result`

fbsql_list_fields() が返す結果ポインタ。

`field_offset`

フィールドの数値オフセット。フィールドのインデックスは 0 からはじまります。

返り値

指定したフィールドの長さを返します。

---

fbsql_field_name

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_name — 結果の指定したフィールドの名前を得る

説明

string **fbsql_field_name** (resource `$result` [, int `$field_index`])

指定したフィールドインデックスの名前を返します。

パラメータ

result

fbsql_list_fields() が返す結果ポインタ。

field_index

フィールドの数値オフセット。フィールドのインデックスは 0 からはじまります。

返り値

名前を表す文字列、あるいはフィールドが存在しない場合に **FALSE** を返します。

例

Example#1 fbsql_field_name() の例

```
<?php
// users テーブルには以下の 3 つのフィールドがあります
//    user_id
//    username
//    password.

$res = fbsql_db_query("users", "select * from users", $link);

echo fbsql_field_name($res, 0) . "¥n";
echo fbsql_field_name($res, 2);
?>
```

上の例の出力は以下となります。

```
user_id
password
```

参考

- fbsql_field_type()

---

fbsql_field_seek

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_seek — 指定したフィールドオフセットに結果ポインタを設定する

説明

bool **fbsql_field_seek** (resource `$result` [, int `$field_offset`])

指定したフィールドオフセットに移動します。 fbsql_fetch_field() がフィールドオフセットを 指定せずにコールされた場合、**fbsql_field_seek()** で指定したフィールドオフセットが返されます。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

field_offset

フィールドの数値オフセット。フィールドのインデックスは 0 からはじまります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fbsql_fetch_field()

---

fbsql_field_table

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_table — 指定したフィールドがあるテーブルの名前を得る

説明

string **fbsql_field_table** (resource `$result` [, int `$field_offset`])

指定したフィールドがあるテーブルの名前を返します。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

field_offset

フィールドの数値オフセット。フィールドのインデックスは 0 からはじまります。

返り値

テーブル名を文字列で返します。

---

fbsql_field_type

(PHP 4 >= 4.0.6, PHP 5)

fbsql_field_type — 結果の中で指定したフィールドの型を得る

説明

string **fbsql_field_type** (resource `$result` [, int `$field_offset`])

fbsql_field_type() は fbsql_field_name() 関数と同じですが、 返される内容がフィールドの型となります。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

field_offset

フィールドの数値オフセット。フィールドのインデックスは 0 からはじまります。

返り値

フィールドの型を文字列で返します。

これは *int*、*real*、 *string*、*blob* あるいは » FrontBase のドキュメント で詳細が説明されている型のいずれかです。

例

Example#1 fbsql_field_type() の例

```
<?php

fbsql_connect("localhost", "_SYSTEM", "");
fbsql_select_db("wisconsin");
$result = fbsql_query("SELECT * FROM onek;");
$fields = fbsql_num_fields($result);
$rows   = fbsql_num_rows($result);
$i = 0;
$table = fbsql_field_table($result, $i);
echo "テーブル '" . $table . "' には、フィールドが " . $fields . " そしてレコードが " . $rows . " 件あります <br />";
echo "テーブルのフィールドは以下のとおりです <br />";
while ($i < $fields) {
    $type  = fbsql_field_type($result, $i);
    $name  = fbsql_field_name($result, $i);
    $len   = fbsql_field_len($result, $i);
    $flags = fbsql_field_flags($result, $i);
    echo $type . " " . $name . " " . $len . " " . $flags . "<br />";
    $i++;
}
fbsql_close();
```

```
?>
```

参考

- fbsql_field_name()

---

fbsql_free_result

(PHP 4 >= 4.0.6, PHP 5)

fbsql_free_result — 結果メモリを開放する

説明

bool **fbsql_free_result** (resource `$result`)

`result` で指定した ID に関連付けられた全メモリを開放します。

fbsql_free_result() は、大きな結果セットを返す クエリでメモリの使用量が心配な場合にのみコールします。 結果に関連付けられたメモリは、スクリプトの実行が終了した際に自動的に開放されます。

パラメータ

`result`

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

---

fbsql_get_autostart_info

(PHP 4 >= 4.0.7, PHP 5)

fbsql_get_autostart_info —

説明

array **fbsql_get_autostart_info** ([resource `$link_identifier`])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

---

fbsql_hostname

(PHP 4 >= 4.0.6, PHP 5)

fbsql_hostname — 接続で使用されているホスト名を取得あるいは設定する

説明

string **fbsql_hostname** (resource `$link_identifier` [, string `$host_name`])

接続で使用されているホスト名を取得あるいは設定します。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

host_name

指定した場合は、これが新しい接続のホスト名となります。

返り値

現在の接続で使用しているホスト名を返します。

参考

- fbsql_username()
- fbsql_password()

---

fbsql_insert_id

(PHP 4 >= 4.0.6, PHP 5)

fbsql_insert_id — 直近の INSERT 処理により生成された ID を得る

説明

int **fbsql_insert_id** ([resource `$link_identifier`])

直近の INSERT クエリにおいて、DEFAULT UNIQUE 定義されているカラムで生成された ID を返します。

注意: FrontBase SQL関数 **fbsql_insert_id()** の値は 直近に生成された DEFAULT UNIQUE の値を常に含み、 クエリ間でリセットすることはありません。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

直近の INSERT クエリが生成した ID を返します。 直近のクエリが DEFAULT UNIQUE 値を生成していない場合は 0 を返します。

後のために値を保存する必要がある場合、値を生成するクエリの 直後にこの関数をコールするようにしてください。

参考

- fbsql_affected_rows()

---

fbsql_list_dbs

(PHP 4 >= 4.0.6, PHP 5)

fbsql_list_dbs — FrontBase サーバで利用可能なデータベースの一覧を得る

説明

resource **fbsql_list_dbs** ([resource `$link_identifier`])

fbsql デーモンで現在使用可能なデータベースを含む結果ポインタを返します。 この結果ポインタの内容を取得するには fbsql_tablename() 関数を使

用します。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

結果ポインタあるいはエラー時に **FALSE** を返します。

例

Example#1 fbsql_list_dbs() の例

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
    echo $row->Database . "¥n";
}
```

上の例の出力は、たとえば以下のようになります。

```
database1
database2
database3
...
```

注意: 上記のコードは、fbsql_fetch_row() または他の 類似の関数でも簡単に同じことが可能です。

参考

- fbsql_list_fields()
- fbsql_list_tables()

---

fbsql_list_fields

(PHP 4 >= 4.0.6, PHP 5)

fbsql_list_fields — FrontBase 結果フィールドの一覧を得る

説明

resource **fbsql_list_fields** (string `$database_name` , string `$table_name` [, resource `$link_identifier`])

指定したテーブルについての情報を取得します。

パラメータ

database_name

データベース名。

table_name

テーブル名。

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

結果ポインタを返します。これは、 *fbsql_field_xxx* で使用できます。 エラー時には **FALSE** を返します。

エラー / 例外

エラーの内容は *$phperrmsg* に書き込まれ、関数が *@fbsql()* のようにコールされていない限り このエラー内容が出力されます。

例

Example#1 fbsql_list_fields() の例

```
<?php
$link = fbsql_connect('localhost', 'myname', 'secret');

$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "¥n";;
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
field1
field2
field3
...
```

参考

- fbsql_field_len()
- fbsql_field_name()
- fbsql_field_type()
- fbsql_field_flags()

---

fbsql_list_tables

(PHP 4 >= 4.0.6, PHP 5)

fbsql_list_tables — FrontBase データベースのテーブル一覧を得る

説明

resource **fbsql_list_tables** (string `$database` [, resource `$link_identifier`])

database について説明する結果ポインタを返します。

パラメータ

database

データベース名。

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

結果ポインタを返します。これを fbsql_tablename() 関数で使用すると、 実際のテーブル名が取得できます。エラー時には **FALSE** を返します。

参考

- fbsql_list_fields()
- fbsql_list_dbs()

---

fbsql_next_result

(PHP 4 >= 4.0.6, PHP 5)

fbsql_next_result — 内部結果ポインタを次の結果に移動する

説明

bool **fbsql_next_result** (resource `$result`)

サーバに複数の SQL 文を送信したり、複数の結果を返す ストアドプロシージャを実行したりすると、複数の結果セットが返されます。 この関数は、サーバからの追加の結果セットが存在するかどうかを調べます。 結果が存在した場合、現在の結果セットを開放して新しい結果セットを取得します。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

追加の結果セットが使用可能な場合に **TRUE**、それ以外の場合に **FALSE** を返します。

例

Example#1 fbsql_next_result() の例

```
<?php
$link = fbsql_connect("localhost", "_SYSTEM", "secret");
fbsql_select_db("MyDB", $link);
$SQL = "Select * from table1; select * from table2;";
$rs = fbsql_query($SQL, $link);
do {
    while ($row = fbsql_fetch_row($rs)) {
    }
} while (fbsql_next_result($rs));
fbsql_free_result($rs);
fbsql_close($link);
?>
```

---

fbsql_num_fields

(PHP 4 >= 4.0.6, PHP 5)

fbsql_num_fields — 結果のフィールド数を得る

説明

int **fbsql_num_fields** (resource `$result`)

指定した結果セット *result* のフィールド数を返します。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

フィールドの数を整数値で返します。

参考

- fbsql_db_query()
- fbsql_query()
- fbsql_fetch_field()
- fbsql_num_rows()

---

fbsql_num_rows

(PHP 4 >= 4.0.6, PHP 5)

fbsql_num_rows — 結果のレコード数を得る

説明

int **fbsql_num_rows** (resource $result)

指定した結果セット *result* の行数を取得します。

このコマンドは SELECT 文に対してのみ使用可能です。 INSERT、UPDATE あるいは DELETE クエリの行数を取得するには、fbsql_affected_rows() を使用します。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

直近の SELECT 文が返す行の数を返します。

例

Example#1 fbsql_num_rows() の例

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows¥n";

?>
```

参考

- fbsql_affected_rows()
- fbsql_connect()
- fbsql_select_db()
- fbsql_query()

---

fbsql_password

(PHP 4 >= 4.0.6, PHP 5)

fbsql_password — 接続に対して使用するユーザパスワードを取得あるいは設定する

説明

string **fbsql_password** (resource $link_identifier [, string $password])

接続時のユーザパスワードを取得あるいは設定します。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

password

指定した場合は、これが新しい接続パスワードとなります。

返り値

現在の接続パスワードを返します。

参考

- fbsql_username()
- fbsql_hostname()

---

fbsql_pconnect

(PHP 4 >= 4.0.6, PHP 5)

fbsql_pconnect — FrontBase サーバへの持続的接続をオープンする

説明

resource **fbsql_pconnect** ([string `$hostname` [, string `$username` [, string `$password`]]])

FrontBase サーバへの持続的な接続を確立します。

サーバのポート番号を指定するには fbsql_select_db() を使用します。

fbsql_pconnect() は fbsql_connect() とほとんど同じように動作しますが、 2 つの大きな違いがあります。

まず、接続の際に、この関数は事前に同じホスト・ユーザ名・パスワードで オープンされている (持続的) リンクを探そうとします。 見つかった場合には、新しい接続をオープンせずにその接続の ID を 返します。

次に、SQL サーバへの接続はスクリプトが終了しても閉じられません。 その代わりに、今後利用されるときのためにオープンされたままとなります。

これらにより、この形式のリンクは「持続的(persistent)」と呼ばれます。

パラメータ

`hostname`

ホスト名。デフォルトは *localhost*。

`username`

接続時のユーザ名。デフォルトは *_SYSTEM*。

`password`

接続時のパスワード。デフォルトは空の文字列。

返り値

成功した場合には正の FrontBase 持続的リンク ID、 エラー時には **`FALSE`** を返します。

参考

- fbsql_connect()

---

fbsql_query

(PHP 4 >= 4.0.6, PHP 5)

fbsql_query — FrontBase クエリを送信する

説明

resource **fbsql_query** (string `$query` [, resource `$link_identifier` [, int `$batch_size`]])

クエリ *`query`* を、 サーバ上の現在アクティブなデータベースに送信します。

クエリが成功したと仮定すると、 fbsql_num_rows() を使用して SELECT 文から 返された行数を取得したり fbsql_affected_rows() を使用して DELETE、INSERT、REPLACE あるいは UPDATE 文で変更された行数を 取得したりすることが可能です。

パラメータ

query

実行する SQL クエリ。

注意: クエリ文字列の最後はセミコロンで終わらなければなりません。

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

batch_size

返り値

fbsql_query() は、クエリが成功したかどうかを 示すために **TRUE** (非ゼロ) あるいは **FALSE** を返します。 返り値が **TRUE** の場合、クエリは正しい形式であってサーバで実行された ことを示します。変更された行数や返された行数については何も示しません。 クエリが成功しても 1 行も変更しなかったり 1 行も返さなかったりすることは 十分にありえることです。

SELECT 文の場合は **fbsql_query()** は新しい結果 ID を返し、これを fbsql_result() に渡すことができます。

クエリが参照しているテーブルへのアクセス権がない場合にも **fbsql_query()** は失敗し、**FALSE** を返します。

例

以下のクエリは文法的に間違っています。そのため **fbsql_query()** は失敗して **FALSE** を返します。

Example#1 fbsql_query() の例

```
<?php
$result = fbsql_query("SELECT * WHERE 1=1")
    or die ("不正なクエリ");
?>
```

テーブル *my_tbl* に *my_col* というカラムが存在しない場合、以下のクエリは意味的に間違ったものと なります。そのため **fbsql_query()** は失敗して **FALSE** を返します。

Example#2 fbsql_query() の例

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl;")
    or die ("不正なクエリ");
?>
```

参考

- fbsql_affected_rows()
- fbsql_db_query()
- fbsql_free_result()
- fbsql_result()
- fbsql_select_db()
- fbsql_connect()

---

fbsql_read_blob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_read_blob — データベースから BLOB を読み込む

説明

string **fbsql_read_blob** (string `$blob_handle` [, resource `$link_identifier`])

データベースから BLOB データを読み込みます。

select 文が BLOB や CLOB のカラムを含んでいる場合、 FrontBase はデータが取得される際にはそのデータを直接返します。 このデフォルトの動作は fbsql_set_lob_mode() で変更することが可能で、そうするとデータの取得時には BLOB および CLOB データのハンドルを返します。ハンドルを取得した場合は、 データベースから実際の BLOB データを取得するために **fbsql_read_blob()** をコールする必要があります。

パラメータ

blob_handle

fbsql_create_blob() が返す BLOB ハンドル。

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

指定した BLOB データを含む文字列を返します。

例

Example#1 fbsql_read_blob() の例

```
<?php
$link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
$sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
$rs = fbsql_query($sql, $link);
$row_data = fbsql_fetch_row($rs);
// $row_data[0] には最初の行の blob データが含まれています
fbsql_free_result($rs);

$rs = fbsql_query($sql, $link);
fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
$row_data = fbsql_fetch_row($rs);
// $row_data[0] には最初の行の BLOB データへのハンドルが含まれています
$blob_data = fbsql_read_blob($row_data[0]);
fbsql_free_result($rs);

?>
```

参考

- fbsql_create_blob()
- fbsql_read_clob()
- fbsql_set_lob_mode()

---

fbsql_read_clob

(PHP 4 >= 4.2.0, PHP 5)

fbsql_read_clob — データベースから CLOB を読み込む

説明

string **fbsql_read_clob** (string `$clob_handle` [, resource `$link_identifier`])

データベースから CLOB データを読み込みます。

select 文が BLOB や CLOB のカラムを含んでいる場合、 FrontBase はデータが取得される際にはそのデータを直接返します。 このデフォルトの動作は fbsql_set_lob_mode() で変更することが可能で、そうするとデータの取得時には BLOB および CLOB データのハンドルを返します。ハンドルを取得した場合は、 データベースから実際の CLOB データを取得するために **fbsql_read_clob()** をコールする必要があります。

パラメータ

clob_handle

fbsql_create_clob() が返す CLOB ハンドル。

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

指定した CLOB データを含む文字列を返します。

例

Example#1 fbsql_read_clob() の例

```
<?php
$link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
    or die("接続できません");
$sql = "SELECT CLOB_COLUMN FROM CLOB_TABLE;";
$rs = fbsql_query($sql, $link);
$row_data = fbsql_fetch_row($rs);
// $row_data[0] には最初の行の clob データが含まれています
fbsql_free_result($rs);

$rs = fbsql_query($sql, $link);
fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
$row_data = fbsql_fetch_row($rs);
// $row_data[0] には最初の行の CLOB データへのハンドルが含まれています
$clob_data = fbsql_read_clob($row_data[0]);
fbsql_free_result($rs);

?>
```

参考

- fbsql_create_clob()
- fbsql_read_blob()
- fbsql_set_lob_mode()

---

fbsql_result

(PHP 4 >= 4.0.6, PHP 5)

fbsql_result — 結果データを得る

説明

mixed **fbsql_result** (resource `$result` [, int `$row` [, mixed `$field`]])

FrontBase の結果セット *result* から、ひとつのセルの内容を返します。

大きな結果セットを処理する場合には、(以下で示すような) 行全体を 取得する関数のうちのひとつを使用することを考慮すべきでしょう。 これらの関数は複数のセルの内容を一度の関数コールで取得することが 可能で、**fbsql_result()** にくらべてかなり高速です。

fbsql_result() は、その他の結果セットを扱う関数と 一緒に使用すべきではありません。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

row

field

フィールドのオフセット・フィールド名・テーブル名とフィールド名をドットで 連結した形式 (テーブル名.フィールド名) のいずれかが指定可能です。

カラム名にエイリアスが指定されている場合 ('select foo as bar from...') は、カラム名のかわりにエイリアスを使用します。

注意: フィールド名やテーブル名.フィールド名を引数に渡すのに比べると、 フィールドの数値オフセットを指定するほうが高速となります。

返り値

参考

推奨される高機能な代替関数は、以下のようになります。

- fbsql_fetch_row()

- fbsql_fetch_array()
- fbsql_fetch_assoc()
- fbsql_fetch_object()

---

fbsql_rollback

(PHP 4 >= 4.0.6, PHP 5)

fbsql_rollback — データベースへのトランザクションをロールバックする

説明

bool **fbsql_rollback** ([resource `$link_identifier`])

直近のコミット以降に発行されたすべての文をロールバックし、 現在のトランザクションを終了します。

このコマンドは、autocommit が false に設定されている場合にのみ必要となります。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- fbsql_autocommit()
- fbsql_commit()

---

fbsql_rows_fetched

(PHP 5 >= 5.1.0)

fbsql_rows_fetched — 直近の文で影響をうけた行の数を取得する

説明

int **fbsql_rows_fetched** (resource `$result`)

直近の文で影響をうけた行の数を取得します。

パラメータ

`result`

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

返り値

行の数を整数値で返します。

---

fbsql_select_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_select_db — FrontBase データベースを選択する

説明

bool **fbsql_select_db** ([string `$database_name` [, resource `$link_identifier`]])

指定したリンク ID 上の、現在アクティブなデータベースを設定します。

クライアントは、データベースへの接続時に使用するポート番号を取得する ために FBExec を使用します。データベース名が番号の場合、システムはこれをポート番号として使用し、FBExec にポート番号を問い合わせません。 FrontBase サーバを開始するには、 FRontBase -FBExec=No -port=<port number> <database name> のようにします。

この後の fbsql_query() のコールは、 アクティブなデータベースが対象となります。

パラメータ

`database_name`

選択するデータベースの名前。

データベースがパスワードで保護されている場合は、データベースを選択する前に fbsql_database_password() をコールする必要があります。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- fbsql_connect()
- fbsql_pconnect()
- fbsql_database_password()
- fbsql_query()

---

fbsql_set_characterset

(PHP 5 >= 5.1.0)

fbsql_set_characterset — 入出力文字セットを変更する

説明

void **fbsql_set_characterset** (resource `$link_identifier` , int `$characterset` [, int `$in_out_both`])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

---

fbsql_set_lob_mode

(PHP 4 >= 4.2.0, PHP 5)

fbsql_set_lob_mode — FrontBase 結果セットの LOB 取得モードを設定する

説明

bool **fbsql_set_lob_mode** (resource `$result` , int `$lob_mode`)

データベースから LOB データを取得する際のモードを設定します。

BLOB および CLOB のデータが FrontBase に格納されている場合、 それは直接的あるいは間接的に保存することが可能です。直接保存された LOB データは lob モードの設定にかかわらず常に取得されます。LOB データが 512 バイトより小さい場合は常に直接保存されます。

パラメータ

result

fbsql_query() あるいは fbsql_db_query() が返す結果 ID。

lob_mode

以下のいずれかです。

- **FBSQL_LOB_DIRECT** - LOB データは直接取得されます。 データベースから fbsql_fetch_row() やその他の関数を使用してデータを取得した場合、すべての CLOB あるいは BLOB カラムはその内容が直接返されます。 これは新規 FrontBase 結果のデフォルト値です。
- **FBSQL_LOB_HANDLE** - LOB データは、実際のデータへのハンドルとして 取得されます。fbsql_fetch_row() あるいはその他の 関数によってデータが取得された際、LOB データが間接的に格納されていた 場合はそのハンドルを、直接格納されていた場合はデータそのものを返します。 ハンドルが返された場合、それは *@'00000000000000000000000000'* のような形式の 27 バイトの文字列となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fbsql_create_blob()
- fbsql_create_clob()
- fbsql_read_blob()
- fbsql_read_clob()

---

fbsql_set_password

(PHP 5)

fbsql_set_password — 指定したユーザのパスワードを変更する

説明

bool **fbsql_set_password** (resource `$link_identifier` , string `$user` , string `$password` , string `$old_password`)

指定したユーザ *user* のパスワードを変更します。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

user

ユーザ名。

password

設定する新しいパスワード。

old_password

もとのパスワード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

fbsql_set_transaction

(PHP 4 >= 4.2.0, PHP 5)

fbsql_set_transaction — トランザクションのロックと分離レベルを設定する

説明

void **fbsql_set_transaction** (resource `$link_identifier` , int `$locking` , int `$isolation`)

トランザクションのロック *locking* および分離レベル *isolation* を設定します。

パラメータ

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

locking

設定するロックの形式。定数 **FBSQL_LOCK_DEFERRED**、 **FBSQL_LOCK_OPTIMISTIC** あるいは **FBSQL_LOCK_PESSIMISTIC** のいずれかとなります。

isolation

設定する分離レベル。定数 **FBSQL_ISO_READ_UNCOMMITTED**、 **FBSQL_ISO_READ_COMMITTED**、 **FBSQL_ISO_REPEATABLE_READ**、 **FBSQL_ISO_SERIALIZABLE** および **FBSQL_ISO_VERSIONED** のいずれかとなります。

返り値

値を返しません。

---

fbsql_start_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_start_db — ローカルまたはリモートサーバのデータベースを開始する

説明

bool **fbsql_start_db** (string `$database_name` [, resource `$link_identifier` [, string `$database_options`]])

ローカルまたはリモートサーバのデータベースを開始します。

パラメータ

database_name

データベース名。

link_identifier

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

database_options

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fbsql_db_status()

- fbsql_stop_db()

---

fbsql_stop_db

(PHP 4 >= 4.0.6, PHP 5)

fbsql_stop_db — ローカルまたはリモートサーバのデータベースを停止する

説明

bool **fbsql_stop_db** (string `$database_name` [, resource `$link_identifier`])

ローカルまたはリモートサーバのデータベースを停止します。

パラメータ

`database_name`

データベース名。

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- fbsql_db_status()
- fbsql_start_db()

---

fbsql_table_name

(PHP 4 >= 4.2.0, PHP 5)

fbsql_table_name — フィールドのテーブル名を得る

説明

string **fbsql_table_name** (resource `$result` , int `$index`)

fbsql_table_name() は、指定した結果セット *`result`* から現在のテーブル名を取得します。

fbsql_num_rows() 関数を使用して、 現在の結果ポインタ内のテーブルの数を判断します。

パラメータ

`result`

fbsql_list_tables() が返す結果ポインタ。

`index`

現在のテーブルの整数値インデックス。

返り値

テーブル名を文字列で返します。

例

Example#1 fbsql_table_name() の例

```
<?php
fbsql_connect("localhost", "_SYSTEM", "");
$result  = fbsql_list_tables("wisconsin");
$i = 0;
while ($i < fbsql_num_rows($result)) {
    $tb_names[$i] = fbsql_table_name($result, $i);
    echo $tb_names[$i] . "<br />";
    $i++;
}
?>
```

---

fbsql_tablename

(PHP 4 >= 4.2.0, PHP 5)

fbsql_tablename — fbsql_table_name() のエイリアス

説明

この関数は次の関数のエイリアスです。 fbsql_table_name().

---

fbsql_username

(PHP 4 >= 4.0.6, PHP 5)

fbsql_username — 接続に使用するホストユーザを取得あるいは設定する

説明

string **fbsql_username** (resource `$link_identifier` [, string `$username`])

接続で使用するユーザ名を取得あるいは設定します。

パラメータ

`link_identifier`

fbsql_connect() あるいは fbsql_pconnect() が返す FrontBase リンク ID。

指定しなかった場合は、この関数は FrontBase サーバでオープンしているリンクを探します。見つからないときは fbsql_connect() に引数を指定せずにコールしたときと同様にして作成します。

`username`

指定した場合は、これを新しいユーザ名として設定します。

返り値

この接続で使用する現在のユーザ名を文字列で返します。

参考

- fbsql_password()
- fbsql_hostname()

---

fbsql_warnings

(PHP 4 >= 4.0.6, PHP 5)

fbsql_warnings — FrontBase 警告を有効または無効にする

説明

bool **fbsql_warnings** ([bool `$OnOff`])

FrontBase の警告を有効または無効にします。

パラメータ

OnOff

警告を有効にするかしないか。

返り値

警告をオンにした場合は **TRUE**、そうでない場合は **FALSE** を返します。

参考

- fbsql_errno()
- fbsql_error()

目次

- fbsql_rollback — データベースへのトランザクションをロールバックする
- fbsql_rows_fetched — 直近の文で影響をうけた行の数を取得する
- fbsql_select_db — FrontBase データベースを選択する
- fbsql_set_characterset — 入出力文字セットを変更する
- fbsql_set_lob_mode — FrontBase 結果セットの LOB 取得モードを設定する
- fbsql_set_password — 指定したユーザのパスワードを変更する
- fbsql_set_transaction — トランザクションのロックと分離レベルを設定する
- fbsql_start_db — ローカルまたはリモートサーバのデータベースを開始する
- fbsql_stop_db — ローカルまたはリモートサーバのデータベースを停止する
- fbsql_table_name — フィールドのテーブル名を得る
- fbsql_tablename — fbsql_table_name のエイリアス
- fbsql_username — 接続に使用するホストユーザを取得あるいは設定する
- fbsql_warnings — FrontBase 警告を有効または無効にする

FTP 関数

導入

この拡張モジュールの関数は、» http://www.faqs.org/rfcs/rfc959 で定義された File Transfer Protocol (FTP) を使用してファイルサーバに アクセスするクライアントの実装です。この拡張を利用することで、FTP サーバにアクセスしてさまざまな操作をするスクリプトを作成することができます。ただ単に FTP サーバ上のファイルを読み書きしたいだけなら、 ファイルシステム関数 で *ftp:// ラッパ* を用いることを考えてください。こちらのほうがシンプルで直感的な インターフェースを提供します。

要件

外部ライブラリを必要としません。

インストール手順

PHP で FTP 関数を使用するには、PHP 4 をインストールする際には --enable-ftp オプション、 PHP 3 を使用する場合には --with-ftp を追加する必要があります。

Windows 版の *PHP* にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールは、1 種類のリソース型を使用します。これは FTP 接続のリソース ID で、ftp_connect() あるいは ftp_ssl_connect() により返されたものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

FTP_ASCII (integer)

FTP_TEXT (integer)

FTP_BINARY (integer)

FTP_IMAGE (integer)

FTP_TIMEOUT_SEC (integer)

詳細は、ftp_set_option() を参照ください。

以下の定数は、PHP 4.3.0で追加されました。

FTP_AUTOSEEK (integer)

詳細は、ftp_set_option() を参照してください。

FTP_AUTORESUME (integer)

GET および PUT リクエスト用のレジューム位置と開始位置を自動的に 定義します（FTP_AUTOSEEK が有効な場合のみ動作します）。

FTP_FAILED (integer)

非同期伝送が失敗しました。

FTP_FINISHED (integer)

非同期伝送が終了しました。

FTP_MOREDATA (integer)

非同期伝送がまだアクティブです。

例

Example#1 FTP の例

```
<?php
// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// 接続できたか確認する
if ((!$conn_id) || (!$login_result)) {
        echo "FTP connection has failed!";
        echo "Attempted to connect to $ftp_server for user $ftp_user_name";
        exit;
    } else {
        echo "Connected to $ftp_server, for user $ftp_user_name";
    }

// ファイルをアップロードする
$upload = ftp_put($conn_id, $destination_file, $source_file, FTP_BINARY);

// アップロード状況を確認する
if (!$upload) {
        echo "FTP upload has failed!";
    } else {
        echo "Uploaded $source_file to $ftp_server as $destination_file";
    }

// FTP ストリームを閉じる
ftp_close($conn_id);
?>
```

---

ftp_alloc

(PHP 5)

ftp_alloc — アップロードされるファイルのためのスペースを確保する

説明

bool **ftp_alloc** (resource $ftp_stream , int $filesize [, string &$result])

ALLO コマンドを FTP サーバに送信し、 アップロードされるファイルのためのスペースを確保します。

> **注意**: 多くの FTP サーバはこのコマンドをサポートしていません。これらのサーバの中には、 「そのコマンドをサポートしていない」という意味で失敗コード（**FALSE**） を返すものもあれば 「事前に確保する必要はない」という意味で成功コード（**TRUE**） を返すものもあります。このような理由から、事前のスペース確保が明示的に 要求されているサーバに対してのみこの関数を使用するようにするとよいでしょう。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

filesize

確保したいバイト数。

return

もし変数が指定されていた場合、サーバからの応答テキストの内容の 参照が格納されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_alloc() の例

```
<?php

$file = "/home/user/myfile";

/* サーバに接続する */
$conn_id = ftp_connect('ftp.example.com');
$login_result = ftp_login($conn_id, 'anonymous', 'user@example.com');

if (ftp_alloc($conn_id, filesize($file), $result)) {
  echo "Space successfully allocated on server.  Sending $file.¥n";
  ftp_put($conn_id, '/incomming/myfile', $file, FTP_BINARY);
} else {
  echo "Unable to allocate space on server.  Server said: $result¥n";
}

ftp_close($conn_id);

?>
```

参考

- ftp_put()
- ftp_fput()

---

ftp_cdup

(PHP 4, PHP 5)

ftp_cdup — 親ディレクトリに移動する

説明

bool **ftp_cdup** (resource `$ftp_stream`)

親ディレクトリに移動します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_cdup() の例

```
<?php
// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// html ディレクトリに移動する
ftp_chdir($conn_id, 'html');

echo ftp_pwd($conn_id); // /html

// 親ディレクトリに戻る
if (ftp_cdup($conn_id)) {
  echo "cdup successful¥n";
} else {
  echo "cdup not successful¥n";
}
```

```
echo ftp_pwd($conn_id); // /

ftp_close($conn_id);
?>
```

参考

- ftp_chdir()
- ftp_pwd()

---

ftp_chdir

(PHP 4, PHP 5)

ftp_chdir — FTP サーバ上でディレクトリを移動する

説明

bool **ftp_chdir** (resource `$ftp_stream` , string `$directory`)

カレントディレクトリを、指定した場所に移動します。

パラメータ

`ftp_stream`

FTP 接続のリンク ID 。

`directory`

対象となるディレクトリ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 ディレクトリの変更に失敗した場合は、PHP は警告を出します。

例

Example#1 ftp_chdir() の例

```
<?php

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// 接続できたか確認する
if ((!$conn_id) || (!$login_result)) {
    die("FTP connection has failed !");
}

echo "Current directory: " . ftp_pwd($conn_id) . "¥n";

// somedir に移動する
if (ftp_chdir($conn_id, "somedir")) {
    echo "Current directory is now: " . ftp_pwd($conn_id) . "¥n";
} else {
    echo "Couldn't change directory¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

参考

- ftp_cdup()
- ftp_pwd()

---

ftp_chmod

(PHP 5)

ftp_chmod — FTP 経由でファイルのパーミッションを設定する

説明

int **ftp_chmod** (resource $ftp_stream , int $mode , string $filename)

指定したリモートファイルのパーミッションを *mode* に設定します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

mode

新しいパーミッション。*8 進数* で指定します。

filename

リモートファイル。

返り値

成功した場合に新しいパーミッションを、エラー時に **FALSE** を返します。

例

Example#1 ftp_chmod() の例

```
<?php
$file = 'public_html/index.php';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $file のパーミッションを 644 に変更する
if (ftp_chmod($conn_id, 0644, $file) !== false) {
 echo "$file chmoded successfully to 644¥n";
} else {
 echo "could not chmod $file¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

参考

- chmod()

---

ftp_close

(PHP 4 >= 4.2.0, PHP 5)

ftp_close — FTP 接続を閉じる

説明

bool **ftp_close** (resource $ftp_stream)

ftp_close() は、指定されたリンク ID を閉じて resource を開放します。

> **注意**: この関数をコールした後は、FTP 接続を利用することはできません。 利用するためには ftp_connect() で新しい接続を 作成しなければなりません。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_close() の例

```
<?php

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// 現在のディレクトリを表示する
echo ftp_pwd($conn_id); // /

// 接続を閉じる
ftp_close($conn_id);
?>
```

参考

- ftp_connect()

---

ftp_connect

(PHP 4, PHP 5)

ftp_connect — FTP 接続をオープンする

説明

resource **ftp_connect** (string $host [, int $port [, int $timeout]])

ftp_connect() は、指定した *host* への FTP 接続をオープンします。

パラメータ

host

FTP サーバのアドレス。このパラメータには、最後のスラッシュや最初の *ftp://* をつけてはいけません。

port

このパラメータは接続先のポートを指定します。もし指定しなかったり ゼロを指定したりした場合は、デフォルトの FTP ポートである 21 が用いられます。

timeout

このパラメータは、以降のネットワーク操作時のタイムアウトを指定します。 指定されなかった場合のデフォルト値は 90 秒です。タイムアウトの変更や 参照は、ftp_set_option() や ftp_get_option() を用いていつでも可能です。

返り値

成功した場合に FTP ストリームを、エラー時に **FALSE** を返します。

例

Example#1 ftp_connect() の例

```
<?php

$ftp_server = "ftp.example.com";

// 接続を確立します。できなければ終了します。
$conn_id = ftp_connect($ftp_server) or die("Couldn't connect to $ftp_server");

?>
```

変更履歴

バージョン	説明

バージョン	説明
4.2.0	*timeout* が追加されました。

参考

- ftp_close()
- ftp_ssl_connect()

---

ftp_delete

(PHP 4, PHP 5)

ftp_delete — FTP サーバ上のファイルを削除する

説明

bool **ftp_delete** (resource $ftp_stream , string $path)

ftp_delete() は、*path* で指定したファイルを FTP サーバから削除します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

path

削除するファイル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_delete() の例

```
<?php
$file = 'public_html/old.txt';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $file の削除を試みる
if (ftp_delete($conn_id, $file)) {
 echo "$file deleted successful¥n";
} else {
 echo "could not delete $file¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

---

ftp_exec

(PHP 4 >= 4.0.3, PHP 5)

ftp_exec — FTP サーバ上でのコマンドの実行をリクエストする

説明

bool **ftp_exec** (resource $ftp_stream , string $command)

FTP サーバに SITE EXEC *command* リクエストを 送信します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

command

実行したいコマンド。

返り値

コマンドが成功した（サーバの応答コードが *200*） 場合に **TRUE** を、それ以外の場合に **FALSE** を返します。

例

Example#1 ftp_exec() の例

```
<?php

// 変数を初期化します。
$command  = 'ls  -al  >files.txt';

// 接続を確立します。
$conn_id  = ftp_connect($ftp_server);

// ユーザ名とパスワードを指定してログインします。
$login_result  = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// コマンドを実行します。
if (ftp_exec($conn_id, $command)) {
    echo "$command  の実行に成功しました\n";
} else {
    echo "$command  の実行に失敗しました\n";
}

// 接続を閉じます。
ftp_close($conn_id);

?>
```

参考

- ftp_raw()

---

ftp_fget

(PHP 4, PHP 5)

ftp_fget — FTP サーバからファイルをダウンロードし、オープン中のファイルに保存する

説明

bool **ftp_fget** (resource $ftp_stream , resource $handle , string $remote_file , int $mode [, int $resumepos])

ftp_fget() は、FTP サーバから *remote_file* を取得し、指定したファイルポインタ *fp* に書きこみます。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

handle

オープンされているファイルのポインタ。ここにデータが保存されます。

remote_file

リモートファイルのパス。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

resumepos

リモートファイル中で、ダウンロードを開始する位置。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_fget() の例

```
<?php

// リモートファイルへのパス。
$remote_file  = 'somefile.txt';
$local_file  = 'localfile.txt';

// 書き込み用のファイルをオープンします。
$handle  = fopen($local_file, 'w');

// 接続を確立します。
$conn_id  = ftp_connect($ftp_server);

// ユーザ名とパスワードを指定してログインします。
$login_result  = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $remote_file をダウンロードし、$handle に保存しようとします。
if (ftp_fget($conn_id, $handle, $remote_file, FTP_ASCII, 0)) {
 echo "$local_file への書き込みに成功しました\n";
} else {
 echo "$remote_file を $local_file にダウンロードする際に問題が発生しました\n";
}

// 接続およびファイルハンドラを閉じます。
ftp_close($conn_id);
fclose($handle);
?>
```

変更履歴

バージョン	説明
4.3.0	*resumepos* が追加されました。

参考

- ftp_get()
- ftp_nb_get()
- ftp_nb_fget()

---

ftp_fput

(PHP 4, PHP 5)

ftp_fput — オープン中のファイルを FTP サーバにアップロードする

説明

bool **ftp_fput** (resource $ftp_stream , string $remote_file , resource $handle , int $mode [, int $startpos])

ftp_fput() は、ファイルポインタが指すデータを FTP サーバ上のリモートファイルへアップロードします。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

remote_file

リモートファイルのパス。

handle

ローカルでオープンされているファイルのポインタ。 ファイルの終端まで進むと読み込みが終了します。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

startpos

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_fput() の例

```
<?php

// ファイルを読み込みモードで開く
$file = 'somefile.txt';
$fp = fopen($file, 'r');

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $file のアップロードを試みる
if (ftp_fput($conn_id, $file, $fp, FTP_ASCII)) {
    echo "Successfully uploaded $file¥n";
} else {
    echo "There was a problem while uploading $file¥n";
}

// 接続を閉じ、ファイルを閉じる
ftp_close($conn_id);
fclose($fp);

?>
```

変更履歴

バージョン	説明
4.3.0	*startpos* が追加されました。

参考

- ftp_put()
- ftp_nb_fput()
- ftp_nb_put()

---

ftp_get_option

(PHP 4 >= 4.2.0, PHP 5)

ftp_get_option — カレント FTP ストリームでの種々の実行時動作を取得する

説明

mixed **ftp_get_option** (resource $ftp_stream , int $option)

この関数は、指定した FTP 接続について *option* の値を返します。

パラメータ

ftp_stream

FTP 接続のリンク ID

option

現在、以下のオプションがサポートされています:

サポートされる実行時 FTP オプション

FTP_TIMEOUT_SEC	ネットワーク関連処理で使用されるカレントのタイムアウトを返します。
FTP_AUTOSEEK	オプションが設定されている場合に **TRUE** を、それ以外の場合に **FALSE** を返します。

返り値

成功した場合はその値を、指定した *option* がサポートされていない場合は **FALSE** を返します。後者の場合は、 同時に警告メッセージも発生します。

例

Example#1 ftp_get_option() の例

```
<?php
// 指定した FTP ストリームのタイムアウトを取得する
$timeout = ftp_get_option($conn_id, FTP_TIMEOUT_SEC);
?>
```

参考

- ftp_set_option()

---

ftp_get

(PHP 4, PHP 5)

ftp_get — FTP サーバからファイルをダウンロードする

説明

bool **ftp_get** (resource $ftp_stream , string $local_file , string $remote_file , int $mode [, int $resumepos])

ftp_get() は FTP サーバからリモートファイルを取得し、 それをローカルファイルに保存します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

local_file

ローカルファイルのパス（ファイルがすでに存在する場合は上書きされます）。

remote_file

リモートファイルのパス。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

resumepos

リモートファイルの、ダウンロードを開始する位置。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_get() の例

```
<?php

// 変数を定義する
$local_file = 'local.zip';
$server_file = 'server.zip';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $server_file をダウンロードし、$local_file への保存を試みる
if (ftp_get($conn_id, $local_file, $server_file, FTP_BINARY)) {
    echo "Successfully written to $local_file¥n";
} else {
    echo "There was a problem¥n";
}

// 接続を閉じる
ftp_close($conn_id);

?>
```

変更履歴

バージョン	説明
4.3.0	*resumepos* が追加されました。

参考

- ftp_pasv()
- ftp_fget()
- ftp_nb_get()
- ftp_nb_fget()

---

ftp_login

(PHP 4, PHP 5)

ftp_login — FTP 接続にログインする

説明

bool **ftp_login** (resource $ftp_stream , string $username , string $password)

指定した FTP ストリームにログインします。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

username

ユーザ名（*USER*）。

password

パスワード（*PASS*）。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 失敗した場合は、PHP が警告を発生します。

例

Example#1 ftp_login() の例

```
<?php

$ftp_server = "ftp.example.com";
$ftp_user = "foo";
$ftp_pass = "bar";

// 接続を確立する。接続に失敗したら終了する。
$conn_id = ftp_connect($ftp_server) or die("Couldn't connect to $ftp_server");

// ログインを試みる
if (@ftp_login($conn_id, $ftp_user, $ftp_pass)) {
    echo "Connected as $ftp_user@$ftp_server¥n";
} else {
    echo "Couldn't connect as $ftp_user¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

---

ftp_mdtm

(PHP 4, PHP 5)

ftp_mdtm — 指定したファイルが最後に更新された時刻を返す

説明

int **ftp_mdtm** (resource $ftp_stream , string $remote_file)

ftp_mdtm() はリモートファイルが最後に更新された時刻を取得します。

> **注意**: すべてのサーバがこの機能をサポートしているわけではありません!
>
> **注意**: **ftp_mdtm()** はディレクトリに対しては機能しません。

パラメータ

ftp_stream

> FTP 接続のリンク ID 。

remote_file

> 最終更新時刻を取得するファイル。

返り値

成功した場合にUNIXのタイムスタンプ、エラー時に -1 を返します。

例

Example#1 ftp_mdtm() の例

```
<?php

$file = 'somefile.txt';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// 最終更新時刻を取得する
$buff = ftp_mdtm($conn_id, $file);

if ($buff != -1) {
    // somefile.txt の最終更新時刻は: March 26 2003 14:16:41.
    echo "$file was last modified on : " . date("F d Y H:i:s.", $buff);
} else {
    echo "Couldn't get mdtime";
}

// 接続を閉じる
ftp_close($conn_id);

?>
```

---

ftp_mkdir

(PHP 4, PHP 5)

ftp_mkdir — ディレクトリを作成する

説明

string **ftp_mkdir** (resource $ftp_stream , string $directory)

FTP サーバ上に、指定した *directory* を作成します。

パラメータ

ftp_stream

> FTP 接続のリンク ID 。

directory

> 作成されるディレクトリの名前。

返り値

成功した時には新規に作成したディレクトリ名、エラー時に **FALSE** を返します。

例

Example#1 ftp_mkdir() の例

```
<?php

$dir = 'www';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// ディレクトリ $dir の作成を試みる
if (ftp_mkdir($conn_id, $dir)) {
 echo "successfully created $dir¥n";
} else {
 echo "There was a problem while creating $dir¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

参考

- ftp_rmdir()

---

ftp_nb_continue

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_continue — ファイルの取得/送信を継続する（非ブロッキング）

説明

int **ftp_nb_continue** (resource `$ftp_stream`)

非ブロッキングモードで、ファイルの取得/送信を継続します。

パラメータ

`ftp_stream`

FTP 接続のリンク ID 。

返り値

FTP_FAILED、**FTP_FINISHED** あるいは **FTP_MOREDATA** を返します。

例

Example#1 ftp_nb_continue() の例

```
<?php

// ダウンロードを開始する
$ret = ftp_nb_get($my_connection, "test", "README", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // ダウンロードを継続する…
   $ret = ftp_nb_continue($my_connection);
}
if ($ret != FTP_FINISHED) {
    echo "There was an error downloading the file...";
    exit(1);
}
?>
```

---

ftp_nb_fget

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_fget — FTP サーバからファイルをダウンロードし、オープン中のファイルに保存する（非ブロッキング）

説明

int **ftp_nb_fget** (resource $ftp_stream , resource $handle , string $remote_file , int $mode [, int $resumepos])

ftp_nb_fget() は、FTP サーバからリモートファイルを取得します。

ftp_fget() との違いは、この関数が 非同期処理でファイルを取得するということです。そのため、 ファイルをダウンロードしている最中に別の処理を行うことができます。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

handle

オープンされているファイルのポインタ。ここにデータが保存されます。

remote_file

リモートファイルのパス。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

resumepos

返り値

FTP_FAILED、**FTP_FINISHED** あるいは **FTP_MOREDATA** を返します。

例

Example#1 ftp_nb_fget() の例

```
<?php

// ファイルをオープンする
$file = 'index.php';
$fp = fopen($file, 'w');

$conn_id = ftp_connect($ftp_server);

$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// ダウンロードを開始する
$ret = ftp_nb_fget($conn_id, $fp, $file, FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // 何かお好みの動作を
   echo ".";

   // ダウンロードを継続する…
   $ret = ftp_nb_continue($conn_id);
}
if ($ret != FTP_FINISHED) {
    echo "There was an error downloading the file...";
    exit(1);
}

// ファイルポインタを閉じる
fclose($fp);
?>
```

参考

- ftp_nb_get()
- ftp_nb_continue()
- ftp_fget()
- ftp_get()

---

ftp_nb_fput

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_fput — オープン中のファイルを FTP サーバに保存する（非ブロッキング）

説明

int **ftp_nb_fput** (resource $ftp_stream , string $remote_file , resource $handle , int $mode [, int $startpos])

ftp_nb_fput() は、ファイルポインタが指すデータを FTP サーバ上のリモートファイルへアップロードします。

ftp_fput() との違いは、この関数が 非同期処理でファイルをアップロードするということです。そのため、 ファイルをアップロードしている最中に別の処理を行うことができます。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

remote_file

リモートファイルのパス。

handle

ローカルでオープンされているファイルのポインタ。 ファイルの終端まで進むと読み込みが終了します。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

startpos

返り値

FTP_FAILED、**FTP_FINISHED** あるいは **FTP_MOREDATA** を返します。

例

Example#1 ftp_nb_fput() の例

```
<?php

$file = 'index.php';

$fp = fopen($file, 'r');

$conn_id = ftp_connect($ftp_server);

$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// アップロードを開始する
$ret = ftp_nb_fput($conn_id, $file, $fp, FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // 何かお好みの動作を
   echo ".";

   // アップロードを継続する…
   $ret = ftp_nb_continue($conn_id);
}
if ($ret != FTP_FINISHED) {
    echo "There was an error uploading the file...";
    exit(1);
}

fclose($fp);
?>
```

参考

- ftp_nb_put()
- ftp_nb_continue()
- ftp_put()
- ftp_fput()

---

ftp_nb_get

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_get — FTP サーバからファイルを取得し、ローカルファイルに書き込む（非ブロッキング）

説明

int **ftp_nb_get** (resource $ftp_stream , string $local_file , string $remote_file , int $mode [, int $resumepos])

ftp_nb_get() は FTP サーバからリモートファイルを取得し、 それをローカルファイルに保存します。

ftp_get() との違いは、この関数が 非同期処理でファイルを取得するということです。そのため、 ファイルをダウンロードしている最中に別の処理を行うことができます。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

local_file

ローカルファイルのパス（ファイルがすでに存在する場合、上書きされます）。

remote_file

リモートファイルのパス。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

resumepos

返り値

FTP_FAILED、**FTP_FINISHED** あるいは **FTP_MOREDATA** を返します。

例

Example#1 ftp_nb_get() の例

```php
<?php

// ダウンロードを開始する
$ret = ftp_nb_get($my_connection, "test", "README", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // 何かお好みの動作を
   echo ".";

   // ダウンロードを継続する…
   $ret = ftp_nb_continue($my_connection);
}
if ($ret != FTP_FINISHED) {
    echo "There was an error downloading the file...";
    exit(1);
}
?>
```

Example#2 ftp_nb_get() でダウンロードを再開する

```php
<?php

// 開始
$ret = ftp_nb_get($my_connection, "test", "README", FTP_BINARY,
                  filesize("test"));
// あるいは: $ret = ftp_nb_get($my_connection, "test", "README",
//                             FTP_BINARY, FTP_AUTORESUME);
while ($ret == FTP_MOREDATA) {

   // 何かお好みの動作を
   echo ".";

   // ダウンロードを継続する…
   $ret = ftp_nb_continue($my_connection);
}
if ($ret != FTP_FINISHED) {
    echo "There was an error downloading the file...";
    exit(1);
}
?>
```

Example#3 ftp_nb_get() を用い、ファイルの 100 バイト目以降から ダウンロードを再開する

```php
<?php

// 自動シークを無効にする
ftp_set_option($my_connection, FTP_AUTOSEEK, false);

// 開始
$ret = ftp_nb_get($my_connection, "newfile", "README", FTP_BINARY, 100);
while ($ret == FTP_MOREDATA) {
```

```
    /* ... */

    // ダウンロードを継続する…
    $ret = ftp_nb_continue($my_connection);
}
?>
```

上の例では、*newfile* のサイズは FTP サーバ上の *README* より 100 バイト小さくなります。なぜなら ダウンロードの開始位置を 100 バイトずらしたからです。 もし **FTP_AUTOSEEK** を無効にしなければ、 *newfile* の最初の 100 バイトは *'¥0'* で埋められます。

参考

- ftp_nb_fget()
- ftp_nb_continue()
- ftp_fget()
- ftp_get()

---

ftp_nb_put

(PHP 4 >= 4.3.0, PHP 5)

ftp_nb_put — FTP サーバにファイルを保存する（非ブロッキング）

説明

int **ftp_nb_put** (resource `$ftp_stream` , string `$remote_file` , string `$local_file` , int `$mode` [, int `$startpos`])

ftp_nb_put() はローカルファイルを FTP サーバに保存します。

ftp_put() との違いは、この関数が 非同期処理でファイルをアップロードするということです。そのため、 ファイルをアップロードしている最中に別の処理を行うことができます。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

remote_file

リモートファイルのパス。

local_file

ローカルファイルのパス。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

startpos

返り値

FTP_FAILED、**FTP_FINISHED** あるいは **FTP_MOREDATA** を返します。

例

Example#1 ftp_nb_put() の例

```
<?php

// アップロードを開始する
$ret = ftp_nb_put($my_connection, "test.remote", "test.local", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // 何かお好みの動作を
   echo ".";

   // アップロードを継続する…
   $ret = ftp_nb_continue($my_connection);
}
if ($ret != FTP_FINISHED) {
    echo "There was an error uploading the file...";
    exit(1);
}
?>
```

Example#2 ftp_nb_put() でアップロードを再開する

```
<?php

// 開始
$ret = ftp_nb_put($my_connection, "test.remote", "test.local",
                  FTP_BINARY, ftp_size("test.remote"));
// あるいは: $ret = ftp_nb_put($my_connection, "test.remote", "test.local",
//                             FTP_BINARY, FTP_AUTORESUME);

while ($ret == FTP_MOREDATA) {

   // 何かお好みの動作を
   echo ".";

   // アップロードを継続する…
   $ret = ftp_nb_continue($my_connection);
}
if ($ret != FTP_FINISHED) {
    echo "There was an error uploading the file...";
    exit(1);
}
?>
```

参考

- ftp_nb_fput()
- ftp_nb_continue()
- ftp_put()
- ftp_fput()

---

ftp_nlist

(PHP 4, PHP 5)

ftp_nlist — 指定したディレクトリのファイルの一覧を返す

説明

array **ftp_nlist** (resource `$ftp_stream` , string `$directory`)

パラメータ

`ftp_stream`

FTP 接続のリンク ID 。

`directory`

一覧を表示するディレクトリ。このパラメータには引数を含めることができます。 例: ftp_nlist($conn_id, "-la /your/dir"); このパラメータはエスケープ処理されません。スペースやその他の文字を含む ファイル名では問題が発生する可能性があることに注意してください。

返り値

成功した場合は指定したディレクトリ内のファイル名の配列を、 エラー時には **FALSE** を返します。

例

Example#1 ftp_nlist() の例

```
<?php

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// カレントディレクトリの内容を得る
$contents = ftp_nlist($conn_id, ".");

// $contents を出力する
var_dump($contents);

?>
```

上の例の出力は、たとえば以下のようになります。

```
array(3) {
  [0]=>
   string(11) "public_html"
```

```
[1]=>
 string(10)  "public_ftp"
[2]=>
 string(3)  "www"
```

参考

- ftp_rawlist()

---

ftp_pasv

(PHP 4, PHP 5)

ftp_pasv — パッシブモードをオンまたはオフにする

説明

bool **ftp_pasv** (resource `$ftp_stream` , bool `$pasv`)

ftp_pasv() はパッシブモードをオンまたはオフにします。 パッシブモードでは、データ接続はサーバではなくクライアントにより 初期化されます。クライアントがファイアウォールの向こうにある場合に 必要となるでしょう。

ftp_pasv() をコールできるのは、 ログインに成功した後だけであることに注意しましょう。 それ以外の場合は、この関数のコールは失敗します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

pasv

TRUE, の場合はパッシブモードをオンに、そうでない場合はオフにします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_pasv() の例

```
<?php
$file = 'somefile.txt';
$remote_file = 'readme.txt';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// パッシブモードをオンにする
ftp_pasv($conn_id, true);

// ファイルをアップロードする
if (ftp_put($conn_id, $remote_file, $file, FTP_ASCII)) {
 echo "successfully uploaded $file¥n";
} else {
 echo "There was a problem while uploading $file¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

---

ftp_put

(PHP 4, PHP 5)

ftp_put — FTP サーバにファイルをアップロードする

説明

bool **ftp_put** (resource $ftp_stream , string $remote_file , string $local_file , int $mode [, int $startpos])

ftp_put() ローカルファイルを FTP サーバに保存します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

remote_file

リモートファイルのパス。

local_file

ローカルファイルのパス。

mode

転送モード。**FTP_ASCII** または **FTP_BINARY** のどちらかを指定する必要があります。

startpos

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_put() の例

```
<?php
$file = 'somefile.txt';
$remote_file = 'readme.txt';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// ファイルをアップロードする
if (ftp_put($conn_id, $remote_file, $file, FTP_ASCII)) {
 echo "successfully uploaded $file¥n";
} else {
 echo "There was a problem while uploading $file¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

変更履歴

バージョン	説明
4.3.0	*startpos* が追加されました。

参考

- ftp_pasv()
- ftp_fput()
- ftp_nb_fput()
- ftp_nb_put()

---

ftp_pwd

(PHP 4, PHP 5)

ftp_pwd — カレントのディレクトリ名を返す

説明

string **ftp_pwd** (resource $ftp_stream)

パラメータ

ftp_stream

FTP 接続のリンク ID 。

返り値

カレントのディレクトリ名、またはエラー時には **FALSE** を返します。

例

Example#1 ftp_pwd() の例

```
<?php

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// public_html ディレクトリに移動する
ftp_chdir($conn_id, 'public_html');

// カレントのディレクトリ名を表示する
echo ftp_pwd($conn_id); // /public_html

// 接続を閉じる
ftp_close($conn_id);
?>
```

参考

- ftp_chdir()
- ftp_cdup()

---

ftp_quit

(PHP 4, PHP 5)

ftp_quit — ftp_close() のエイリアス

説明

この関数は次の関数のエイリアスです。 ftp_close().

---

ftp_raw

(PHP 5)

ftp_raw — FTP サーバに任意のコマンドを送信する

説明

array **ftp_raw** (resource `$ftp_stream` , string `$command`)

任意の *command* を FTP サーバに送信します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

command

実行したいコマンド。

返り値

サーバからの応答を文字列の配列で返します。 結果の文字列に対して、何の処理も行いません。また、 **ftp_raw()** はそのコマンドが成功したのかどうか

を 判断できません。

例

Example#1 ftp_raw() を用いて FTP サーバに手動でログインする

```
<?php
$fp = ftp_connect("ftp.example.com");

/* ftp_login($fp, "joeblow", "secret");
   を実行するのと同じ意味 */
ftp_raw($fp, "USER joeblow");
ftp_raw($fp, "PASS secret");
?>
```

参考

- ftp_exec()

---

ftp_rawlist

(PHP 4, PHP 5)

ftp_rawlist — 指定したディレクトリの詳細なファイル一覧を返す

説明

array **ftp_rawlist** (resource `$ftp_stream` , string `$directory` [, bool `$recursive`])

ftp_rawlist() は、FTP **LIST** コマンドを実行し、結果を配列として返します。

パラメータ

`ftp_stream`

FTP 接続のリンク ID 。

`directory`

ディレクトリのパス。

`recursive`

TRUE を設定した場合、発行されるコマンドは **LIST -R** となります。

返り値

各要素が 1 行分のテキストに対応する配列を返します。

出力に関する処理は全く行われません。結果の解釈の仕方を定義するために ftp_systype() から返されるシステム型 ID を使用することができます。

例

Example#1 ftp_rawlist() の例

```
<?php

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// / のファイル一覧を得る
$buff = ftp_rawlist($conn_id, '/');

// 接続を閉じる
ftp_close($conn_id);

// バッファの内容を出力する
var_dump($buff);
?>
```

上の例の出力は、たとえば以下のようになります。

```
array(3) {
  [0]=>
  string(65) "drwxr-x---    3 vincent   vincent        4096 Jul 12 12:16 public_ftp"
  [1]=>
```

```
   string(66) "drwxr-x---   15 vincent   vincent        4096 Nov   3 21:31 public_html"
   [2]=>
   string(73) "lrwxrwxrwx    1 vincent   vincent          11 Jul 12 12:16 www -> public_html"
}
```

変更履歴

バージョン	説明
4.3.0	*recursive* が追加されました。

参考

- ftp_nlist()

---

ftp_rename

(PHP 4, PHP 5)

ftp_rename — FTP サーバ上のファイルまたはディレクトリの名前を変更する

説明

bool **ftp_rename** (resource $ftp_stream , string $oldname , string $newname)

ftp_rename() は FTP サーバ上のファイルやディレクトリの 名前を変更します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

oldname

現在のファイル/ディレクトリの名前。

newname

新しい名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_rename() の例

```
<?php
$old_file = 'somefile.txt.bak';
$new_file = 'somefile.txt';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $old_file を $new_file に変更することを試みる
if (ftp_rename($conn_id, $old_file, $new_file)) {
 echo "successfully renamed $old_file to $new_file¥n";
} else {
 echo "There was a problem while renaming $old_file to $new_file¥n";
}

// 接続を閉じる
ftp_close($conn_id);
?>
```

---

ftp_rmdir

(PHP 4, PHP 5)

ftp_rmdir — ディレクトリを削除する

説明

bool **ftp_rmdir** (resource $ftp_stream , string $directory)

FTP サーバ上の、指定した *directory* を削除します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

directory

削除するディレクトリ。空のディレクトリへの絶対パス または相対パスでなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ftp_rmdir() の例

```
<?php

$dir = 'www/';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $dir ディレクトリの削除を試みる
if (ftp_rmdir($conn_id, $dir)) {
     echo "Successfully deleted $dir¥n";
} else {
     echo "There was a problem while deleting $dir¥n";
}

ftp_close($conn_id);

?>
```

参考

- ftp_mkdir()

---

ftp_set_option

(PHP 4 >= 4.2.0, PHP 5)

ftp_set_option — さまざまな FTP 実行時オプションを設定する

説明

bool **ftp_set_option** (resource $ftp_stream , int $option , mixed $value)

この関数は、指定した FTP ストリームに関してのさまざまな実行時オプションを 制御します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

option

現在、以下のオプションがサポートされています:

サポートされる実行時 FTP オプション

FTP_TIMEOUT_SEC	全てのネットワーク関連関数に関して秒単位でタイムアウトを変更 します。*value* は、 0 より大きい整数値である必要があります。 デフォルトのタイムアウトは90秒です。
FTP_AUTOSEEK	有効になっている場合は、GET や PUT のリクエストが *resumepos* や *startpos* のパラメータ付きで実行されるとファイル中の該当位置をシークします。 デフォルトで有効になっています。

value

このパラメータの内容は、どの *option* を変更しようとしているかによって変わります。

返り値

オプションが設定できた場合に **TRUE** 、そうでない場合に **FALSE** を返します。 *option* がサポートされていなかった場合や *option* が想定していない値を *value* に渡した場合は警告メッセージが発生します。

例

Example#1 ftp_set_option() の例

```
<?php
// ネットワークのタイムアウトを 10 秒に設定する
ftp_set_option($conn_id, FTP_TIMEOUT_SEC, 10);
?>
```

参考

- ftp_get_option()

---

ftp_site

(PHP 4, PHP 5)

ftp_site — SITEコマンドをサーバに送信する

説明

bool **ftp_site** (resource `$ftp_stream` , string `$command`)

ftp_site() は、指定された *SITE* コマンドを FTP サーバに送信します。

SITE コマンドの規格は統一されていないため、 サーバにより仕様が異なります．ファイルのパーミッションや グループメンバーの設定のような処理を行う際には有用です。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

command

SITE コマンド。このパラメータはエスケープされないので、スペースや その他の文字を含むファイル名は問題を引き起こす可能性があることに注意してください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 SITE コマンドを FTP サーバに送信する

```
<?php
/* Connect to FTP server */
$conn = ftp_connect('ftp.example.com');
if (!$conn) die('Unable to connect to ftp.example.com');

/* "user" として、パスワード "pass" でログインする */
if (!ftp_login($conn, 'user', 'pass')) die('Error logging into ftp.example.com');

/* FTP サーバに "SITE CHMOD 0600 /home/user/privatefile" コマンドを発行する */
if (ftp_site($conn, 'CHMOD 0600 /home/user/privatefile')) {
```

```
    echo "Command executed successfully.¥n";
} else {
    die('Command failed.');
}
?>
```

参考

- ftp_raw()

---

ftp_size

(PHP 4, PHP 5)

ftp_size — 指定したファイルのサイズを返す

説明

int **ftp_size** (resource $ftp_stream , string $remote_file)

ftp_size() は指定されたファイルのサイズを バイト数で返します。

> **注意**: すべてのサーバがこの機能をサポートしているわけではありません。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

remote_file

リモートファイル。

返り値

成功した場合はファイルのサイズを、エラー時には -1 を返します。

例

Example#1 ftp_size() の例

```
<?php

$file = 'somefile.txt';

// 接続を確立する
$conn_id = ftp_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// $file のサイズを取得する
$res = ftp_size($conn_id, $file);

if ($res != -1) {
    echo "size of $file is $res bytes";
} else {
    echo "couldn't get the size";
}

// 接続を閉じる
ftp_close($conn_id);

?>
```

参考

- ftp_rawlist()

---

ftp_ssl_connect

(PHP 4 >= 4.3.0, PHP 5)

ftp_ssl_connect — セキュアな SSL-FTP 接続をオープンする

説明

resource **ftp_ssl_connect** (string $host [, int $port [, int $timeout]])

ftp_ssl_connect() は、指定した *host* への SSL-FTP 接続をオープンします。

> **注意**: **この関数が存在しないことがあるのはなぜですか？ ftp_ssl_connect()** は、 OpenSSL サポートがあなたの使っている PHP で有効になっているときにのみ使うことができます。 FTP サポートを有効にしているにもかかわらず この関数が定義されていないのは、それが理由です。 Windows でこの関数のサポートを有効にするには、自分で PHP バイナリを コンパイルする必要があります。

パラメータ

host

> FTP サーバのアドレス。このパラメータには、最後のスラッシュや 先頭の *ftp://* をつけてはいけません。

port

> *port* パラメータは別のポートに接続することを 指定します。これを省略するか 0 にした場合、デフォルトの FTP ポート、 つまり 21 が使用されます。

timeout

> このパラメータは、以降の全てのネットワーク処理の タイムアウトを指定します。省略された場合の デフォルト値は、90 秒となります。 timeout は、 ftp_set_option() および ftp_get_option() でいつでも変更および取得可能です。

返り値

成功した場合に SSL-FTP ストリーム、エラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.2.2	この関数は、SSL 接続を使用できないときに **FALSE** を返すようになりました。 これまでは、SSL ではない接続に移行していました。

例

Example#1 ftp_ssl_connect() の例

```
<?php

// SSL 接続を確立する
$conn_id  = ftp_ssl_connect($ftp_server);

// ユーザ名とパスワードでログインする
$login_result  = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

echo ftp_pwd($conn_id); // /

// SSL 接続を閉じる
ftp_close($conn_id);
?>
```

参考

- ftp_connect()

---

ftp_systype

(PHP 4, PHP 5)

ftp_systype — リモート FTP サーバのシステム型 ID を返す

説明

string **ftp_systype** (resource $ftp_stream)

リモート FTP サーバのシステム型 ID を返します。

パラメータ

ftp_stream

FTP 接続のリンク ID 。

返り値

リモートシステム型を返します。エラー時には **FALSE** を返します。

例

Example#1 ftp_systype() の例

```
<?php

// 接続を確立する
$ftp = ftp_connect('ftp.example.com');
ftp_login($ftp, 'user', 'password');

// システム型を取得する
if ($type = ftp_systype($ftp)) {
    echo "Example.com is powered by $type¥n";
} else {
    echo "Couldn't get the systype";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
Example.com is powered by UNIX
```

目次

関数処理関数(funchand)

導入

これらの関数は、関数で行う様々な処理を行います。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

定義済み定数

定数は定義されていません。

---

call_user_func_array

(PHP 4 >= 4.0.4, PHP 5)

call_user_func_array — パラメータの配列を指定してユーザ関数をコールする

説明

mixed **call_user_func_array** (callback `$function` , array `$param_arr`)

param_arr にパラメータを指定して、 *function* で指定したユーザ定義関数をコールします。

パラメータ

function

コールする関数。

param_arr

関数に渡すパラメータを指定する配列。

返り値

関数の結果、あるいはエラー時に **FALSE** を返します。

例

Example#1 call_user_func_array() の例

```
<?php
function debug($var, $val)
{
    echo "***DEBUGGING¥nVARIABLE:  $var¥nVALUE:";
    if (is_array($val) || is_object($val) || is_resource($val)) {
        print_r($val);
    } else {
        echo "¥n$val¥n";
    }
    echo "***¥n";
}

$c = mysql_connect();
$host = $_SERVER["SERVER_NAME"];

call_user_func_array('debug', array("host", $host));
call_user_func_array('debug', array("c", $c));
call_user_func_array('debug', array("_POST", $_POST));
?>
```

注意

注意: *param_arr* 内で参照される変数は、 関数に参照渡しされます。それ以外は値渡しとなります。 つまり、パラメータが値渡しとなるか参照渡しとなるのかは、 関数のシグネチャには依存しないということです。

参考

- call_user_func()
- callback 型に関する情報

---

call_user_func

(PHP 4, PHP 5)

call_user_func — 最初の引数で指定したユーザ関数をコールする

説明

mixed **call_user_func** (callback `$function` [, mixed `$parameter` [, mixed `$...`]])

パラメータ *function* で指定した ユーザ定義のコールバック関数をコールします。

パラメータ

function

コールする関数。このパラメータに *array($classname, $methodname)* を指定することにより、 クラスメソッドも静的にコールすることができます。

parameter

この関数に渡す、ゼロ個以上のパラメータ。

注意: **call_user_func()** のパラメータは 参照渡しではないことに注意しましょう。

```
<?php
function increment(&$var)
{
    $var++;
}

$a = 0;
call_user_func('increment', $a);
echo $a; // 0

call_user_func_array('increment', array(&$a)); //  このようにしてもかまいません
echo $a; // 1
?>
```

返り値

関数の結果、あるいはエラー時に **FALSE** を返します。

例

Example#1 call_user_func() の例

```
<?php
function barber($type)
{
    echo "$type ですね、わかりました。";
}
call_user_func('barber', "マッシュルームカット");
call_user_func('barber', "髭剃り");
?>
```

Example#2 クラスメソッドの使用

```
<?php
class myclass {
     function say_hello()
    {
        echo "Hello!¥n";
    }
}

$classname = "myclass";
```

```
call_user_func(array($classname, 'say_hello'));
?>
```

参考

- call_user_func_array()
- is_callable()
- callback 型に関する情報

---

create_function

(PHP 4 >= 4.0.1, PHP 5)

create_function — 匿名関数 (ラムダ形式) を作成する

説明

string **create_function** (string $args , string $code)

指定したパラメータにより匿名関数を作成し、その関数のユニークな名前を返します。

パラメータ

通常、*args* には、シングルクオートで括った文字列を 指定し、*code* の場合も同様に指定することが推奨されます。 シングルクオートで括った文字列を使用する理由は、パース時に変数名を保護するためです。 ダブルクオートを使用した場合には、*¥$avar*のように変数名を エスケープする必要があります。

args

関数の引数。

code

関数のコード。

返り値

一意な関数名を表す文字列、あるいはエラー時に **FALSE** を返します。

例

Example#1 create_function() による匿名関数の作成

この関数を使用すると、(たとえば) 実行時に取得した情報をもとにして関数を作成できます。

```
<?php
$newfunc = create_function('$a,$b', 'return "ln($a) + ln($b) = " . log($a * $b);');
echo "新しい匿名関数: $newfunc\n";
echo $newfunc(2, M_E) . "¥n";
// 出力
// 新しい匿名関数: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
?>
```

もしくは、パラメータリストに一連の処理を行うことができる 一般的なハンドラ関数を定義できます。

Example#2 create_function() による一般的な処理関数の作成

```
<?php
function process($var1, $var2, $farr)
{
    foreach ($farr as $f) {
        echo $f($var1, $var2) . "¥n";
    }
}

// 数学関数群を作成します
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
$f2 = "return ¥"min(b^2+a, a^2,b) = ¥".min(¥$a*¥$a+¥$b,¥$b*¥$b+¥$a);";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b; } else { return false; }';
$farr = array(
    create_function('$x,$y', 'return "some trig: ".(sin($x) + $x*cos($y));'),
    create_function('$x,$y', 'return "a hypotenuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
    create_function('$a,$b', $f3)
    );

echo "\n無名関数群の最初の配列を使用します\n";
```

```
echo "パラメータ: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);

// 文字列処理関数群を作成します
$garr = array(
    create_function('$b,$a', 'if (strncmp($a, $b, 3) == 0) return "** ¥"$a¥" '.
    'and ¥"$b¥"¥n** Look the same to me! (looking at the first 3 chars)";'),
    create_function('$a,$b', '; return "CRCs: " . crc32($a) . " , ".crc32(b);'),
    create_function('$a,$b', '; return "similar(a,b) = " . similar_text($a, $b, &$p) . "($p%)";')
    );
echo "\n無名関数群の二番目の配列を使用します\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
?>
```

上の例の出力は以下となります。

```
無名関数群の最初の配列を使用します
パラメータ: 2.3445, M_PI
some trig: -1.6291725057799
a hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

無名関数群の二番目の配列を使用します
** "Twas the night" and "Twas brilling and the slithy toves"
** Look the same to me! (looking at the first 3 chars)
CRCs: -725381282 , 1908338681
similar(a,b) = 11(45.833333333333%)
```

ラムダ形式の (匿名の) 関数の最も一般的な使用法は、 コールバック関数を作成することでしょう。たとえば array_walk() あるいは usort() などで使用します。

Example#3 コールバック関数としての匿名関数の使用法

```
<?php
$av = array("the ", "a ", "that ", "this ");
array_walk($av, create_function('&$v,$k', '$v = $v . "mango";'));
print_r($av);
?>
```

上の例の出力は以下となります。

```
Array
(
   [0] => the mango
   [1] => a mango
   [2] => that mango
   [3] => this mango
)
```

文字列の配列が、短い順に並べ替えられます。

```
<?php

$sv = array("small", "larger", "a big string", "it is a string thing");
print_r($sv);

?>
```

上の例の出力は以下となります。

```
Array
(
   [0] => small
   [1] => larger
   [2] => a big string
   [3] => it is a string thing
)
```

長い順に並べ替えます。

```
<?php

usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);

?>
```

上の例の出力は以下となります。

```
Array
(
   [0] => it is a string thing
   [1] => a big string
   [2] => larger
```

```
   [3] => small
)
```

---

func_get_arg

(PHP 4, PHP 5)

func_get_arg — 引数のリストから要素をひとつ返す

説明

mixed **func_get_arg** (int $arg_num)

ユーザが定義した関数の引数リストから、指定した引数を取得します。

この関数は、 func_num_args()および func_get_args()と組み合わせて使用され、これにより ユーザ定義の関数が可変長の引数リストをとることができるようになります。

パラメータ

arg_num

引数の位置。関数の引数はゼロから数え始めます。

返り値

指定した引数、あるいはエラー時に **FALSE** を返します。

エラー / 例外

ユーザ定義関数の外部からコールされた場合、あるいは *arg_num* が実際に渡された引数の数より多い場合に警告を発生します。

例

Example#1 func_get_arg() の例

```
<?php
function foo()
{
    $numargs = func_num_args();
    echo "Number of arguments: $numargs<br />¥n";
    if ($numargs >= 2) {
        echo "二番目の引数は " . func_get_arg(1) . " です。<br />\n";
    }
}

foo (1, 2, 3);
?>
```

注意

注意: この関数は、カレントスコープに依存してパラメータの詳細を決定しますので、関数パラメータとして使用することはできません。もし、この値を渡さなければならない場合、戻り値を変数に割り当て、その変数を渡してください。

注意: この関数は、渡された引数のみのコピーを返します。 デフォルトの (渡されていない) 引数については感知しません。

参考

- func_get_args()
- func_num_args()

---

func_get_args

(PHP 4, PHP 5)

func_get_args — 関数の引数リストを配列として返す

説明

array **func_get_args** (void)

関数の引数リストを配列で取得します。

この関数は func_num_args() および func_get_arg() と組み合わせて使用され、 これによりユーザ定義の章において可変長の引数リストを使用することができるようになります。

返り値

配列を返します。この配列の各要素は、 現在のユーザ定義関数の引数リストにおける対応するメンバのコピーとなります。

エラー / 例外

ユーザ定義関数の外部からコールされた際に警告を発生します。

例

Example#1 func_get_args() の例

```
<?php
function foo()
{
    $numargs = func_num_args();
    echo "引数の数: $numargs<br />\n";
    if ($numargs >= 2) {
        echo "二番目の引数は: " . func_get_arg(1) . " です。<br />\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
        echo "引数 $i は: " . $arg_list[$i] . " です。<br />\n";
    }
}

foo(1, 2, 3);
?>
```

注意

> **注意**: この関数は、カレントスコープに依存してパラメータの詳細を決定しますので、関数パラメータとして使用することはできません。もし、この値を渡さなければならない場合、戻り値を変数に割り当て、その変数を渡してください。
>
> **注意**: この関数は、渡された引数のみのコピーを返します。 デフォルトの（渡されていない）引数については考慮しません。

参考

- func_get_arg()
- func_num_args()

---

func_num_args

(PHP 4, PHP 5)

func_num_args — 関数に渡された引数の数を返す

説明

int **func_num_args** (void)

関数に渡された引数の数を取得します。

この関数は func_get_arg() および func_get_args() と組み合わせて使用され、 ユーザ定義関数において可変長の引数リストを使用することができるようになります。

返り値

現在のユーザ定義関数に渡された引数の数を返します。

エラー / 例外

ユーザ定義関数の外部からコールされた場合に警告を発生します。

例

Example#1 func_num_args() の例

```
<?php
function  foo()
{
    $numargs  = func_num_args();
    echo "引数の数:  $numargs\n";
}

foo(1, 2, 3);    // '引数の数: 3' という出力になります
?>
```

注意

> **注意**: この関数は、カレントスコープに依存してパラメータの詳細を決定しますので、関数パラメータとして使用することはできません。もし、この値を渡さなければならない場合、戻り値を変数に割り当て、その変数を渡してください。

参考

- func_get_arg()
- func_get_args()

---

function_exists

(PHP 4, PHP 5)

function_exists — 指定した関数が定義されている場合に **TRUE** を返す

説明

bool **function_exists** (string `$function_name`)

組み込みの内部関数およびユーザ定義関数の中から、 *function_name* で指定した名前の関数を探します。

パラメータ

function_name

> 関数名を表す文字列。

返り値

function_name が存在し、関数である場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

> **注意**: この関数は、 include_once() や echo() のような言語構造については **FALSE** を返します。

例

Example#1 function_exists() の例

```
<?php
if (function_exists('imap_open')) {
    echo "IMAP  関数が利用可能です。<br  />\n";
} else {
    echo "IMAP  関数は利用できません。<br  />\n";
}
?>
```

注意

> **注意**: ある関数がそれ自体設定やコンパイルオプションの問題で使用できない 場合でもその関数の名前が存在する可能性があることに注意してください (例としては image 関数などがあります)。

参考

- method_exists()
- is_callable()
- get_defined_functions()

---

get_defined_functions

(PHP 4 >= 4.0.4, PHP 5)

get_defined_functions — 定義済みの全ての関数を配列で返す

説明

array **get_defined_functions** (void)

すべての定義済み関数を配列で返します。

返り値

この関数は、組込 (内部) 関数およびユーザ定義関数を共に含む定義済み の全ての関数のリストを有する多次元配列を返します。内部関数は、*$arr["internal"]*、ユーザ定義関数は *$arr["user"]* によりアクセス可能となります (以下の例を参照ください)。

例

Example#1 get_defined_functions() の例

```
<?php
function myrow($id, $data)
{
    return "<tr><th>$id</th><td>$data</td></tr>¥n";
}

$arr = get_defined_functions();

print_r($arr);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            [3] => func_get_args
            [4] => strlen
            [5] => strcmp
            [6] => strncmp
            ...
            [750] => bcscale
            [751] => bccomp
        )

    [user] => Array
        (
            [0] => myrow
        )

)
```

参考

- function_exists()
- get_defined_vars()
- get_defined_constants()

---

register_shutdown_function

(PHP 4, PHP 5)

register_shutdown_function — シャットダウン時に実行する関数を登録する

説明

void **register_shutdown_function** (callback `$function` [, mixed `$parameter` [, mixed `$...`]])

スクリプト処理が完了した際に実行される *func* という名前の関数を登録します。

register_shutdown_function() は複数回コールする ことが可能で、登録された順に関数がコールされます。 登録した関数内で exit() をコールした場合、 処理はそこで終了してその他のシャットダウン関数はコールされません。

PHP 4.0.6 以前のバージョンを Apache で動かしている場合、 登録されたシャットダウン関数はリクエストの完了後（結果の出力を含む） に実行され

ます。そのため、echo() や print() を用いてブラウザに出力したり ob_get_contents() を用いて出力バッファの内容を取得したりすることができませんでした。 PHP 4.1 以降、シャットダウン関数はリクエストの一部として実行されるようになり、 そこから結果を出力できるようになりました。現在のところ、 シャットダウン関数の中で出力バッファリング関数を用いてデータを加工する方法はありません。 シャットダウン関数はすべての出力バッファを閉じてからコールされます。 そのため、たとえ zlib.output_compression が有効になっていたとしても出力結果は圧縮されません。

PHP 4 以降、**register_shutdown_function()** に追加のパラメータを渡すことで シャットダウン関数にパラメータを渡せるようになりました。

パラメータ

function

parameter

...

返り値

値を返しません。

注意

> **注意**: 一般に、PHP では未定義の関数は致命的なエラーを引き起こします。 しかし **register_shutdown_function()** で指定された *function* が未定義の場合はそのかわりに **E_WARNING** レベルのエラーが発生します。 また、PHP の内部的な理由により、このエラーは *Unknown* の 0 行目で発生したとみなされます。
>
> **注意**: Apache などいくつかの Web サーバでは、スクリプトの実行時ディレクトリを シャットダウン関数内で変更可能です。
>
> **注意**: シャットダウン関数はスクリプトがシャットダウンする際にコールされるので、 その際は常にヘッダが送信されています。

参考

- auto_append_file
- exit()
- 接続処理

---

register_tick_function

(PHP 4 >= 4.0.3, PHP 5)

register_tick_function — 各 tick で実行する関数を登録する

説明

bool **register_tick_function** (callback `$function` [, mixed `$arg` [, mixed `$...`]])

tick がコールされた際に実行される *func* という名前の関数を登録します。

パラメータ

function

> 関数名を表す文字列、あるいはオブジェクトとメソッドを指定した配列。

arg

...

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 register_tick_function() の例

```
<?php
// 関数をコールバックとして使用します
register_tick_function('my_function', true);

// オブジェクトのメソッドを使用します
$object = new my_class();
```

```
register_tick_function(array(&$object, 'my_method'), true);
?>
```

注意

警告

register_tick_function() はスレッド化された Web サーバモジュールとともに使用することはできません。ZTS モードでは Tick は動作せず、Web サーバがクラッシュしてしまいます。

参考

- declare
- unregister_tick_function()

---

unregister_tick_function

(PHP 4 >= 4.0.3, PHP 5)

unregister_tick_function — 各 tick の実行用の関数の登録を解除する

説明

void **unregister_tick_function** (string `$function_name`)

function_name という名前の関数の登録を解除します。 これにより、tick がコールされた 場合でもこの関数は実行されなくなります。

パラメータ

function_name

関数名を表す文字列。

返り値

値を返しません。

参考

- register_tick_function()

目次

GeoIP 関数

導入

GeoIP 拡張モジュールを使用すると、IP アドレスの場所を探すことができます。 市、州、国、経度、緯度そして ISP や接続方式などの情報が GeoIP を用いて取得できます。

要件

この拡張モジュールを使用するには、GeoIP C ライブラリのバージョン 1.4.0 以降がインストールされていなければなりません。最新のバージョンは » http://www.maxmind.com/app/c から取得できるので、これをコンパイルします。

デフォルトでは、Free GeoIP Country データベースあるいは GeoLite City データベースにしかアクセスできません。 しかし、それ以外のデータベースを使用することも可能です。その場合には » Maxmind から商用ライセンスを購入しなければなりません。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/geoip.

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

GEOIP_COUNTRY_EDITION (integer)
GEOIP_REGION_EDITION_REV0 (integer)
GEOIP_CITY_EDITION_REV0 (integer)
GEOIP_ORG_EDITION (integer)
GEOIP_ISP_EDITION (integer)
GEOIP_CITY_EDITION_REV1 (integer)
GEOIP_REGION_EDITION_REV1 (integer)
GEOIP_PROXY_EDITION (integer)
GEOIP_ASNUM_EDITION (integer)
GEOIP_NETSPEED_EDITION (integer)
GEOIP_DOMAIN_EDITION (integer)

以下の定数は、ネット接続のスピードを表します。

GEOIP_UNKNOWN_SPEED (integer)
GEOIP_DIALUP_SPEED (integer)
GEOIP_CABLEDSL_SPEED (integer)
GEOIP_CORPORATE_SPEED (integer)

---

geoip_country_code_by_name

(PECL geoip:0.2.0-1.0.1)

geoip_country_code_by_name — 二文字の国コードを取得する

説明

string **geoip_country_code_by_name** (string `$hostname`)

geoip_country_code_by_name() 関数は、 ホスト名あるいは IP アドレスに対応する二文字の国コードを返します。

パラメータ

`hostname`

場所を探す対象となるホスト名あるいは IP アドレス。

返り値

成功した場合には二文字の ISO 国コード、 アドレスがデータベースで見つからない場合には **FALSE** を返します。

例

Example#1 geoip_country_code_by_name() の例

これは、ホスト example.com がどこにあるのかを表示します。

```
<?php
$country  = geoip_country_code_by_name('www.example.com');
if ($country) {
    echo 'This host is located in: ' . $country;
}
?>
```

上の例の出力は以下となります。

```
This host is located in: US
```

参考

- geoip_country_code3_by_name()
- geoip_country_name_by_name()

---

geoip_country_code3_by_name

(PECL geoip:0.2.0-1.0.1)

geoip_country_code3_by_name — 三文字の国コードを取得する

説明

string **geoip_country_code3_by_name** (string $hostname)

geoip_country_code3_by_name() 関数は、 ホスト名あるいは IP アドレスに対応する三文字の国コードを返します。

パラメータ

hostname

場所を探す対象となるホスト名あるいは IP アドレス。

返り値

成功した場合には三文字の国コード、 アドレスがデータベースで見つからない場合には **FALSE** を返します。

例

Example#1 geoip_country_code3_by_name() の例

これは、ホスト example.com がどこにあるのかを表示します。

```
<?php
$country  = geoip_country_code3_by_name('www.example.com');
if ($country) {
    echo 'This host is located in: ' . $country;
}
?>
```

上の例の出力は以下となります。

```
This host is located in: USA
```

参考

- geoip_country_code_by_name()
- geoip_country_name_by_name()

---

geoip_country_name_by_name

(PECL geoip:0.2.0-1.0.1)

geoip_country_name_by_name — 完全な国名を取得する

説明

string **geoip_country_name_by_name** (string $hostname)

geoip_country_name_by_name() 関数は、 ホスト名あるいは IP アドレスに対応する完全な国名を返します。

パラメータ

hostname

場所を探す対象となるホスト名あるいは IP アドレス。

返り値

成功した場合には国名、 アドレスがデータベースで見つからない場合には **FALSE** を返します。

例

Example#1 geoip_country_name_by_name() の例

これは、ホスト example.com がどこにあるのかを表示します。

```
<?php
$country  = geoip_country_name_by_name('www.example.com');
if ($country) {
    echo 'This host is located in: ' . $country;
}
?>
```

上の例の出力は以下となります。

```
This host is located in: United States
```

参考

- geoip_country_code_by_name()
- geoip_country_code3_by_name()

---

geoip_database_info

(PECL geoip:0.2.0-1.0.1)

geoip_database_info — GeoIP データベースの情報を取得する

説明

string **geoip_database_info** ([int $database])

geoip_database_info() 関数は、 GeoIP データベースのバージョンを返します。 このバージョンはバイナリファイルの内部で定義されています。

この関数を引数なしでコールすると、GeoIP Free Country Edition のバージョンを返します。 ents, it returns the version of the GeoIP Free Country Edition.

パラメータ

database

データベースの型を整数値で指定します。この拡張モジュールで定義されている さまざまな定数 (例: GEOIP_*_EDITION) を使用することができます。

返り値

対応するデータベースのバージョン、あるいはエラー時に **NULL** を返します。

例

Example#1 geoip_region_by_name() の例

これは、現在のデータベースのバージョンを文字列で返します。

```
<?php
print geoip_database_info(GEOIP_COUNTRY_EDITION);
?>
```

上の例の出力は以下となります。

```
GEO-106FREE 20060801 Build 1 Copyright (c) 2006 MaxMind LLC All Rights Reserved
```

---

geoip_db_avail

(PECL geoip:1.0.1)

geoip_db_avail — GeoIP データベースが使用可能かどうかを調べる

説明

bool **geoip_db_avail** (int $database)

geoip_db_avail() 関数は、対応する GeoIP データベースが存在し、ディスク上にオープンできるかどうかを調べます。

これは、そのファイルが適切なデータベース形式かどうかを調べるものではありません。 単にそのファイルが読み込み可能かどうかを調べるだけです。

パラメータ

database

データベースの形式を表す整数値。この拡張モジュールで定義している さまざまな定数 (GEOIP_*_EDITION) を使用することができます。

返り値

データベースが存在する場合に **TRUE**、存在しない場合に **FALSE**、 エラー時に **NULL** を返します。

例

Example#1 geoip_db_avail() の例

これは、現在のデータベースのバージョンを文字列で出力します。

```
<?php

if (geoip_db_avail(GEOIP_COUNTRY_EDITION))
    print geoip_database_info(GEOIP_COUNTRY_EDITION);
?>
```

上の例の出力は以下となります。

```
GEO-106FREE 20080801 Build 1 Copyright (c) 2006 MaxMind LLC All Rights Reserved
```

---

geoip_db_filename

(PECL geoip:1.0.1)

geoip_db_filename — 対応する GeoIP データベースのファイル名を返す

説明

string **geoip_db_filename** (int $database)

geoip_db_filename() 関数は、 対応する GeoIP データベースのファイル名を返します。

これは、そのファイルが存在するかどうかを表すものではありません。 単に、ライブラリがデータベースを探す先を返すだけのものです。

パラメータ

database

データベースの形式を表す整数値。この拡張モジュールで定義している さまざまな定数 (GEOIP_*_EDITION) を使用することができます。

返り値

対応するデータベースのファイル名、あるいはエラー時に **NULL** を返します。

例

Example#1 geoip_db_filename() の例

これは、対応するデータベースのファイル名を出力します。

```
<?php
print geoip_db_filename(GEOIP_COUNTRY_EDITION);
?>
```

上の例の出力は以下となります。

```
/usr/share/GeoIP/GeoIP.dat
```

---

geoip_db_get_all_info

(PECL geoip:1.0.1)

geoip_db_get_all_info — すべての GeoIP データベース形式についての詳細情報を返す

説明

array **geoip_db_get_all_info** (void)

geoip_db_get_all_info() 関数は、 すべての GeoIP データベース形式についての詳細情報を多次元配列で返します。

この関数は、データベースがインストールされていない場合でも使用可能です。 その場合は、使用できないことが返されます。

返される連想配列のキーは、次のようになります。

- "available" -- DB が使用可能かどうか (geoip_db_avail() を参照ください)
- "description" -- データベースについての説明
- "filename" -- ディスク上でのデータベースのファイル名 (geoip_db_filename() を参照ください)

返り値

連想配列を返します。

例

Example#1 geoip_db_get_all_info() の例

これは、すべての情報を含む配列を表示します。

```
<?php
$infos = geoip_db_get_all_info();
if (is_array($infos)) {
    var_dump($infos);
}
?>
```

上の例の出力は以下となります。

```
array(11) {
  [1]=>
  array(3) {
    ["available"]=>
    bool(true)
    ["description"]=>
    string(21) "GeoIP Country Edition"
    ["filename"]=>
     string(32) "/usr/share/GeoIP/GeoIP.dat"
  }

[ ... ]

  [11]=>
  array(3) {
    ["available"]=>
```

```
      bool(false)
      ["description"]=>
      string(25) "GeoIP Domain Name Edition"
      ["filename"]=>
       string(38) "/usr/share/GeoIP/GeoIPDomain.dat"
  }
}
```

Example#2 geoip_db_get_all_info() の例

さまざまな定数を使用することで、特定の情報のみを取得することができます。

```
<?php
$infos = geoip_db_get_all_info();
if ($infos[GEOIP_COUNTRY_EDITION]['available']) {
    echo $infos[GEOIP_COUNTRY_EDITION]['description'];
}
?>
```

上の例の出力は以下となります。

```
GeoIP Country Edition
```

---

geoip_id_by_name

(PECL geoip:0.2.0-1.0.1)

geoip_id_by_name — インターネット接続のスピードを取得する

説明

int **geoip_id_by_name** (string `$hostname`)

geoip_id_by_name() 関数は、 ホスト名あるいは IP アドレスに対応する国および地域を返します。

返り値は数値で、以下の定数と比較できます。

- GEOIP_UNKNOWN_SPEED
- GEOIP_DIALUP_SPEED
- GEOIP_CABLEDSL_SPEED
- GEOIP_CORPORATE_SPEED

パラメータ

hostname

ネット接続のスピードを探す対象となるホスト名あるいは IP アドレス。

返り値

ネット接続のスピードを返します。

例

Example#1 geoip_id_by_name() の例

これは、ホスト example.com のネット接続のスピードを返します。

```
<?php
$netspeed = geoip_id_by_name('www.example.com');

echo 'The connection type is ';

switch ($netspeed) {
    case GEOIP_DIALUP_SPEED:
        echo 'dial-up';
        break;
    case GEOIP_CABLEDSL_SPEED:
        echo 'cable or DSL';
        break;
    case GEOIP_CORPORATE_SPEED:
        echo 'corporate';
        break;
    case GEOIP_UNKNOWN_SPEED:
     default:
```

```
        echo 'unknown';
}
?>
```

上の例の出力は以下となります。

```
The connection type is corporate
```

---

geoip_isp_by_name

(No version information available, might be only in CVS)

geoip_isp_by_name — インターネットサービスプロバイダ (ISP) 名を取得する

説明

string **geoip_isp_by_name** (string $hostname)

geoip_isp_by_name() 関数は、 IP アドレスに対応するインターネットサービスプロバイダ (ISP) 名を返します。

現在この関数を使用できるのは、商用の GeoIP ISP Edition を購入した人だけです。 適切なデータベースが見つからない場合には警告が発生します。

パラメータ

hostname

ホスト名あるいは IP アドレス。

返り値

成功した場合には ISP 名、 アドレスがデータベースで見つからない場合には **FALSE** を返します。

例

Example#1 geoip_isp_by_name() の例

これは、ホスト example.com の ISP 名を表示します。

```
<?php
$isp = geoip_isp_by_name('www.example.com');
if ($isp) {
    echo 'This host IP is from ISP: ' . $isp;
}
?>
```

上の例の出力は以下となります。

```
This host IP is allocated to: ICANN c/o Internet Assigned Numbers Authority
```

---

geoip_org_by_name

(PECL geoip:0.2.0-1.0.1)

geoip_org_by_name — 組織名を取得する

説明

string **geoip_org_by_name** (string $hostname)

geoip_org_by_name() 関数は、 その IP アドレスが割り当てられている組織の名前を返します。

現在この関数を使用できるのは、商用の GeoIP Organization, ISP あるいは AS Edition を購入した人だけです。 適切なデータベースが見つからない場合には警告が発生します。

パラメータ

hostname

ホスト名あるいは IP アドレス。

返り値

成功した場合には組織名、 アドレスがデータベースで見つからない場合には **FALSE** を返します。

例

Example#1 geoip_org_by_name() の例

これは、ホスト example.com の IP が誰に割り当てられているかを表示します。

```
<?php
$org = geoip_country_code_by_name('www.example.com');
if ($org) {
    echo 'This host IP is allocated to: ' . $org;
}
?>
```

上の例の出力は以下となります。

```
This host IP is allocated to: ICANN c/o Internet Assigned Numbers Authority
```

---

geoip_record_by_name

(PECL geoip:0.2.0-1.0.1)

geoip_record_by_name — GeoIP データベースで見つかった詳細な都市情報を返す

説明

array **geoip_record_by_name** (string $hostname)

geoip_record_by_name() 関数は、 ホスト名あるいは IP アドレスに対応するレコード情報を返します。

この関数は、GeoLite City Edition および商用の GeoIP City Edition のどちらでも使用可能です。 適切なデータベースが見つからない場合には警告が発生します。

返される連想配列には、以下のようなさまざまな名前のキーが含まれます。

- "country_code" -- 二文字の国コード (geoip_country_code_by_name() を参照ください)
- "region" -- 地域コード (例: カリフォルニアなら CA)
- "city" -- 市
- "postal_code" -- 郵便番号、FSA あるいは Zip コード
- "latitude" -- 緯度 (符号付き浮動小数点形式)
- "longitude" -- 経度 (符号付浮動小数点形式)
- "dma_code"
- "area_code" -- PSTN エリアコード (例: 212)

パラメータ

hostname

レコードを探す対象となるホスト名あるいは IP アドレス。

返り値

成功した場合には連想配列、 アドレスがデータベースで見つからない場合には **FALSE** を返します。

例

Example#1 geoip_record_by_name() の例

これは、ホスト example.com のレコードを含む連想配列を表示します。

```
<?php
$record = geoip_record_by_name('www.example.com');
if ($record) {
    print_r($record);
}
?>
```

上の例の出力は以下となります。

```
Array
(
      [country_code] => US
     [region] => CA
     [city] => Marina Del Rey
     [postal_code] =>
      [latitude] => 33.9776992798
      [longitude] => -118.435096741
     [dma_code] => 803
     [area_code] => 310
)
```

---

geoip_region_by_name

(PECL geoip:0.2.0-1.0.1)

geoip_region_by_name — 国コードおよび地域を取得する

説明

array **geoip_region_by_name** (string $hostname)

geoip_region_by_name() 関数は、 ホスト名あるいは IP アドレスに対応する国および地域を返します。

現在この関数を使用できるのは、商用の GeoIP Region Edition を購入した人だけです。 適切なデータベースが見つからない場合には警告が発生します。

返される連想配列には、以下のようなさまざまな名前のキーが含まれます。

- "country_code" -- 二文字の国コード (geoip_country_code_by_name() を参照ください)
- "region" -- 地域コード (例: カリフォルニアなら CA)

パラメータ

hostname

地域を探す対象となるホスト名あるいは IP アドレス。

返り値

成功した場合には連想配列、 アドレスがデータベースで見つからない場合には **FALSE** を返します。

例

Example#1 geoip_region_by_name() の例

これは、ホスト example.com の国コードおよび地域を含む配列を表示します。

```
<?php
$region = geoip_region_by_name('www.example.com');
if ($region) {
    print_r($region);
}
?>
```

上の例の出力は以下となります。

```
Array
(
      [country_code] => US
     [region] => CA
)
```

目次

- geoip_database_info — GeoIP データベースの情報を取得する
- geoip_db_avail — GeoIP データベースが使用可能かどうかを調べる
- geoip_db_filename — 対応する GeoIP データベースのファイル名を返す
- geoip_db_get_all_info — すべての GeoIP データベース形式についての詳細情報を返す
- geoip_id_by_name — インターネット接続のスピードを取得する
- geoip_isp_by_name — インターネットサービスプロバイダ (ISP) 名を取得する
- geoip_org_by_name — 組織名を取得する
- geoip_record_by_name — GeoIP データベースで見つかった詳細な都市情報を返す
- geoip_region_by_name — 国コードおよび地域を取得する

Gettext 関数

導入

gettext 関数は、NLS (Native Language Support) API を実装するもので、 PHP アプリケーションを国際化する際に使用することが可能です。 これらの関数の詳細については、システムの gettext ドキュメントを 参照するか、 » http://www.gnu.org/software/gettext/manual/gettext.html のドキュメントをごらんください。

要件

以下の関数を使用するには、» http://www.gnu.org/software/gettext/gettext.html から GNU gettext パッケージをダウンロードし、インストールする必要があります。

インストール手順

PHP で GNU gettext サポートを有効にするには、オプション --with-gettext[=DIR] を追加する必要が あります。ただし、DIR は gettext をインストールするディレクトリで、 デフォルトは */usr/local* です。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

bind_textdomain_codeset

(PHP 4 >= 4.2.0, PHP 5)

bind_textdomain_codeset — DOMAIN メッセージカタログから返されるメッセージの文字エンコーディングを指定する

説明

string **bind_textdomain_codeset** (string `$domain` , string `$codeset`)

bind_textdomain_codeset() を使用することで、 gettext() やその類似の関数において返される、 *domain* からのメッセージの文字エンコーディングを 設定することが可能です。

パラメータ

domain

ドメイン。

codeset

コードセット。

返り値

成功した場合に文字列を返します。

---

bindtextdomain

(PHP 4, PHP 5)

bindtextdomain — ドメインのパスを設定する

説明

string **bindtextdomain** (string `$domain` , string `$directory`)

bindtextdomain() 関数は、ドメインへのパスを設定します。

パラメータ

domain

ドメイン。

directory

ディレクトリのパス。

返り値

現在設定されているドメインへのフルパスを返します。

例

Example#1 bindtextdomain() の例

```
<?php

$domain = 'myapp';
echo bindtextdomain($domain, '/usr/share/myapp/locale');

?>
```

上の例の出力は以下となります。

```
/usr/share/myapp/locale
```

---

dcgettext

(PHP 4, PHP 5)

dcgettext — 単一の参照に関するドメインを上書きする

説明

string **dcgettext** (string `$domain` , string `$message` , int `$category`)

この関数により単一の参照についてカレントのドメインを上書きすることができます。

パラメータ

domain

ドメイン。

message

メッセージ。

category

カテゴリ。

返り値

成功した場合に文字列を返します。

参考

- gettext()

---

dcngettext

(PHP 4 >= 4.2.0, PHP 5)

dcngettext — dcgettext の複数形版

説明

string **dcngettext** (string `$domain` , string `$msgid1` , string `$msgid2` , int `$n` , int `$category`)

この関数により単一の複数形メッセージの参照について カレントのドメインを上書きすることができます。

パラメータ

`domain`

ドメイン。

`msgid1`

`msgid2`

`n`

`category`

返り値

成功した場合に文字列を返します。

参考

- ngettext()

---

dgettext

(PHP 4, PHP 5)

dgettext — 現在のドメインを上書きする

説明

string **dgettext** (string `$domain` , string `$message`)

dgettext() 関数により、単一のメッセージ参照について 現在のドメインを上書きすることができます。

パラメータ

`domain`

ドメイン。

`message`

メッセージ。

返り値

成功した場合に文字列を返します。

参考

- gettext()

---

dngettext

(PHP 4 >= 4.2.0, PHP 5)

dngettext — dgettext の複数形版

説明

string **dngettext** (string `$domain` , string `$msgid1` , string `$msgid2` , int `$n`)

dngettext() 関数により、単一の複数形メッセージ参照について 現在のドメインを上書きすることができます。

パラメータ

`domain`

ドメイン。

`msgid1`

`msgid2`

`n`

返り値

成功した場合に文字列を返します。

参考

- ngettext()

---

gettext

(PHP 4, PHP 5, PECL axis2:0.1.0-0.1.1)

gettext — 現在のドメインのメッセージを参照する

説明

string **gettext** (string `$message`)

現在のドメインのメッセージを参照します。

パラメータ

`message`

返り値

翻訳テーブルに翻訳文字列が見つかった場合にその文字列、 あるいは見つからなかった場合に元の文字列を返します。

例

Example#1 gettext() のチェック

```
<?php
// ドイツ語に設定します
setlocale(LC_ALL, 'de_DE');

// 変換テーブルの場所を指定します
bindtextdomain("myPHPApp", "./locale");
```

```
// ドメインを選択します
textdomain("myPHPApp");

//  翻訳内容は  ./locale/de_DE/LC_MESSAGES/myPHPApp.mo  から検索されます

// テストメッセージを出力します
echo gettext("Welcome to My PHP Application");

// あるいは、gettext() のかわりに _() も使用可能です
echo _("Have a nice day");
?>
```

注意

注意: この関数のエイリアスとして、アンダースコア文字 '_' を使用することができます。

参考

- setlocale()

---

ngettext

(PHP 4 >= 4.2.0, PHP 5)

ngettext — gettext の複数形版

説明

string **ngettext** (string `$msgid1` , string `$msgid2` , int `$n`)

複数形版の gettext() です。 言語によっては、数量に応じていくつかの複数形が存在することがあります。

パラメータ

msgid1

msgid2

n

返り値

msgid1 および *msgid2* で表されるメッセージの、数 *n* に対応する複数形を返します。

例

Example#1 ngettext() の例

```
<?php

setlocale(LC_ALL, 'cs_CZ');
printf(ngettext("%d window", "%d windows", 1), 1); // 1 okno
printf(ngettext("%d window", "%d windows", 2), 2); // 2 okna
printf(ngettext("%d window", "%d windows", 5), 5); // 5 oken

?>
```

---

textdomain

(PHP 4, PHP 5)

textdomain — デフォルトドメインを設定する

説明

string **textdomain** (string `$text_domain`)

この関数は、gettext() がコールされた際に検索を 行うドメインを設定します。このドメインは、通常はアプリケーション名から 付けられます。

パラメータ

text_domain

新しいメッセージドメイン。**NULL** を指定すると、現在の設定をそのまま 取得し、変更しません。

返り値

成功すると、この関数はドメインを変更した後に現在のメッセージドメインを 返します。

目次

GMP 関数

導入

以下の関数により、GNU MP ライブラリを使用して 任意長の整数を使用することが可能になります。

これらの関数は、PHP 4.0.4 で追加されました。

> **注意**: 多くの GMP 関数は、*resource* で定義された GMP 数を 引数としてとります。しかし、これらの関数の多くは、数値と文字列の 両方を引数として指定可能で、後者は数値に変換することが可能です。 また、整数引数を使用して処理を行うより高速な関数がある場合には、 指定された引数が整数である場合により低速となる関数の代わりに 使用されます。これは透過的に行われるため、結果的に、GMP 数値を 引数とする全ての関数について整数を使用することが可能です。関数 gmp_init() も参照ください。

警告

より大きな整数を明示的に指定するには、文字列として指定してください。 そうしない場合、PHP は値ををまず整数リテラルとして解釈し、 *GMP* にわたるまでに精度の劣化を生じる可能性が あります。

> **注意**: PHP 5.1.0 以降、この拡張モジュールは Windows でも使用可能です。

要件

» http://www.swox.com/gmp/ から GMP ライブラリをダウンロード可能です。 このサイトでは、GMP のマニュアルも入手可能です。

これらの関数を使用するには、GMP バージョン 2 以降が必要です。 中には、より新しいバージョンの GMP ライブラリを必要とする関数も あります。

インストール手順

これらの関数を利用可能とするには、オプション --with-gmp を使用することにより GMP サポートを有効にして PHP をコンパイルする必要が あります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

ほとんどの GPM 関数は、GMP 数のリソースを使用するか、それを返します。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

GMP_ROUND_ZERO (integer)
GMP_ROUND_PLUSINF (integer)
GMP_ROUND_MINUSINF (integer)
GMP_VERSION (string)
GMP ライブラリのバージョン。

例

Example#1 GMP を使用した階乗関数

```
<?php
function fact($x)
{
    $return = 1;
    for ($i=2; $i < $x; $i++) {
        $return = gmp_mul($return, $i);
    }
    return $return;
}

echo gmp_strval(fact(1000)) . "¥n";
?>
```

この例は、1000 の階乗(非常に大きな数です)を非常に高速に計算します。

参考

より数学的な関数が、 BCMath 任意精度関数 および 数学関数 の節にあります。

---

gmp_abs

(PHP 4 >= 4.0.4, PHP 5)

gmp_abs — 絶対値

説明

resource **gmp_abs** (resource $a)

ある数の絶対値を返します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a の絶対値を GMP 数で返します。

例

Example#1 gmp_abs() の例

```
<?php
$abs1 = gmp_abs("274982683358");
$abs2 = gmp_abs("-274982683358");

echo gmp_strval($abs1) . "¥n";
echo gmp_strval($abs2) . "¥n";
?>
```

上の例の出力は以下となります。

```
274982683358
274982683358
```

---

gmp_add

(PHP 4 >= 4.0.4, PHP 5)

gmp_add — 数値を加算する

説明

resource **gmp_add** (resource $a , resource $b)

2 つの数を加算します。

パラメータ

a

足される数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

足す数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

加算の結果を GMP 数で返します。

例

Example#1 gmp_add() の例

```
<?php
$sum = gmp_add("123456789012345", "76543210987655");
echo gmp_strval($sum) . "¥n";
?>
```

上の例の出力は以下となります。

```
200000000000000
```

---

gmp_and

(PHP 4 >= 4.0.4, PHP 5)

gmp_and — 論理積を計算する

説明

resource **gmp_and** (resource $a , resource $b)

2 つの GMP 数の論理積を計算します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

論理 *AND* 演算の結果を GMP 数で返します。

例

Example#1 gmp_and() の例

```php
<?php
$and1 = gmp_and("0xfffffffff4", "0x4");
$and2 = gmp_and("0xfffffffff4", "0x8");
echo gmp_strval($and1) . "¥n";
echo gmp_strval($and2) . "¥n";
?>
```

上の例の出力は以下となります。

```
4
0
```

---

gmp_clrbit

(PHP 4 >= 4.0.4, PHP 5)

gmp_clrbit — ビットをクリアする

説明

void **gmp_clrbit** (resource &$a , int $index)

a のビット *index* をクリア (0 に設定) します。index は 0 から始まります。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

index

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_clrbit() の例

```php
<?php
$a = gmp_init("0xff");
gmp_clrbit($a, 0); // インデックスは 0 から始まり、これは最下位ビットを表します
echo gmp_strval($a) . "¥n";
?>
```

上の例の出力は以下となります。

```
254
```

注意

注意: 他の大半の GMP 関数とは異なり、**gmp_clrbit()** は必ず既存の GMP リソース (例えば gmp_init() を使用して取得したもの) を使用してコールしなければなりません。 リソースは自動的には作成されません。

参考

- gmp_setbit()
- gmp_testbit()

---

gmp_cmp

(PHP 4 >= 4.0.4, PHP 5)

gmp_cmp — 数を比較する

説明

int **gmp_cmp** (resource $a , resource $b)

ふたつの数を比較します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a > *b* の場合に正の値、*a* = *b* の場合にゼロ、*a* < *b* の場合に負の値を返します。

例

Example#1 gmp_cmp() の例

```
<?php
$cmp1 = gmp_cmp("1234", "1000"); //  より大きい
$cmp2 = gmp_cmp("1000", "1234"); //  より小さい
$cmp3 = gmp_cmp("1234", "1234"); //  等しい

echo "$cmp1  $cmp2  $cmp3¥n";
?>
```

上の例の出力は以下となります。

```
1 -1 0
```

---

gmp_com

(PHP 4 >= 4.0.4, PHP 5)

gmp_com — 1 の補数を計算する

説明

resource **gmp_com** (resource $a)

a について、1 の補数を返します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a についての 1 の補数を GMP 数で返します。

例

Example#1 gmp_com() の例

```
<?php
$com = gmp_com("1234");
echo gmp_strval($com) . "¥n";
?>
```

上の例の出力は以下となります。

```
-1235
```

gmp_div_q

(PHP 4 >= 4.0.4, PHP 5)

gmp_div_q — 数値を除算する

説明

resource **gmp_div_q** (resource $a , resource $b [, int $round])

a を *b* で割り、 結果を整数で返します。

パラメータ

a

割られる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

a を割る数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

round

結果の丸め方は *round* で指定し、次の値を指定可能です。

- *GMP_ROUND_ZERO*：結果は 0 の方に丸められます。
- *GMP_ROUND_PLUSINF*：結果は、 *+infinity* の方に丸められます。
- *GMP_ROUND_MINUSINF*：結果は、 *-infinity* の方に丸められます。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_div_q() の例

```
<?php
$div1 = gmp_div_q("100", "5");
echo gmp_strval($div1) . "¥n";

$div2 = gmp_div_q("1", "3");
echo gmp_strval($div2) . "¥n";

$div3 = gmp_div_q("1", "3", GMP_ROUND_PLUSINF);
echo gmp_strval($div3) . "¥n";

$div4 = gmp_div_q("-1", "4", GMP_ROUND_PLUSINF);
echo gmp_strval($div4) . "¥n";

$div5 = gmp_div_q("-1", "4", GMP_ROUND_MINUSINF);
echo gmp_strval($div5) . "¥n";
?>
```

上の例の出力は以下となります。

```
20
0
1
0
-1
```

注意

注意: この関数は、gmp_div() のようにコールすることも可能です。

参考

- gmp_div_r()
- gmp_div_qr()

---

gmp_div_qr

(PHP 4 >= 4.0.4, PHP 5)

gmp_div_qr — 除算を行い、商と余りを得る

説明

array **gmp_div_qr** (resource $n , resource $d [, int $round])

この関数は、*n* を *d* で割ります。

パラメータ

n

割られる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d

n を割る数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

round

引数 *round* の説明については、 gmp_div_q() 関数を参照ください。

返り値

配列を返します。配列の最初の要素は *[n/d]* (割算の結果の整数値)、2 番目の要素は *(n - [n/d] * d)* (割算の余り) です。

例

Example#1 GMP 数の割算

```
<?php
$a = gmp_init("0x41682179fbf5");
$res = gmp_div_qr($a, "0xDEFE75");
printf("Result is: q - %s, r - %s",
       gmp_strval($res[0]), gmp_strval($res[1]));
?>
```

参考

- gmp_div_q()
- gmp_div_r()

---

gmp_div_r

(PHP 4 >= 4.0.4, PHP 5)

gmp_div_r — 除算の余りを計算する

説明

resource **gmp_div_r** (resource $n , resource $d [, int $round])

n を *d* で整数として割った際の余りを計算します。余りは、引数 *n* がゼロでないばあいに、これと同じ符号を有します。

パラメータ

n

割られる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d

n を割る数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

round

引数 *round* の説明については、関数 gmp_div_q() を参照ください。

返り値

余りを GMP 数で返します。

例

Example#1 gmp_div_r() の例

```
<?php
$div = gmp_div_r("105", "20");
echo gmp_strval($div) . "¥n";
?>
```

上の例の出力は以下となります。

```
5
```

参考

- gmp_div_q()
- gmp_div_qr()

---

gmp_div

(PHP 4 >= 4.0.4, PHP 5)

gmp_div — gmp_div_q() のエイリアス

説明

この関数は次の関数のエイリアスです。 gmp_div_q().

---

gmp_divexact

(PHP 4 >= 4.0.4, PHP 5)

gmp_divexact — 正確な除算

説明

resource **gmp_divexact** (resource $n , resource $d)

高速な "exact division" アルゴリズムを使用して *n* を *d* で割ります。 この関数は、*n* が *d* で割り切れることがわかっている場合にのみ正確な結果を出力します。

パラメータ

n

割られる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d

a を割る数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_divexact() の例

```
<?php
$div1 = gmp_divexact("10", "2");
echo gmp_strval($div1) . "¥n";

$div2 = gmp_divexact("10", "3"); //  間違った結果となります
echo gmp_strval($div2) . "¥n";
?>
```

上の例の出力は以下となります。

```
5
2863311534
```

---

gmp_fact

(PHP 4 >= 4.0.4, PHP 5)

gmp_fact — 階乗

説明

resource **gmp_fact** (int $a)

a の階乗 (*a!*) を計算します。

パラメータ

a

階乗を求める数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_fact() の例

```
<?php
$fact1 = gmp_fact(5); // 5 * 4 * 3 * 2 * 1
echo gmp_strval($fact1) . "¥n";

$fact2 = gmp_fact(50); // 50 * 49 * 48, ... etc
echo gmp_strval($fact2) . "¥n";
?>
```

上の例の出力は以下となります。

```
120
30414093201713378043612608166064768844377641568960512000000000000
```

---

gmp_gcd

(PHP 4 >= 4.0.4, PHP 5)

gmp_gcd — 最大公約数を計算する

説明

resource **gmp_gcd** (resource $a , resource $b)

a と *b* の最大公約数を計算します。 引数のどちらかまたは両方が負の場合でも結果は常に正となります。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a と *b* の両方を割り切ることができる正の数を GMP 数で返します。

例

Example#1 gmp_gcd() の例

```
<?php
$gcd = gmp_gcd("12", "21");
echo gmp_strval($gcd) . "¥n";
?>
```

上の例の出力は以下となります。

```
3
```

---

gmp_gcdext

(PHP 4 >= 4.0.4, PHP 5)

gmp_gcdext — 最大公約数と乗数を計算する

説明

array **gmp_gcdext** (resource $a , resource $b)

*a*s + *b*t = *g* = *gcd(a,b)* となるような g, s, t を計算します。ただし、gcd は最大公約数です。g, s, t を要素とする配列を返します。

この関数は、2 変数の線形不定方程式 (linear Diophantine equations) を解く際に使用することが可能です。 この方程式は、*a*x + *b*y = *c* のような形式で、 整数のみを解とするものです。 詳細な情報は、» MathWorld の "Diophantine Equation" についてのページを参照ください。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

GMP 数の配列を返します。

例

Example#1 線形不定方程式を解く

```php
<?php
// 方程式 a*s + b*t = g を解きます。
// a = 12, b = 21, g = gcd(12, 21) = 3 とします。
$a = gmp_init(12);
$b = gmp_init(21);
$g = gmp_gcd($a, $b);
$r = gmp_gcdext($a, $b);

$check_gcd  = (gmp_strval($g) == gmp_strval($r['g']));
$eq_res  = gmp_add(gmp_mul($a, $r['s']), gmp_mul($b, $r['t']));
$check_res  = (gmp_strval($g) == gmp_strval($eq_res));

if ($check_gcd  && $check_res) {
    $fmt = "Solution: %d*%d + %d*%d = %d¥n";
    printf($fmt, gmp_strval($a), gmp_strval($r['s']), gmp_strval($b),
    gmp_strval($r['t']), gmp_strval($r['g']));
} else {
    echo "方程式を解く際にエラーが発生しました¥n";
}

// 出力は、解: 12*2 + 21*-1 = 3 のようになります。
?>
```

---

gmp_hamdist

(PHP 4 >= 4.0.4, PHP 5)

gmp_hamdist — ハミング距離

説明

int **gmp_hamdist** (resource $a , resource $b)

a と *b* の間のハミング距離を返します。オペランドは共に非負とする必要があります。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

正の数である必要があります。

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

正の数である必要があります。

返り値

A GMP number resource.

例

Example#1 gmp_hamdist() の例

```php
<?php
$ham1 = gmp_init("1001010011", 2);
$ham2 = gmp_init("1011111100", 2);
echo gmp_hamdist($ham1, $ham2) . "¥n";

/* ハミング距離は、以下と等しくなります */
echo gmp_popcount(gmp_xor($ham1, $ham2)) . "¥n";
?>
```

上の例の出力は以下となります。

```
6
6
```

参考

- gmp_popcount()
- gmp_xor()

---

gmp_init

(PHP 4 >= 4.0.4, PHP 5)

gmp_init — GMP 数を作成する

説明

resource **gmp_init** (mixed $number [, int $base])

整数または文字列から GMP 数を生成します。

パラメータ

number

整数値あるいは文字列。文字列表現には、 十進数か十六進数、あるいは八進数を使用可能です。

base

基数。デフォルトは 0。

基数には 2 から 36 までの値を指定することができます。 基数を 0 (デフォルト値) にすると、最初の文字に応じて実際の基数を決定します。 最初の二文字が *0x* あるいは *0X* の場合は十六進数、それ以外で最初の文字が "0" の場合は八進数、 それ以外の場合は十進数となります。

返り値

A GMP number resource.

変更履歴

バージョン	説明
4.1.0	オプションのパラメータ *base* が追加されました。

例

Example#1 GMP 数の作成

```
<?php
$a = gmp_init(123456);
$b = gmp_init("0xFFFFDEBACDFEDF7200");
?>
```

注意

注意: gmp_add() のような GMP 関数において、GMP 数を指定するところに整数または文字列を使用したい場合には、 この関数をコールする必要はありません。この場合、変換が必要な場合には、 **gmp_init()** と同様の方法で関数の引数は自動的に GMP 数に変換されます。

---

gmp_intval

(PHP 4 >= 4.0.4, PHP 5)

gmp_intval — GMP 数を整数に変換する

説明

int **gmp_intval** (resource $gmpnumber)

この関数により、GMP 数を整数に変換することが可能になります。

パラメータ

gmpnumber

GMP 数。

返り値

gmpnumber を整数に変換した結果を返します。

例

Example#1 gmp_intval() の例

```
<?php
// 正しい結果を表示します
echo gmp_intval("2147483647") . "¥n";

// PHP の整数型の制限をこえているので、正しい結果を表示しません
echo gmp_intval("2147483648") . "¥n";

// 正しい結果を表示します
echo gmp_strval("2147483648") . "¥n";
?>
```

上の例の出力は以下となります。

```
2147483647
2147483647
2147483648
```

注意

警告

この関数は、数値が実際に PHP の整数型に適合する場合(すなわち、 符号付き long 型)にのみ有用な結果を返します。単に GMP 数を出力したい場合には、gmp_strval() を使用してください。

---

gmp_invert

(PHP 4 >= 4.0.4, PHP 5)

gmp_invert — 法による逆

説明

resource **gmp_invert** (resource $*a* , resource $*b*)

b を法とした *a* の逆を計算します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

成功した場合に GMP 数、逆が存在しない場合に **FALSE** を返します。

例

Example#1 gmp_invert() の例

```
<?php
echo gmp_invert("5", "10"); //  逆は存在しないので何も出力せず、結果は  FALSE  となります
$invert  = gmp_invert("5", "11");
echo gmp_strval($invert) . "¥n";
?>
```

上の例の出力は以下となります。

```
9
```

---

gmp_jacobi

(PHP 4 >= 4.0.4, PHP 5)

gmp_jacobi — ヤコビ記号

説明

int **gmp_jacobi** (resource $a , resource $p)

a および *p* の » ヤコビ記号 を計算します。*p* は正の奇数である必要があります。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

p

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

正の奇数でなければなりません。

返り値

A GMP number resource.

例

Example#1 gmp_jacobi() の例

```
<?php
echo gmp_jacobi("1", "3") . "¥n";
echo gmp_jacobi("2", "3") . "¥n";
?>
```

上の例の出力は以下となります。

```
1
0
```

---

gmp_legendre

(PHP 4 >= 4.0.4, PHP 5)

gmp_legendre — ルジェンドル記号

説明

int **gmp_legendre** (resource $a , resource $p)

a と *p* の » ルジェンドル記号 を計算します。*p* は、正の奇数である必要があります。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

p

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

正の奇数でなければなりません。

返り値

A GMP number resource.

例

Example#1 gmp_legendre() の例

```
<?php
echo gmp_legendre("1", "3") . "¥n";
echo gmp_legendre("2", "3") . "¥n";
?>
```

上の例の出力は以下となります。

```
1
0
```

---

gmp_mod

(PHP 4 >= 4.0.4, PHP 5)

gmp_mod — モジュロ演算

説明

resource **gmp_mod** (resource $n , resource $d)

d を法として *n* を計算します。結果は常に非負であり、 *d* の符号は無視されます。

パラメータ

n

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

d

法として使用する値。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_mod() の例

```
<?php
$mod = gmp_mod("8", "3");
echo gmp_strval($mod) . "¥n";
?>
```

上の例の出力は以下となります。

```
2
```

---

gmp_mul

(PHP 4 >= 4.0.4, PHP 5)

gmp_mul — 数値を乗算する

説明

resource **gmp_mul** (resource $a , resource $b)

a と *b* をかけ、 結果を返します。

パラメータ

a

b を掛けられる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

a に掛ける数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_mul() の例

```
<?php
$mul = gmp_mul("12345678", "2000");
echo gmp_strval($mul) . "¥n";
?>
```

上の例の出力は以下となります。

```
24691356000
```

---

gmp_neg

(PHP 4 >= 4.0.4, PHP 5)

gmp_neg — 符号を反転する

説明

resource **gmp_neg** (resource $*a*)

ある数の符号を反転したものを返します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

-a を GMP 数で返します。

例

Example#1 gmp_neg() の例

```
<?php
$neg1 = gmp_neg("1");
echo gmp_strval($neg1) . "¥n";
$neg2 = gmp_neg("-1");
echo gmp_strval($neg2) . "¥n";
?>
```

上の例の出力は以下となります。

```
-1
1
```

---

gmp_nextprime

(PHP 5 >= 5.2.0)

gmp_nextprime — 次の素数を見つける

説明

resource **gmp_nextprime** (int $*a*)

次の素数を見つけます。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a より大きい次の素数を GMP 数として返します。

例

Example#1 gmp_nextprime() の例

```
<?php
$prime1 = gmp_nextprime(10); // 10 より大きい次の素数
$prime2 = gmp_nextprime(-1000); // -1000 より大きい次の素数

echo gmp_strval($prime1) . "¥n";
echo gmp_strval($prime2) . "¥n";
?>
```

上の例の出力は以下となります。

```
11
-997
```

注意

注意: この関数は素数を識別するのに確率的アルゴリズムを使用します。 誤って合成数を取得してしまうことは、まずありません。

---

gmp_or

(PHP 4 >= 4.0.4, PHP 5)

gmp_or — 論理和を計算する

説明

resource **gmp_or** (resource $*a* , resource $*b*)

2 つの GMP 数の論理和を計算します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_or() の例

```php
<?php
$or1 = gmp_or("0xfffffff2", "4");
echo gmp_strval($or1, 16) . "¥n";
$or2 = gmp_or("0xfffffff2", "2");
echo gmp_strval($or2, 16) . "¥n";
?>
```

上の例の出力は以下となります。

```
fffffff6
fffffff2
```

---

gmp_perfect_square

(PHP 4 >= 4.0.4, PHP 5)

gmp_perfect_square — 平方数かどうかを調べる

説明

bool **gmp_perfect_square** (resource $a)

その数が平方数であるかどうかを調べます。

パラメータ

a

平方数かどうかを調べたい数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a が平方数の場合に **TRUE**、その他の場合に **FALSE** を返します。

例

Example#1 gmp_perfect_square() の例

```php
<?php
// 3 * 3、平方数です
var_dump(gmp_perfect_square("9"));

// 平方数ではありません
var_dump(gmp_perfect_square("7"));

// 1234567890 * 1234567890、平方数です
var_dump(gmp_perfect_square("1524157875019052100"));
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(false)
bool(true)
```

参考

- gmp_sqrt()
- gmp_sqrtrem()

---

gmp_popcount

(PHP 4 >= 4.0.4, PHP 5)

gmp_popcount — セットされているビットの数

説明

int **gmp_popcount** (resource $a)

セットされているビットの数を返します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a にセットされているビットの数を返します。

例

Example#1 gmp_popcount() の例

```
<?php
$pop1 = gmp_init("10000101", 2); // 1 が 3 つ
echo gmp_popcount($pop1) . "¥n";
$pop2 = gmp_init("11111110", 2); // 1 が 7 つ
echo gmp_popcount($pop2) . "¥n";
?>
```

上の例の出力は以下となります。

```
3
7
```

---

gmp_pow

(PHP 4 >= 4.0.4, PHP 5)

gmp_pow — べき乗を計算する

説明

resource **gmp_pow** (resource $base , int $exp)

base の *exp* 乗を計算します。

パラメータ

base

もととなる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

exp

正の数で、*base* を何乗するかを指定します。

返り値

べき乗の結果を GMP 数で返します。 *0^0* は 1 となります。

例

Example#1 gmp_pow() の例

```
<?php
$pow1 = gmp_pow("2", 31);
echo gmp_strval($pow1) . "¥n";
$pow2 = gmp_pow("0", 0);
echo gmp_strval($pow2) . "¥n";
$pow3 = gmp_pow("2", -1); //  負のべき乗を指定したため、警告を発生します
echo gmp_strval($pow3) . "¥n";
?>
```

上の例の出力は以下となります。

```
2147483648
```

1

---

gmp_powm

(PHP 4 >= 4.0.4, PHP 5)

gmp_powm — べき乗とモジュロを計算する

説明

resource **gmp_powm** (resource $base , resource $exp , resource $mod)

mod を法として (*base* の *exp* 乗) を計算します。 *exp* が負の場合、結果は未定義(undefined) となります。

パラメータ

base

もととなる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

exp

正の数で、*base* を何乗するかを指定します。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

mod

モジュロ。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

結果を GMP 数で返します。

例

Example#1 gmp_powm() の例

```
<?php
$pow1 = gmp_powm("2", "31", "2147483649");
echo gmp_strval($pow1) . "¥n";
?>
```

上の例の出力は以下となります。

```
2147483648
```

---

gmp_prob_prime

(PHP 4 >= 4.0.4, PHP 5)

gmp_prob_prime — 数が"ほぼ素数"であるかどうかを調べる

説明

int **gmp_prob_prime** (resource $a [, int $reps])

この関数は、Miller-Rabin の予測テストを使用して、 その数が素数かどうかを調べます。

パラメータ

a

素数かどうかを調べたい数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

reps

reps の値 (デフォルトは 10) は、5 から 10 までです。より大きい値を指定すると、素数でない数を 「おそらく素数である」と誤認識する可能性が小さくなります。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

この関数が 0 を返す場合、*a* は確実に素数ではありません。 1 を返す場合、*a* は「おそらく」 素数です。2 を返す場合、*a* は確実に素数です。

例

Example#1 gmp_prob_prime() の例

```
<?php
// 明らかに素数ではありません
echo gmp_prob_prime("6") . "¥n";

// おそらく素数です
echo gmp_prob_prime("1111111111111111111") . "¥n";

// 明らかに素数です
echo gmp_prob_prime("11") . "¥n";
?>
```

上の例の出力は以下となります。

```
0
1
2
```

---

gmp_random

(PHP 4 >= 4.0.4, PHP 5)

gmp_random — 乱数を生成する

説明

resource **gmp_random** (int $limiter)

乱数を生成します。乱数の範囲は、ゼロから *limiter* * limb のビット数となります。 *limiter* が負の場合、 負の数が生成されます。

limb は GMP の内部機構です。limb のビット数は固定ではなく、 システムによって変化します。一般的には limb は 16 あるいは 32 ビットですが、それが保証されているわけではありません。

パラメータ

limiter

リミッタ。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

ランダムな GMP 数を返します。

例

Example#1 gmp_random() の例

```
<?php
$rand1 = gmp_random(1); // 0 から 1 * limb のビット数までの間の乱数
$rand2 = gmp_random(2); // 0 から 2 * limb のビット数までの間の乱数

echo gmp_strval($rand1) . "¥n";
echo gmp_strval($rand2) . "¥n";
?>
```

上の例の出力は以下となります。

```
1915834968
86425640758903280 87
```

---

gmp_scan0

(PHP 4 >= 4.0.4, PHP 5)

gmp_scan0 — 0 を探す

説明

int **gmp_scan0** (resource $a , int $start)

ビット *start* から最上位ビットの方に、 最初のクリアビットが見付かるまで *a* をスキャンします。

パラメータ

a

スキャンする数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

start

開始するビット。

返り値

ビットが見つかった場所のインデックスを整数値で返します。 インデックスは 0 から始まります。

例

Example#1 gmp_scan0() の例

```php
<?php
// 0 番目から探しはじめ、3 番目の位置に "0" ビットが見つかります
$s1 = gmp_init("10111", 2);
echo gmp_scan0($s1, 0) . "¥n";

// 5 番目から探しはじめ、7 番目の位置に "0" ビットが見つかります
$s2 = gmp_init("101110000", 2);
echo gmp_scan0($s2, 5) . "¥n";
?>
```

上の例の出力は以下となります。

```
3
7
```

---

gmp_scan1

(PHP 4 >= 4.0.4, PHP 5)

gmp_scan1 — 1 を探す

説明

int **gmp_scan1** (resource $a , int $start)

ビット *start* から最上位ビットの方に、 セットされているビットが最初に見付かるまで *a* をスキャンします。

パラメータ

a

スキャンする数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

start

開始するビット。

返り値

ビットが見つかった場所のインデックスを整数値で返します。 セットされているビットが存在しない場合には -1 を返します。

例

Example#1 gmp_scan1() の例

```
<?php
// 0 番目から探しはじめ、3 番目の位置に "1" ビットが見つかります
$s1 = gmp_init("01000", 2);
echo gmp_scan1($s1, 0) . "¥n";

// 5 番目から探しはじめ、9 番目の位置に "1" ビットが見つかります
$s2 = gmp_init("01000001111", 2);
echo gmp_scan1($s2, 5) . "¥n";
?>
```

上の例の出力は以下となります。

```
3
9
```

---

gmp_setbit

(PHP 4 >= 4.0.4, PHP 5)

gmp_setbit — ビットを設定する

説明

void **gmp_setbit** (resource `&$a` , int `$index` [, bool `$set_clear`])

a のビット *index* を設定します。

パラメータ

a

ビットをセットしたい数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

index

設定するビット。

set_clear

そのビットを 0 または 1 のどちらにするかを定義します。デフォルトで、ビットは 1 に設定されます。

返り値

A GMP number resource.

例

Example#1 gmp_setbit() の例

```
<?php
$a = gmp_init("0xfd");
gmp_setbit($a, 1); // インデックスは 0 から始まります
echo gmp_strval($a) . "¥n";
?>
```

上の例の出力は以下となります。

```
255
```

注意

注意: 他の大半の GMP 関数とは異なり、**gmp_setbit()** は必ず既存の GMP リソース (例えば gmp_init() を使用して取得したもの) を使用してコールしなければなりません。 リソースは自動的には作成されません。

参考

- gmp_clrbit()
- gmp_testbit()

---

gmp_sign

(PHP 4 >= 4.0.4, PHP 5)

gmp_sign — 数の符号

説明

int **gmp_sign** (resource $a)

数の符号を調べます。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

a が正の場合に 1、 *a* が負の場合に -1、そして *a* がゼロの場合に 0 を返します。

例

Example#1 gmp_sign() の例

```php
<?php
// 正
echo gmp_sign("500") . "¥n";

// 負
echo gmp_sign("-500") . "¥n";

// ゼロ
echo gmp_sign("0") . "¥n";
?>
```

上の例の出力は以下となります。

```
1
-1
0
```

---

gmp_sqrt

(PHP 4 >= 4.0.4, PHP 5)

gmp_sqrt — 平方根を計算する

説明

resource **gmp_sqrt** (resource $a)

a の平方根を計算します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

平方根の整数部分を GMP 数で返します。

例

Example#1 gmp_sqrt() の例

```
<?php
$sqrt1 = gmp_sqrt("9");
$sqrt2 = gmp_sqrt("7");
$sqrt3 = gmp_sqrt("1524157875019052100");

echo gmp_strval($sqrt1) . "¥n";
echo gmp_strval($sqrt2) . "¥n";
echo gmp_strval($sqrt3) . "¥n";
?>
```

上の例の出力は以下となります。

```
3
2
1234567890
```

---

gmp_sqrtrem

(PHP 4 >= 4.0.4, PHP 5)

gmp_sqrtrem — 余りつきの平方根

説明

array **gmp_sqrtrem** (resource $*a*)

ある数の平方根を余りつきで計算します。

パラメータ

a

平方根を計算したい数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

最初の要素が *a* の整数平方根 (gmp_sqrt()も参照ください)、2 番目の要素が余り (すなわち、*a* と最初の要素の 2 乗の差) であるような配列を返します。

例

Example#1 gmp_sqrtrem() の例

```
<?php
list($sqrt1, $sqrt1rem) = gmp_sqrtrem("9");
list($sqrt2, $sqrt2rem) = gmp_sqrtrem("7");
list($sqrt3, $sqrt3rem) = gmp_sqrtrem("1048576");

echo gmp_strval($sqrt1) . ", " . gmp_strval($sqrt1rem) . "¥n";
echo gmp_strval($sqrt2) . ", " . gmp_strval($sqrt2rem) . "¥n";
echo gmp_strval($sqrt3) . ", " . gmp_strval($sqrt3rem) . "¥n";
?>
```

上の例の出力は以下となります。

```
3, 0
2, 3
1024, 0
```

gmp_strval

(PHP 4 >= 4.0.4, PHP 5)

gmp_strval — GMP 数を文字列に変換する

説明

string **gmp_strval** (resource $gmpnumber [, int $base])

GMP 数を *base* を基数とする文字列表現に変換します。 デフォルトの基数は 10 です。

パラメータ

gmpnumber

文字列の変換したい GMP 数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

base

返り値の基数。デフォルトは 10 です。 基数として使用可能な値は 2 から 36 までです。

返り値

数を文字列で表したものを返します。

例

Example#1 GMP 数を文字列に変換する

```
<?php
$a = gmp_init("0x41682179fbf5");
printf("10 進数: %s, 36 進数: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_sub

(PHP 4 >= 4.0.4, PHP 5)

gmp_sub — 数値の減算

説明

resource **gmp_sub** (resource $a , resource $b)

a から *b* を引いた結果を返します。

パラメータ

a

引かれる数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

a から引く数。

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_sub() の例

```
<?php
$sub = gmp_sub("281474976710656", "4294967296"); // 2^48 - 2^32
echo gmp_strval($sub) . "¥n";
?>
```

上の例の出力は以下となります。

```
281470681743360
```

---

gmp_testbit

(No version information available, might be only in CVS)

gmp_testbit — ビットが設定されているかどうかを調べる

説明

bool **gmp_testbit** (resource $a , int $index)

指定したビットが設定されているかどうかを調べます。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

index

調べたいビット。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

index に負の数を指定すると **E_WARNING** が発生します。

例

Example#1 gmp_testbit() の例

```
<?php
$n = gmp_init("1000000");
var_dump(gmp_testbit($n, 1));
gmp_setbit($n, 1);
var_dump(gmp_testbit($n, 1));
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(true)
```

参考

- gmp_setbit()
- gmp_clrbit()

---

gmp_xor

(PHP 4 >= 4.0.4, PHP 5)

gmp_xor — 排他的論理和を計算する

説明

resource **gmp_xor** (resource $a , resource $b)

2 つの GMP 数の排他的論理和 (XOR) を計算します。

パラメータ

a

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

b

It can be either a GMP number resource, or a numeric string given that it is possible to convert the latter to a number.

返り値

A GMP number resource.

例

Example#1 gmp_xor() の例

```
<?php
$xor1 = gmp_init("1101101110011101", 2);
$xor2 = gmp_init("0110011001011001", 2);

$xor3 = gmp_xor($xor1, $xor2);

echo gmp_strval($xor3, 2) . "¥n";
?>
```

上の例の出力は以下となります。

```
1011110111000100
```

目次

- gmp_pow — べき乗を計算する
- gmp_powm — べき乗とモジュロを計算する
- gmp_prob_prime — 数が"ほぼ素数"であるかどうかを調べる
- gmp_random — 乱数を生成する
- gmp_scan0 — 0 を探す
- gmp_scan1 — 1 を探す
- gmp_setbit — ビットを設定する
- gmp_sign — 数の符号
- gmp_sqrt — 平方根を計算する
- gmp_sqrtrem — 余りつきの平方根
- gmp_strval — GMP 数を文字列に変換する
- gmp_sub — 数値の減算
- gmp_testbit — ビットが設定されているかどうかを調べる
- gmp_xor — 排他的論理和を計算する

---

gnupg 関数

導入

このモジュールにより、» gnupg を扱うことが可能となります。

要件

gnupg 拡張モジュールは PHP 4.3 以降を必要とします。 この拡張モジュールをオブジェクト指向形式で使用するには、PHP 5 が必要です。

この拡張モジュールは » gpgme ライブラリを必要とします。

インストール手順

gnupg 拡張モジュールは PHP にバンドルされていません。これは » PECL 拡張モジュールであり、 » http://pecl.php.net/package/gnupg にあります。

定義済み定数

GNUPG_SIG_MODE_NORMAL (integer)
GNUPG_SIG_MODE_DETACH (integer)
GNUPG_SIG_MODE_CLEAR (integer)
GNUPG_VALIDITY_UNKNOWN (integer)
GNUPG_VALIDITY_UNDEFINED (integer)
GNUPG_VALIDITY_NEVER (integer)
GNUPG_VALIDITY_MARGINAL (integer)
GNUPG_VALIDITY_FULL (integer)
GNUPG_VALIDITY_ULTIMATE (integer)
GNUPG_PROTOCOL_OpenPGP (integer)
GNUPG_PROTOCOL_CMS (integer)
GNUPG_SIGSUM_VALID (integer)
GNUPG_SIGSUM_GREEN (integer)
GNUPG_SIGSUM_RED (integer)
GNUPG_SIGSUM_KEY_REVOKED (integer)
GNUPG_SIGSUM_KEY_EXPIRED (integer)
GNUPG_SIGSUM_KEY_MISSING (integer)
GNUPG_SIGSUM_SIG_EXPIRED (integer)
GNUPG_SIGSUM_CRL_MISSING (integer)
GNUPG_SIGSUM_CRL_TOO_OLD (integer)
GNUPG_SIGSUM_BAD_POLICY (integer)
GNUPG_SIGSUM_SYS_ERROR (integer)
GNUPG_ERROR_WARNING (integer)
GNUPG_ERROR_EXCEPTION (integer)
GNUPG_ERROR_SILENT (integer)

注意

この拡張モジュールは、現在のユーザのキーリングを使用します。キーリングは、 通常は ~./.gnupg/ にあります。別の場所を指定するには、環境変数 GNUPGHOME にキーリングへのパスを格納します。その詳細な方法については putenv を参照ください。

キーを特定する設定項目を必要とする関数も存在します。この設定項目は、何らかの ユニークなキー (ユーザ ID、キー ID、フィンガープリント、…) を参照します。 このドキュメントでは、すべての例でフィンガープリントを使用しています。

keylistiterator

この拡張モジュールでは、キーリングの内容を繰り返し処理することも可能です。

```
<?php
// 'example' にマッチするキーを抜き出して繰り返し処理をするイテレータを生成します
$iterator = new gnupg_keylistiterator("example");
foreach($iterator as $fingerprint => $userid){
    echo $fingerprint." -> ".$userid."¥n";
}
?>
```

例

この例は、指定したテキストに署名をします。

Example#1 gnupg での署名の例 (手続き型)

```
<?php
// gnupg を初期化します
$res = gnupg_init();
// 必須ではありません。Clearsign はデフォルト値です
gnupg_setsignmode($res,GNUPG_SIG_MODE_CLEAR);
// 署名のためのキーを、パスフレーズ 'test' で追加します
gnupg_addsignkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
// 署名します
$signed = gnupg_sign($res,"just a test");
echo $signed;
?>
```

Example#2 gnupg での署名の例 (オブジェクト指向)

```
<?php
// 新しいクラス
$gnupg = new gnupg();
// 必須ではありません。Clearsign はデフォルト値です
$gnupg->setsignmode(gnupg::SIG_MODE_CLEAR);
// 署名のためのキーを、パスフレーズ 'test' で追加します
$gnupg->addsignkey("8660281B6051D071D94B5B230549F9DC851566DC","test");
// 署名します
$signed = $gnupg->sign("just a test");
echo $signed;
?>
```

---

gnupg_adddecryptkey

(PECL gnupg:0.5-1.3.1)

gnupg_adddecryptkey — 復号のためのキーを追加する

説明

bool **gnupg_adddecryptkey** (resource `$identifier` , string `$fingerprint` , string `$passphrase`)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_adddecryptkey() の例

```
<?php
$res = gnupg_init();
gnupg_adddecryptkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
?>
```

Example#2 オブジェクト指向の gnupg_adddecryptkey() の例

```
<?php
$gpg = new gnupg();
$gpg -> adddecryptkey("8660281B6051D071D94B5B230549F9DC851566DC","test");
?>
```

---

gnupg_addencryptkey

(PECL gnupg:0.5-1.3.1)

gnupg_addencryptkey — 暗号化のためのキーを追加する

説明

bool **gnupg_addencryptkey** (resource `$identifier` , string `$fingerprint`)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_addencryptkey() の例

```
<?php
$res = gnupg_init();
gnupg_addencryptkey($res,"8660281B6051D071D94B5B230549F9DC851566DC");
?>
```

Example#2 オブジェクト指向の gnupg_addencryptkey() の例

```
<?php
$gpg = new gnupg();
$gpg -> addencryptkey("8660281B6051D071D94B5B230549F9DC851566DC");
?>
```

---

gnupg_addsignkey

(PECL gnupg:0.5-1.3.1)

gnupg_addsignkey — 署名のためのキーを追加する

説明

bool **gnupg_addsignkey** (resource `$identifier` , string `$fingerprint` [, string `$passphrase`])

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_addsignkey() の例

```
<?php
$res = gnupg_init();
gnupg_addsignkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
?>
```

Example#2 オブジェクト指向の gnupg_addsignkey() の例

```
<?php
$gpg = new gnupg();
$gpg -> addsignkey("8660281B6051D071D94B5B230549F9DC851566DC","test");
?>
```

---

gnupg_cleardecryptkeys

(No version information available, might be only in CVS)

gnupg_cleardecryptkeys — 事前に復号のために設定されたすべてのキーを削除する

説明

bool **gnupg_cleardecryptkeys** (resource `$identifier`)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_cleardecryptkeys() の例

```
<?php
$res = gnupg_init();
gnupg_cleardecryptkeys($res);
?>
```

Example#2 オブジェクト指向の gnupg_cleardecryptkeys() の例

```
<?php
$gpg = new gnupg();
$gpg -> cleardecryptkeys();
?>
```

---

gnupg_clearencryptkeys

(PECL gnupg:0.5-1.3.1)

gnupg_clearencryptkeys — 事前に暗号化のために設定されたすべてのキーを削除する

説明

bool **gnupg_clearencryptkeys** (resource `$identifier`)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_clearencryptkeys() の例

```
<?php
$res = gnupg_init();
gnupg_clearencryptkeys($res);
?>
```

Example#2 オブジェクト指向の gnupg_clearencryptkeys() の例

```
<?php
$gpg = new gnupg();
$gpg -> clearencryptkeys();
?>
```

---

gnupg_clearsignkeys

(No version information available, might be only in CVS)

gnupg_clearsignkeys — 事前に署名のために設定されたすべてのキーを削除する

説明

bool **gnupg_clearsignkeys** (resource `$identifier`)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_clearsignkeys() の例

```
<?php
$res = gnupg_init();
gnupg_clearsignkeys($res);
?>
```

Example#2 オブジェクト指向の gnupg_clearsignkeys() の例

```
<?php
```

```
$gpg = new gnupg();
$gpg -> clearsignkeys();
?>
```

---

gnupg_decrypt

(PECL gnupg:0.1-1.3.1)

gnupg_decrypt — 指定されたテキストを復号する

説明

string **gnupg_decrypt** (resource `$identifier` , string `$text`)

事前に gnupg_adddecryptkey で設定されたキーを使用し、指定されたテキストを復号します。

返り値

成功した場合、この関数は復号されたテキストを返します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 手続き型の gnupg_decrypt() の例

```
<?php
$res = gnupg_init();
gnupg_adddecryptkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
$plain = gnupg_decrypt($res,$encrypted_text);
echo $plain;
?>
```

Example#2 オブジェクト指向の gnupg_encrypt() の例

```
<?php
$gpg = new gnupg();
$gpg -> adddecryptkey("8660281B6051D071D94B5B230549F9DC851566DC","test");
$plain = $gpg -> decrypt($encrypted_text);
echo $plain;
?>
```

---

gnupg_decryptverify

(PECL gnupg:0.2-1.3.1)

gnupg_decryptverify — 指定されたテキストを復号し、検証する

説明

array **gnupg_decryptverify** (resource `$identifier` , string `$text` , string `&$plaintext`)

指定したテキストを復号・検証し、署名の情報を返します。 復号されたテキストは *plaintext* に保存されます。

返り値

成功した場合、この関数は署名の情報を返し、復号されたテキストを *plaintext* に保存します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 手続き型の gnupg_decryptverify() の例

```
<?php
$plaintext = "";
$res = gnupg_init();
gnupg_adddecryptkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
$info = gnupg_decryptverify($res,$text,$plaintext);
print_r($info);
?>
```

Example#2 オブジェクト指向の gnupg_decryptverify() の例

```
<?php
$plaintext = "";
$gpg = new gnupg();
$gpg -> adddecryptkey("8660281B6051D071D94B5B230549F9DC851566DC","test");
$info = $gpg -> decryptverify($text,$plaintext);
print_r($info);
```

```
?>
```

---

gnupg_encrypt

(PECL gnupg:0.1-1.3.1)

gnupg_encrypt — 指定したテキストを暗号化する

説明

string **gnupg_encrypt** (resource `$identifier` , string `$plaintext`)

事前に gnupg_addencryptkey で設定したキーを使用し、指定されたテキストを暗号化します。暗号化された テキストを返します。

返り値

成功した場合、この関数は暗号化されたテキストを返します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 手続き型の gnupg_encrypt() の例

```
<?php
$res = gnupg_init();
gnupg_addencryptkey($res,"8660281B6051D071D94B5B230549F9DC851566DC");
$enc = gnupg_encrypt($res, "just a test");
echo $enc;
?>
```

Example#2 オブジェクト指向の gnupg_encrypt() の例

```
<?php
$gpg = new gnupg();
$gpg -> addencryptkey("8660281B6051D071D94B5B230549F9DC851566DC");
$enc = $gpg -> encrypt("just a test");
echo $enc;
?>
```

---

gnupg_encryptsign

(No version information available, might be only in CVS)

gnupg_encryptsign — 指定したテキストを暗号化し、署名する

説明

string **gnupg_encryptsign** (resource `$identifier` , string `$plaintext`)

事前に gnupg_addsignkey および gnupg_addencryptkey で設定したキーを使用し、指定した *plaintext* を暗号化・署名します。暗号化・署名済みのテキストを返します。

返り値

成功した場合、この関数は暗号化されて署名されたテキストを返します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 手続き型の gnupg_encryptsign() の例

```
<?php
$res = gnupg_init();
gnupg_addencryptkey($res,"8660281B6051D071D94B5B230549F9DC851566DC");
gnupg_addsignkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
$enc = gnupg_encryptsign($res, "just a test");
echo $enc;
?>
```

Example#2 オブジェクト指向の gnupg_encryptsign() の例

```
<?php
$gpg = new gnupg();
$gpg -> addencryptkey("8660281B6051D071D94B5B230549F9DC851566DC");
$gpg -> addsignkey("8660281B6051D071D94B5B230549F9DC851566DC","test");
$enc = $gpg -> encryptsign("just a test");
echo $enc;
?>
```

gnupg_export

(PECL gnupg:0.1-1.3.1)

gnupg_export — キーをエクスポートする

説明

string **gnupg_export** (resource `$identifier` , string `$fingerprint`)

キーの *fingerprint* をエクスポートします。

返り値

成功した場合、この関数はキーのデータを返します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 Procedural gnupg_export() example

```
<?php
$res = gnupg_init();
$export = gnupg_export($res,"8660281B6051D071D94B5B230549F9DC851566DC");
echo $export;
?>
```

Example#2 OO gnupg_export() example

```
<?php
$gpg = new gnupg();
$export = $gpg -> export("8660281B6051D071D94B5B230549F9DC851566DC");
?>
```

gnupg_geterror

(PECL gnupg:0.1-1.3.1)

gnupg_geterror — 関数が失敗した場合にエラー文字列を返す

説明

string **gnupg_geterror** (resource `$identifier`)

返り値

エラーが発生した場合にエラー文字列、それ以外の場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_geterror() の例

```
<?php
$res = gnupg_init();
echo gnupg_geterror($res);
?>
```

Example#2 オブジェクト指向の gnupg_geterror() の例

```
<?php
$gpg = new gnupg();
echo $gpg -> geterror();
?>
```

gnupg_getprotocol

(PECL gnupg:0.1-1.3.1)

gnupg_getprotocol — すべての操作で現在アクティブなプロトコルを返す

説明

int **gnupg_getprotocol** (resource $identifier)

返り値

現在アクティブなプロトコルを返します。結果は **GNUPG_PROTOCOL_OpenPGP** あるいは **GNUPG_PROTOCOL_CMS** のいずれかとなります。

例

Example#1 手続き型の gnupg_getprotocol() の例

```
<?php
$res = gnupg_init();
echo gnupg_getprotocol($res);
?>
```

Example#2 オブジェクト指向の gnupg_getprotocol() の例

```
<?php
$gpg = new gnupg();
echo $gpg -> getprotocol();
?>
```

---

gnupg_import

(PECL gnupg:0.3-1.3.1)

gnupg_import — キーをインポートする

説明

array **gnupg_import** (resource $identifier , string $keydata)

キー *keydata* をインポートし、インポート処理に ついての情報を配列で返します。

返り値

成功した場合、この関数はインポート処理の情報を配列で返します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 手続き型の gnupg_import() の例

```
<?php
$res = gnupg_init();
$info = gnupg_import($res,$keydata);
print_r($info);
?>
```

Example#2 オブジェクト指向の gnupg_import() の例

```
<?php
$gpg = new gnupg();
$info = $gpg -> import($keydata);
print_r($info);
?>
```

---

gnupg_keyinfo

(PECL gnupg:0.1-1.3.1)

gnupg_keyinfo — 指定したパターンに一致するすべてのキーについての情報を配列で返す

説明

array **gnupg_keyinfo** (resource $identifier , string $pattern)

返り値

指定したパターンに一致するすべてのキーについての情報を配列で返します。 エラーが発生した場合は **FALSE** を返します。

例

Example#1 手続き型の gnupg_keyinfo() の例

```
<?php
$res = gnupg_init();
$info = gnupg_keyinfo($res, 'test');
print_r($info);
?>
```

Example#2 オブジェクト指向の gnupg_keyinfo() の例

```
<?php
$gpg = new gnupg();
$info = $gpg -> keyinfo("test");
print_r($info);
?>
```

---

gnupg_setarmor

(PECL gnupg:0.1-1.3.1)

gnupg_setarmor — armor 形式を切り替える

説明

bool **gnupg_setarmor** (resource $identifier , int $armor)

ゼロ以外の整数値をこの関数に渡すと、armor 形式の出力を有効にします (デフォルト)。 0 を渡すと、armor 形式の出力を無効にします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_setarmor() の例

```
<?php
$res = gnupg_init();
gnupg_setarmor($res,1); // enable armored output;
gnupg_setarmor($res,0); // disable armored output;
?>
```

Example#2 オブジェクト指向の gnupg_setarmor() の例

```
<?php
$gpg = new gnupg();
$gpg -> setarmor(1); // enable armored output;
$gpg -> setarmor(0); // disable armored output;
?>
```

---

gnupg_seterrormode

(PECL gnupg:0.6-1.3.1)

gnupg_seterrormode — エラー報告モードを設定する

説明

void **gnupg_seterrormode** (resource $identifier , int $errormode)

errormode には、使用するエラー報告形式を表す 定数を指定します。 **GNUPG_ERROR_WARNING**、 **GNUPG_ERROR_EXCEPTION** および **GNUPG_ERROR_SILENT** が使用可能です。 デフォルトでは **GNUPG_ERROR_SILENT** が使用されます。

返り値

値を返しません。

例

Example#1 手続き型の gnupg_seterrormode() の例

```
<?php
$res = gnupg_init();
gnupg_seterrormode($res,GNUPG_ERROR_WARNING); // エラー時には PHP の警告を発生させます
?>
```

Example#2 オブジェクト指向の gnupg_seterrormode() の例

```
<?php
$gpg = new gnupg();
$gpg -> seterrormode(gnupg::ERROR_EXCEPTION); //  エラー時には例外をスローします
?>
```

---

gnupg_setsignmode

(PECL gnupg:0.1-1.3.1)

gnupg_setsignmode — 署名方式を設定する

説明

bool **gnupg_setsignmode** (resource $identifier , int $signmode)

signmode には、作成する署名の形式を表す 定数をしていします。 **GNUPG_SIG_MODE_NORMAL**、 **GNUPG_SIG_MODE_DETACH** および **GNUPG_SIG_MODE_CLEAR** が使用可能です。 デフォルトでは **GNUPG_SIG_MODE_CLEAR** が使用されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 手続き型の gnupg_setsignmode() の例

```
<?php
$res = gnupg_init();
gnupg_setsignmode($res,GNUPG_SIG_MODE_DETACH); //  独立した署名を作成します
?>
```

Example#2 オブジェクト指向の gnupg_setsignmode() の例

```
<?php
$gpg = new gnupg();
$gpg -> setsignmode(gnupg::SIG_MODE_DETACH); //  独立した署名を作成します
?>
```

---

gnupg_sign

(PECL gnupg:0.1-1.3.1)

gnupg_sign — 指定したテキストに署名する

説明

string **gnupg_sign** (resource $identifier , string $plaintext)

事前に gnupg_addsignkey で設定したキーを使用して *plaintext* に署名し、 gnupg_setsignmode の設定に応じて署名済みテキストあるいは署名を返します。

返り値

成功した場合、この関数は署名済みテキストあるいは署名を返します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 手続き型の gnupg_sign() の例

```
<?php
$res = gnupg_init();
gnupg_addsignkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
$signed = gnupg_sign($res, "just a test");
echo $signed;
?>
```

Example#2 オブジェクト指向の gnupg_sign() の例

```
<?php
$gpg = new gnupg();
$gpg -> setsignkey("8660281B6051D071D94B5B230549F9DC851566DC","test");
$signed = $gpg -> sign("just a test");
```

```
echo $signed;
?>
```

---

gnupg_verify

(PECL gnupg:0.1-1.3.1)

gnupg_verify — 署名済みテキストを検証する

説明

array **gnupg_verify** (resource `$identifier` , string `$signed_text` , string `$signature` [, string `&$plaintext`])

指定した *signed_text* を検証し、署名についての 情報を返します。clearsign で署名されたテキストを検証するには、 signature に **FALSE** を設定します。 オプションのパラメータ *plaintext* を渡すと、 復号されたテキストがそこに保存されます。

返り値

成功した場合、この関数は署名についての情報を返します。 失敗した場合、この関数は **FALSE** を返します。

例

Example#1 手続き型の gnupg_verify() の例

```
<?php
$plaintext = "";
$res = gnupg_init();
// clearsigned
$info = gnupg_verify($res,$signed_text,false,$plaintext);
print_r($info);
// detached signature
$info = gnupg_verify($res,$signed_text,$signature);
print_r($info);
?>
```

Example#2 オブジェクト指向の gnupg_verify() の例

```
<?php
$plaintext = "";
$gpg = new gnupg();
// clearsigned
$info = $gpg -> verify($signed_text,false,$plaintext);
print_r($info);
// detached signature
$info = $gpg -> verify($signed_text,$signature);
print_r($info);
?>
```

目次

Net_Gopher

導入

gopher プロトコルは » RFC 1436 で定義されており、現在の HTTP プロトコルの先祖にあたるものと みなされています。しかし gopher は、非 gopher リソースへの 参照機能も提供しており、telnet・wais・nntp そしてもちろん http についても参照可能です。この拡張モジュールは PHP の URL ラッパに gopher の サポートを追加し、gopher フォーマットのディレクトリー覧を表示する ヘルパ関数 gopher_parsedir() を提供します。

要件

インストール手順

前提条件: PHP 4.3.0 以降が必要です。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 »
http://pecl.php.net/package/net_gopher

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

Net_Gopher 定数

定数	値	説明
GOPHER_DOCUMENT	0	標準的な *text/plain* ドキュメント。
GOPHER_DIRECTORY	1	gopher フォーマットのディレクトリー覧を含むリソース。
GOPHER_BINHEX	4	BinHex エンコードされたバイナリファイル。
GOPHER_DOSBINARY	5	DOS フォーマットのバイナリアーカイブ。
GOPHER_UUENCODED	6	UUEncode されたファイル。
GOPHER_BINARY	9	一般的なバイナリファイル。
GOPHER_INFO	255	情報エントリ。
GOPHER_HTTP	254	HTTP リソースへの参照。
GOPHER_UNKNOWN	-1	特定のカテゴリにあてはまらないエントリ。

例

```
<?php
readfile("gopher://gopher.example.com/somedocument");
?>
```

---

gopher_parsedir

(PECL net_gopher:0.1-1.0.0)

gopher_parsedir — gopher フォーマットのディレクトリエントリを連想配列に変換する

説明

array **gopher_parsedir** (string `$dirent`)

gopher_parsedir() は、gopher フォーマットされたディレクトリエントリを連想配列に変換します。

gopher は、リクエストに対して *text/plain* 形式の ドキュメントを返します。ディレクトリ（例えば / など）へのリクエストには 特別にエンコードされた一連の行を返し、その個々の行が 1 つのディレクトリ エントリあるいは情報行を表します。

パラメータ

dirent

ディレクトリエントリ。

返り値

以下の要素からなる連想配列を返します。

- *type* - *GOPHER_XXX* 定数のいずれか。
- *title* - リソースの名前。
- *path* - リソースのパス。
- *host* - このドキュメント (あるいはディレクトリ) を保持するホストのドメイン名。
- *port* - *host* に接続するポート。

失敗した場合には、さらに *data* というエントリがこの配列に追加され、そこにパースした行が含まれます。

例

Example#1 *gopher://gopher.example.com/* からの出力を以下のように仮定する

```
0All about my gopher site.                  /allabout.txt                   gopher.example.com     70
9A picture of my cat.                        /pics/cat.png                  gopher.example.com     70
1A collection of my writings.               /stories                        gopher.example.com     70
hThe HTTP version of this site.               URL:http://www.example.com    gopher.example.com     70
1Mirror of this site in Spain.              /                                gopher.ejemplo.co.es   70
iWelcome to my gopher site.                                                 error.host             1
iPlease select one of the options above                                     error.host             1
iSend complaints to /dev/null                                               error.host             1
iLong live gopher!                                                          error.host             1
```

上の例で、gopher.example.com のルートディレクトリには ID *0* の ドキュメント が *gopher://gopher.example.com:70/allabout.txt* にあります。また、2 つのディレクトリ（それぞれ独自のファイル一覧を保持します）が *gopher://gopher.exmaple.com:70/stories* および *gopher://gopher.ejemplo.co.es:70/* にあります。 その他にバイナリファイル・HTTP URL へのリンク・そして情報の行があります。

ディレクトリ一覧の各行を **gopher_parsedir()** に渡すと、 そのデータを連想配列形式にフォーマットします。

Example#2 gopher_parsedir() の使用例

```php
<?php
$directory  = file("gopher://gopher.example.com");

foreach($directory  as $dirent) {
    print_r(gopher_parsedir($dirent));
}
?>
```

上の例の出力は以下となります。

```
Array  (
   [type] => 0
   [title] => All about my gopher site.
   [path] => /allabout.txt
   [host] => gopher.example.com
   [port] => 70
)
Array  (
   [type] => 9
   [title] => A picture of my cat.
   [path] => /pics/cat.png
   [host] => gopher.example.com
   [port] => 70
)
Array  (
   [type] => 1
   [title] => A collection of my writings.
   [path] => /stories
   [host] => gopher.example.com
   [port] => 70
)
Array  (
   [type] => 254
   [title] => The HTTP version of this site.
   [path] => URL:http://www.example.com
   [host] => gopher.example.com
   [port] => 70
)
Array  (
   [type] => 1
   [title] => Mirror of this site in Spain.
```

```
   [path] => /
    [host] => gopher.ejemplo.co.es
   [port] => 70
)
Array (
   [type] => 255
   [title] => Welcome to my gopher site.
   [path] =>
   [host] => error.host
   [port] => 1
)
Array (
   [type] => 255
   [title] => Please select one of the options above.
   [path] =>
   [host] => error.host
   [port] => 1
)
Array (
   [type] => 255
   [title] => Send complaints to /dev/null
   [path] =>
   [host] => error.host
   [port] => 1
)
Array (
   [type] => 255
   [title] => Long live gopher!
   [path] =>
   [host] => error.host
   [port] => 1
)
```

Haru PDF 関数

導入

PECL/haru 拡張モジュールは、Haru Free PDF Library を使用するためのものです。 これは、フリーでクロスプラットフォームであるオープンソースライブラリで、 PDF ファイルを作成するためのものです。

このライブラリは » http://libharu.sourceforge.net にあります。

警告

この拡張モジュールは、 *実験的* なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

要件

PECL/haru を使用するには、まず libharu をインストールする必要があります。 PECL/haru の動作検証は libharu 2.0.8 で行っています。 これより古いバージョンでは、動作するかもしれませんし、しないかもしれません。 PECL/haru は、PHP 5.1.3 以降でないと動作しません。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/haru.

最新の PECL/haru Win32 DLL はこちらからダウンロードできます。 » php_haru.dll.

実行時設定

設定ディレクティブは定義されていません。

例

Example#1 基本的な PECL/haru の例

```php
<?php

$doc = new HaruDoc;

$doc->setPageMode(HaruDoc::PAGE_MODE_USE_THUMBS); /* サムネイルを表示します */
```

```
$page = $doc->addPage(); /* ドキュメントにページを追加します */
$page->setSize(HaruPage::SIZE_A4, HaruPage::LANDSCAPE); /* ページを A4 横に設定します */

$courier = $doc->getFont("Courier-Bold"); /* 数行先で、組み込みのフォントを使用します */

$page->setRGBStroke(0, 0, 0); /* 色を設定します */
$page->setRGBFill(0.7, 0.8, 0.9);
$page->rectangle(150, 150, 550, 250); /* 矩形を描画します */

$page->fillStroke(); /* 塗りつぶします */

$page->setDash(array(3, 3), 0); /* このページの破線の形式を設定します */
$page->setFontAndSize($courier, 60); /* フォントとサイズを設定します */

$page->setRGBStroke(0.5, 0.5, 0.1); /* 線の色を設定します */
$page->setRGBFill(1, 1, 1); /* 塗りつぶし色を設定します */

$page->setTextRenderingMode(HaruPage::FILL_THEN_STROKE); /* テキストを塗りつぶし、描画します */

/* テキストを表示します */
$page->beginText();
$page->textOut(210, 270, "Hello World!");
$page->endText();

$doc->save("/tmp/test.pdf"); /* ドキュメントをファイルに保存します */

?>
```

できあがったドキュメントをお好みの PDF ビューアで開くと、 ライトブルーの矩形と、その中にある白字の "Hello World!" が見られることでしょう。

定義済みクラス

- HaruException
- HaruDoc
- HaruPage
- HaruFont
- HaruImage
- HaruEncoder
- HaruOutline
- HaruAnnotation
- HaruDestination

組み込みのフォントおよびエンコーディング

組み込みフォント

これら 14 種類の基本フォントは PDF に組み込まれており、 すべてのビューアで表示することができます。 これらのフォントを使用すると、出来上がるファイルの大きさを抑えることができ、 高速に処理できます。また外部フォントの読み込みを避けることができます。 しかし、これらのフォントは latin1 文字セットしかサポートしていません。 その他の文字セットを使用する必要がある場合は、外部のフォントを読み込まなければなりません。

基本の 14 フォントは次のとおりです。

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol

- ZapfDingbats

組み込みエンコーディング

シングルバイト文字セットエンコーディング

名前	説明
StandardEncoding	PDF のデフォルトエンコーディング。
MacRomanEncoding	Mac OS の標準エンコーディング。
WinAnsiEncoding	Windows の標準エンコーディング。
FontSpecific	フォントの組み込みエンコーディング。
ISO8859-2	Latin2 (東欧)
ISO8859-3	Latin3 (南欧)
ISO8859-4	Latin4 (北欧)
ISO8859-5	キリル文字
ISO8859-6	アラビア語
ISO8859-7	ギリシャ語
ISO8859-8	ヘブライ語
ISO8859-9	Latin5 (トルコ語)
ISO8859-10	Latin6 (北欧)
ISO8859-11	タイ語
ISO8859-13	Latin7 (バルト語)
ISO8859-14	Latin8 (ケルト語)
ISO8859-15	Latin9
ISO8859-16	Latin10
CP1250	MS Windows コードページ 1250。
CP1251	MS Windows コードページ 1251。
CP1252	MS Windows コードページ 1252。
CP1253	MS Windows コードページ 1253。
CP1254	MS Windows コードページ 1254。
CP1255	MS Windows コードページ 1255。
CP1256	MS Windows コードページ 1256。
CP1257	MS Windows コードページ 1257。
CP1258	MS Windows コードページ 1258。
KOI8-R	キリル文字セット。

マルチバイト文字セットエンコーディング

名前	説明
GB-EUC-H	EUC-CN エンコーディング。
GB-EUC-V	GB-EUC-H の縦書き版。
GBK-EUC-H	Microsoft コードページ 936 (lfCharSet 0x86) GBK エンコーディング。
GBK-EUC-V	GBK-EUC-H の縦書き版。
ETen-B5-H	Microsoft コードページ 950 (lfCharSet 0x88) Big Five 文字セット (倚天拡張)。
ETen-B5-V	ETen-B5-H の縦書き版。
90ms-RKSJ-H	Microsoft コードページ 932, JIS X 0208 文字。
90ms-RKSJ-V	90ms-RKSJ-V の縦書き版。
90msp-RKSJ-H	Microsoft コードページ 932, JIS X 0208 文字 (プロポーショナル)。
EUC-H	JIS X 0208 文字セット, EUC-JP エンコーディング。
EUC-V	EUC-H の縦書き版。
KSC-EUC-H	KS X 1001:1992 文字セット, EUC-KR エンコーディング。
KSC-EUC-V	KSC-EUC-V の縦書き版。
KSCms-UHC-H	Microsoft コードページ (lfCharSet 0x81), KS X 1001:1992 文字セットに 8822 のハングルを追加したもの, Unified Hangul Code (UHC) エンコーディング (プロポーショナル)。

名前	説明
KSCms-UHC-HW-H	Microsoft コードページ 949 (lfCharSet 0x81), KS X 1001:1992 文字セットに 8822 のハングルを追加したもの, Unified Hangul Code (UHC) エンコーディング (固定幅)。
KSCms-UHC-HW-V	KSCms-UHC-HW-H の縦書き版。

HaruException

(No version information available, might be only in CVS)

HaruException — Haru PDF 例外クラス

```
class HaruException
```

クラスのメンバ

メソッド

HaruDoc

(No version information available, might be only in CVS)

HaruDoc — Haru PDF ドキュメントクラス

```
class HaruDoc
```

クラスのメンバ

定義済み定数

型	名前	説明
int	HaruDoc::CS_DEVICE_GRAY	
int	HaruDoc::CS_DEVICE_RGB	
int	HaruDoc::CS_DEVICE_CMYK	
int	HaruDoc::CS_CAL_GRAY	
int	HaruDoc::CS_CAL_RGB	
int	HaruDoc::CS_LAB	
int	HaruDoc::CS_ICC_BASED	
int	HaruDoc::CS_SEPARATION	
int	HaruDoc::CS_DEVICE_N	
int	HaruDoc::CS_INDEXED	
int	HaruDoc::CS_PATTERN	
int	HaruDoc::ENABLE_READ	
int	HaruDoc::ENABLE_PRINT	
int	HaruDoc::ENABLE_EDIT_ALL	
int	HaruDoc::ENABLE_COPY	
int	HaruDoc::ENABLE_EDIT	
int	HaruDoc::ENCRYPT_R2	
int	HaruDoc::ENCRYPT_R3	
int	HaruDoc::INFO_AUTHOR	
int	HaruDoc::INFO_CREATOR	
int	HaruDoc::INFO_TITLE	
int	HaruDoc::INFO_SUBJECT	
int	HaruDoc::INFO_KEYWORDS	

型	名前	説明
int	HaruDoc::INFO_CREATION_DATE	
int	HaruDoc::INFO_MOD_DATE	
int	HaruDoc::COMP_NONE	
int	HaruDoc::COMP_TEXT	
int	HaruDoc::COMP_IMAGE	
int	HaruDoc::COMP_METADATA	
int	HaruDoc::COMP_ALL	
int	HaruDoc::PAGE_LAYOUT_SINGLE	
int	HaruDoc::PAGE_LAYOUT_ONE_COLUMN	
int	HaruDoc::PAGE_LAYOUT_TWO_COLUMN_LEFT	
int	HaruDoc::PAGE_LAYOUT_TWO_COLUMN_RIGHT	
int	HaruDoc::PAGE_MODE_USE_NONE	
int	HaruDoc::PAGE_MODE_USE_OUTLINE	
int	HaruDoc::PAGE_MODE_USE_THUMBS	
int	HaruDoc::PAGE_MODE_FULL_SCREEN	

メソッド

- HaruDoc::__construct()
- HaruDoc::addPageLabel()
- HaruDoc::addPage()
- HaruDoc::createOutline()
- HaruDoc::getCurrentEncoder()
- HaruDoc::getCurrentPage()
- HaruDoc::getEncoder()
- HaruDoc::getFont()
- HaruDoc::getInfoAttr()
- HaruDoc::getPageLayout()
- HaruDoc::getPageMode()
- HaruDoc::getStreamSize()
- HaruDoc::insertPage()
- HaruDoc::loadJPEG()
- HaruDoc::loadPNG()
- HaruDoc::loadRAW()
- HaruDoc::loadTTC()
- HaruDoc::loadTTC()
- HaruDoc::loadType1()
- HaruDoc::output()
- HaruDoc::readFromStream()
- HaruDoc::resetError()
- HaruDoc::resetStream()
- HaruDoc::save()
- HaruDoc::setCompressionMode()
- HaruDoc::setCurrentEncoder()
- HaruDoc::setEncryptionMode()
- HaruDoc::setInfoAttr()
- HaruDoc::setInfoDateAttr()
- HaruDoc::setOpenAction()
- HaruDoc::setPageLayout()
- HaruDoc::setPageMode()
- HaruDoc::setPagesConfiguration()
- HaruDoc::setPassword()
- HaruDoc::setPermission()
- HaruDoc::useCNSEncodings()
- HaruDoc::useCNSFonts()
- HaruDoc::useCNTEncodings()
- HaruDoc::useCNTFonts()
- HaruDoc::useJPEncodings()

- HaruDoc::useJPFonts()
- HaruDoc::useKREncodings()
- HaruDoc::useKRFonts()

HaruPage

(No version information available, might be only in CVS)

HaruPage — Haru PDF ページクラス

```
class HaruPage
```

クラスのメンバ

定義済み定数

型	名前	説明
int	HaruPage::GMODE_PAGE_DESCRIPTION	
int	HaruPage::GMODE_TEXT_OBJECT	
int	HaruPage::GMODE_PATH_OBJECT	
int	HaruPage::GMODE_CLIPPING_PATH	
int	HaruPage::GMODE_SHADING	
int	HaruPage::GMODE_INLINE_IMAGE	
int	HaruPage::GMODE_EXTERNAL_OBJECT	
int	HaruPage::BUTT_END	
int	HaruPage::ROUND_END	
int	HaruPage::PROJECTING_SCUARE_END	
int	HaruPage::MITER_JOIN	
int	HaruPage::ROUND_JOIN	
int	HaruPage::BEVEL_JOIN	
int	HaruPage::FILL	
int	HaruPage::STROKE	
int	HaruPage::FILL_THEN_STROKE	
int	HaruPage::INVISIBLE	
int	HaruPage::FILL_CLIPPING	
int	HaruPage::STROKE_CLIPPING	
int	HaruPage::FILL_STROKE_CLIPPING	
int	HaruPage::CLIPPING	
int	HaruPage::TALIGN_LEFT	
int	HaruPage::TALIGN_RIGHT	
int	HaruPage::TALIGN_CENTER	
int	HaruPage::TALIGN_JUSTIFY	
int	HaruPage::SIZE_LETTER	
int	HaruPage::SIZE_LEGAL	
int	HaruPage::SIZE_A3	
int	HaruPage::SIZE_A4	
int	HaruPage::SIZE_A5	
int	HaruPage::SIZE_B4	
int	HaruPage::SIZE_B5	
int	HaruPage::SIZE_EXECUTIVE	
int	HaruPage::SIZE_US4x6	
int	HaruPage::SIZE_US4x8	

型	名前	説明
int	HaruPage::SIZE_US5x7	
int	HaruPage::SIZE_COMM10	
int	HaruPage::PORTRAIT	
int	HaruPage::LANDSCAPE	
int	HaruPage::TS_WIPE_LIGHT	
int	HaruPage::TS_WIPE_UP	
int	HaruPage::TS_WIPE_LEFT	
int	HaruPage::TS_WIPE_DOWN	
int	HaruPage::TS_BARN_DOORS_HORIZONTAL_OUT	
int	HaruPage::TS_BARN_DOORS_HORIZONTAL_IN	
int	HaruPage::TS_BARN_DOORS_VERTICAL_OUT	
int	HaruPage::TS_BARN_DOORS_VERTICAL_IN	
int	HaruPage::TS_BOX_OUT	
int	HaruPage::TS_BOX_IN	
int	HaruPage::TS_BLINDS_HORIZONTAL	
int	HaruPage::TS_BLINDS_VERTICAL	
int	HaruPage::TS_DISSOLVE	
int	HaruPage::TS_GLITTER_RIGHT	
int	HaruPage::TS_GLITTER_DOWN	
int	HaruPage::TS_GLITTER_TOP_LEFT_TO_BOTTOM_RIGHT	
int	HaruPage::TS_REPLACE	
int	HaruPage::NUM_STYLE_DECIMAL	
int	HaruPage::NUM_STYLE_UPPER_ROMAN	
int	HaruPage::NUM_STYLE_LOWER_ROMAN	
int	HaruPage::NUM_STYLE_UPPER_LETTERS	
int	HaruPage::NUM_STYLE_LOWER_LETTERS	

メソッド

- HaruPage::arc()
- HaruPage::beginText()
- HaruPage::circle()
- HaruPage::closePath()
- HaruPage::concat()
- HaruPage::createDestination()
- HaruPage::createLinkAnnotation()
- HaruPage::createTextAnnotation()
- HaruPage::createUrlAnnotation()
- HaruPage::curveTo()
- HaruPage::curveTo2()
- HaruPage::curveTo3()
- HaruPage::drawImage()
- HaruPage::ellipse()
- HaruPage::endPath()
- HaruPage::endText()
- HaruPage::eofillStroke()
- HaruPage::eofill()
- HaruPage::fillStroke()
- HaruPage::fill()
- HaruPage::getCharSpace()
- HaruPage::getCMYKFill()
- HaruPage::getCMYKStroke()
- HaruPage::getCurrentFontSize()
- HaruPage::getCurrentFont()
- HaruPage::getCurrentPos()

- HaruPage::getCurrentTextPos()
- HaruPage::getDash()
- HaruPage::getFillingColorSpace()
- HaruPage::getFlatness()
- HaruPage::getGMode()
- HaruPage::getGrayFill()
- HaruPage::getGrayStroke()
- HaruPage::getHeight()
- HaruPage::getHorizontalScaling()
- HaruPage::getLineCap()
- HaruPage::getLineJoin()
- HaruPage::getLineWidth()
- HaruPage::getMiterLimit()
- HaruPage::getRGBFill()
- HaruPage::getRGBStroke()
- HaruPage::getStrokingColorSpace()
- HaruPage::getTextLeading()
- HaruPage::getTextMatrix()
- HaruPage::getTextRenderingMode()
- HaruPage::getTextRise()
- HaruPage::getTextWidth()
- HaruPage::getTransMatrix()
- HaruPage::getWidth()
- HaruPage::getWordSpace()
- HaruPage::lineTo()
- HaruPage::measureText()
- HaruPage::moveTextPos()
- HaruPage::moveToNextLine()
- HaruPage::moveTo()
- HaruPage::rectangle()
- HaruPage::setCharSpace()
- HaruPage::setCMYKFill()
- HaruPage::setCMYKStroke()
- HaruPage::setDash()
- HaruPage::setFlatness()
- HaruPage::setFontAndSize()
- HaruPage::setGrayFill()
- HaruPage::setGrayStroke()
- HaruPage::setHeight()
- HaruPage::setHorizontalScaling()
- HaruPage::setLineCap()
- HaruPage::setLineJoin()
- HaruPage::setLineWidth()
- HaruPage::setMiterLimit()
- HaruPage::setRGBFill()
- HaruPage::setRGBStroke()
- HaruPage::setRotate()
- HaruPage::setSize()
- HaruPage::setSlideShow()
- HaruPage::setTextLeading()
- HaruPage::setTextMatrix()
- HaruPage::setTextTenderingMode()
- HaruPage::setWidth()
- HaruPage::setWordSpace()
- HaruPage::showTextNextLine()
- HaruPage::showText()
- HaruPage::stroke()
- HaruPage::textOut()
- HaruPage::textRect()

HaruFont

(No version information available, might be only in CVS)

HaruFont — Haru PDF フォントクラス

```
class HaruFont
```

クラスのメンバ

メソッド

- HaruFont::getAscent()
- HaruFont::getCapHeight()
- HaruFont::getDescent()
- HaruFont::getEncodingName()
- HaruFont::getFontName()
- HaruFont::getTextWidth()
- HaruFont::getUnicodeWidth()
- HaruFont::getXHeight()
- HaruFont::measureText()

HaruImage

(No version information available, might be only in CVS)

HaruImage — Haru PDF 画像クラス

```
class HaruImage
```

クラスのメンバ

メソッド

- HaruImage::getBitsPerComponent()
- HaruImage::getColorSpace()
- HaruImage::getHeight()
- HaruImage::getSize()
- HaruImage::getWidth()
- HaruImage::setColorMask()
- HaruImage::setMaskImage()

HaruEncoder

(No version information available, might be only in CVS)

HaruEncoder — Haru PDF エンコーダクラス

```
class HaruEncoder
```

クラスのメンバ

定義済み定数

型	名前	説明
int	HaruEncoder::TYPE_SINGLE_BYTE	
int	HaruEncoder::TYPE_DOUBLE_BYTE	
int	HaruEncoder::TYPE_UNINITIALIZED	
int	HaruEncoder::UNKNOWN	
int	HaruEncoder::WMODE_HORIZONTAL	
int	HaruEncoder::WMODE_VERTICAL	
int	HaruEncoder::BYTE_TYPE_SINGLE	

型	名前	説明
int	HaruEncoder::BYTE_TYPE_LEAD	
int	HaruEncoder::BYTE_TYPE_TRAIL	
int	HaruEncoder::BYTE_TYPE_UNKNOWN	

メソッド

- HaruEncoder::**getByteType()**
- HaruEncoder::getType()
- HaruEncoder::**getUnicode()**
- HaruEncoder::**getWritingMode()**

HaruOutline

(No version information available, might be only in CVS)

HaruOutline — Haru PDF アウトラインクラス

```
class HaruOutline
```

クラスのメンバ

メソッド

- HaruOutline::**setDestination()**
- HaruOutline::**setOpened()**

HaruAnnotation

(No version information available, might be only in CVS)

HaruAnnotation — Haru PDF アノテーションクラス

```
class HaruAnnotation
```

クラスのメンバ

定義済み定数

型	名前	説明
int	HaruAnnotation::NO_HIGHLIGHT	
int	HaruAnnotation::INVERT_BOX	
int	HaruAnnotation::INVERT_BORDER	
int	HaruAnnotation::DOWN_APPEARANCE	
int	HaruAnnotation::ICON_COMMENT	
int	HaruAnnotation::ICON_KEY	
int	HaruAnnotation::ICON_NOTE	
int	HaruAnnotation::ICON_HELP	
int	HaruAnnotation::ICON_NEW_PARAGRAPH	
int	HaruAnnotation::ICON_PARAGRAPH	
int	HaruAnnotation::ICON_INSERT	

メソッド

- HaruAnnotation::**setBorderStyle()**
- HaruAnnotation::**setHighlightMode()**
- HaruAnnotation::**setIcon()**

- HaruAnnotation::**setOpened()**

HaruDestination

(No version information available, might be only in CVS)

HaruDestination — Haru PDF 対象クラス

```
class HaruDestination
```

クラスのメンバ

メソッド

- HaruDestination::**setFitBH()**
- HaruDestination::**setFitBV()**
- HaruDestination::**setFitB()**
- HaruDestination::**setFitH()**
- HaruDestination::**setFitR()**
- HaruDestination::**setFitV()**
- HaruDestination::**setFit()**
- HaruDestination::**setXYZ()**

HaruAnnotation::setBorderStyle

(PECL haru:0.0.1)

HaruAnnotation::setBorderStyle — アノテーションの枠の形式を設定する

説明

bool **HaruAnnotation::setBorderStyle** (float `$width` , int `$dash_on` , int `$dash_off`)

アノテーションの枠の形式を定義します。 この関数は、リンクアノテーションに対してのみ使用します。

パラメータ

`width`

枠線の幅。

`dash_on`

破線の形式。

`dash_off`

破線の形式。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruAnnotation::setHighlightMode

(PECL haru:0.0.1)

HaruAnnotation::setHighlightMode — アノテーションの強調モードを設定する

説明

bool **HaruAnnotation::setHighlightMode** (int $mode)

アノテーションをクリックした際の表示方法を定義します。 この関数は、リンクアノテーションに対してのみ使用します。

パラメータ

mode

アノテーションの強調モード。以下のいずれかの値となります。

- **HaruAnnotation::NO_HIGHLIGHT** - 強調しない。
- **HaruAnnotation::INVERT_BOX** - アノテーションの内容を反転する。
- **HaruAnnotation::INVERT_BORDER** - アノテーションの枠線を反転する。
- **HaruAnnotation::DOWN_APPEARANCE** - アノテーションを目立たなくする。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruAnnotation::setIcon

(PECL haru:0.0.1)

HaruAnnotation::setIcon — アノテーションのアイコンの形式を設定する

説明

bool **HaruAnnotation::setIcon** (int $icon)

アノテーションアイコンの形式を定義します。 この関数は、テキストアノテーションに対してのみ使用します。

パラメータ

icon

アイコンの形式、以下のいずれかの値となります。

- **HaruAnnotation::ICON_COMMENT**
- **HaruAnnotation::ICON_KEY**
- **HaruAnnotation::ICON_NOTE**
- **HaruAnnotation::ICON_HELP**
- **HaruAnnotation::ICON_NEW_PARAGRAPH**
- **HaruAnnotation::ICON_PARAGRAPH**
- **HaruAnnotation::ICON_INSERT**

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruAnnotation::setOpened

(PECL haru:0.0.1)

HaruAnnotation::setOpened — アノテーションの初期状態を設定する

説明

bool **HaruAnnotation::setOpened** (bool $opened)

アノテーションが初期状態で開いているかどうかを定義します。 この関数は、テキストアノテーションに対してのみ使用します。

パラメータ

opened

TRUE は、アノテーションが最初から開いて表示されていることを意味します。 **FALSE** は、閉じていることを意味します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setFit

(PECL haru:0.0.1)

HaruDestination::setFit — ページの表示を、ウィンドウにあわせるよう設定する

説明

bool **HaruDestination::setFit** (void)

ページの表示を、ウィンドウにあわせるよう設定します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setFitB

(PECL haru:0.0.1)

HaruDestination::setFitB — ページの表示を、ページのバウンディングボックスをウィンドウにあわせるよう設定する

説明

bool **HaruDestination::setFitB** (void)

ページの表示を、ページのバウンディングボックスをウィンドウにあわせるよう設定します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setFitBH

(PECL haru:0.0.1)

HaruDestination::setFitBH — ページの表示を、バウンディングボックスの幅にあわせるよう設定する

説明

bool **HaruDestination::setFitBH** (float $top)

ページの表示を拡大し、バウンディングボックスの幅にあわせるようにします。 ページの上端の位置は *top* に設定します。

パラメータ

top

ページの上端の座標。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setFitBV

(PECL haru:0.0.1)

HaruDestination::setFitBV — ページの表示を、バウンディングボックスの高さにあわせるよう設定する

説明

bool **HaruDestination::setFitBV** (float $left)

ページの表示を拡大し、バウンディングボックスの高さにあわせるようにします。 ページの左端の位置は *left* に設定します。

パラメータ

left

ページの左端の座標。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setFitH

(PECL haru:0.0.1)

HaruDestination::setFitH — ページの表示を、ウィンドウの幅にあわせるよう設定する

説明

bool **HaruDestination::setFitH** (float $top)

ページの表示を、ウィンドウの幅にあわせるようにします。 ページの上端の位置は *top* に設定します。

パラメータ

top

ページの上端の位置。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setFitR

(PECL haru:0.0.1)

HaruDestination::setFitR — ページの表示を、指定した矩形にあわせるよう設定する

説明

bool **HaruDestination::setFitR** (float `$left` , float `$bottom` , float `$right` , float `$top`)

ページの表示を、パラメータで指定した矩形にあわせるよう設定します。

パラメータ

`left`

ページの左端の座標。

`bottom`

ページの下端の座標。

`right`

ページの右端の座標。

`top`

ページの上端の座標。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setFitV

(PECL haru:0.0.1)

HaruDestination::setFitV — ページの表示を、ウィンドウの高さにあわせるよう設定する

説明

bool **HaruDestination::setFitV** (float `$left`)

ページの表示を、ウィンドウの高さにあわせるようにします。

パラメータ

`left`

ページの左端の位置。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDestination::setXYZ

(PECL haru:0.0.1)

HaruDestination::setXYZ — ページの表示を設定する

説明

bool **HaruDestination::setXYZ** (float `$left` , float `$top` , float `$zoom`)

ページの表示方法を、三つのパラメータ *left* 、 *top* および *zoom* で設定します。

パラメータ

left

ページの左端の位置。

top

ページの上端の位置。

zoom

拡大率。0.08 (8%) から 32 (3200%) までの値でなければなりません。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::addPage

(PECL haru:0.0.1)

HaruDoc::addPage — 新しいページをドキュメントに追加する

説明

object **HaruDoc::addPage** (void)

新しいページをドキュメントに追加し、新しい HaruPage インスタンスを返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**insertPage()**

HaruDoc::addPageLabel

(PECL haru:0.0.1)

HaruDoc::addPageLabel — 指定した範囲のページにおけるページラベルの形式を設定する

説明

bool **HaruDoc::addPageLabel** (int `$first_page` , int `$style` , int `$first_num` [, string `$prefix`])

指定した範囲のページにおけるページラベルの形式を設定します。

パラメータ

first_page

ラベル付けの範囲に含む最初のページ。

style

番号の形式。以下のいずれかの値となります。

- **HaruPage::NUM_STYLE_DECIMAL** - ページラベルを十進数で表示します。
- **HaruPage::NUM_STYLE_UPPER_ROMAN** - ページラベルを大文字のローマ数字で表示します。
- **HaruPage::NUM_STYLE_LOWER_ROMAN** - ページラベルを小文字のローマ数字で表示します。
- **HaruPage::NUM_STYLE_UPPER_LETTER** - ページラベルを大文字 (A から Z まで) で表示します。
- **HaruPage::NUM_STYLE_LOWER_LETTERS** - ページラベルを小文字 (a から z まで) で表示します。

first_num

この範囲内の最初のページ番号。

prefix

ページラベルの接頭辞。オプションで、デフォルトでは空白。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::__construct

(PECL haru:0.0.1)

HaruDoc::__construct — 新しい HaruDoc のインスタンスを作成する

説明

void **HaruDoc::__construct** (void)

新しい HaruDoc のインスタンスを作成します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::createOutline

(PECL haru:0.0.1)

HaruDoc::createOutline — HaruOutline のインスタンスを作成する

説明

object **HaruDoc::createOutline** (string `$title` [, object `$parent_outline` [, object `$encoder`]])

新しい HaruOutline のインスタンスを作成し、それを返します。

パラメータ

title

新しいアウトラインオブジェクトの見出し。

parent_outline

有効な HaruOutline のインスタンスあるいは **NULL**。

encoder

有効な HaruEncoder のインスタンスあるいは **NULL**。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::getCurrentEncoder

(PECL haru:0.0.1)

HaruDoc::getCurrentEncoder — ドキュメントが現在使用している HaruEncoder を取得する

説明

object **HaruDoc::getCurrentEncoder** (void)

ドキュメントが現在使用している HaruEncoder、 あるいはエンコーダがまだ設定されていない場合に **FALSE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**setCurrentEncoder()**

HaruDoc::getCurrentPage

(PECL haru:0.0.1)

HaruDoc::getCurrentPage — ドキュメントの現在のページを返す

説明

object **HaruDoc::getCurrentPage** (void)

ドキュメントの現在のページを取得します。成功した場合に HaruPage のインスタンス、 現在のページが設定されていない場合に **FALSE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::getEncoder

(PECL haru:0.0.1)

HaruDoc::getEncoder — 指定したエンコーディングの HaruEncoder のインスタンスを取得する

説明

object **HaruDoc::getEncoder** (string `$encoding`)

指定したエンコーディングの HaruEncoder のインスタンスを返します。

パラメータ

encoding

エンコーディング名。使用できる値の一覧は 組み込みのエンコーディング を参照ください。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::setCurrentEncoder()
- HaruDoc::getCurrentEncoder()

HaruDoc::getFont

(PECL haru:0.0.1)

HaruDoc::getFont — HaruFont のインスタンスを取得する

説明

object **HaruDoc::getFont** (string `$fontname` [, string `$encoding`])

指定した *fontname* および *encoding* で HaruFont のインスタンスを作成し、それを返します。

パラメータ

fontname

フォントの名前。組み込みフォントの一覧は 組み込みのフォント を参照ください。 HaruDoc::loadTTF()、 HaruDoc::loadTTC() および HaruDoc::loadType1() で読み込んだフォントの名前を使用することもできます。

encoding

使用するエンコーディング。サポートするエンコーディングの一覧は 組み込みのエンコーディング を参照ください。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::setFontAndSize()
- HaruPage::getCurrentFont()

HaruDoc::getInfoAttr

(PECL haru:0.0.1)

HaruDoc::getInfoAttr — 指定したドキュメント属性の現在の値を取得する

説明

string **HaruDoc::getInfoAttr** (int `$type`)

指定したドキュメント属性の文字列値を返します。

パラメータ

type

属性の型。以下のいずれかを指定します。

- **HaruDoc::INFO_AUTHOR**
- **HaruDoc::INFO_CREATOR**
- **HaruDoc::INFO_TITLE**
- **HaruDoc::INFO_SUBJECT**
- **HaruDoc::INFO_KEYWORDS**
- **HaruDoc::INFO_CREATION_DATE**
- **HaruDoc::INFO_MOD_DATE**

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::setInfoAttr()
- HaruDoc::setInfoDateAttr()

HaruDoc::getPageLayout

(PECL haru:0.0.1)

HaruDoc::getPageLayout — 現在のページレイアウトを取得する

説明

int **HaruDoc::getPageLayout** (void)

現在ドキュメントに設定されているページレイアウトを返します。 ページレイアウトが設定されていない場合は **FALSE** を返します。 とりうる値の一覧は HaruDoc::setPageLayout() を参照ください。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::setPageLayout()

HaruDoc::getPageMode

(PECL haru:0.0.1)

HaruDoc::getPageMode — 現在のページモードを取得する

説明

int **HaruDoc::getPageMode** (void)

現在ドキュメントに設定されているページモードを返します。 とりうる値の一覧については HaruDoc::setPageMode() を参照ください。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::setPageMode()

HaruDoc::getStreamSize

(PECL haru:0.0.1)

HaruDoc::getStreamSize — 一時ストリームの大きさを取得する

説明

int **HaruDoc::getStreamSize** (void)

ドキュメントの一時ストリーム内のデータの大きさを返します。 ドキュメントを一時ストリームに保存していない場合は、大きさはゼロとなります。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::saveToStream()
- HaruDoc::resetStream()
- HaruDoc::readFromStream()

HaruDoc::insertPage

(PECL haru:0.0.1)

HaruDoc::insertPage — 指定したページの直前に新しいページを挿入する

説明

object **HaruDoc::insertPage** (object `$page`)

新しいページを作成し、指定したページの直前に挿入します。 新しい HaruPage のインスタンスを返します。

パラメータ

`page`

有効な HaruPage のインスタンス。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::addPage()

HaruDoc::loadJPEG

(PECL haru:0.0.1)

HaruDoc::loadJPEG — JPEG 画像を読み込み、新しい HaruImage のインスタンスを返す

説明

object **HaruDoc::loadJPEG** (string `$filename`)

指定した JPEG 画像を読み込み、新しい HaruImage のインスタンスを返します。

パラメータ

`filename`

有効な JPEG 画像ファイル。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::loadPNG()
- HaruDoc::loadRAW()

HaruDoc::loadPNG

(PECL haru:0.0.1)

HaruDoc::loadPNG — PNG 画像を読み込み、HaruImage のインスタンスを返す

説明

object **HaruDoc::loadPNG** (string `$filename` [, bool `$deferred`])

PNG 画像を読み込み、HaruImage のインスタンスを返します。 パラメータ *deferred* を **TRUE** にすると、 遅延データ読み込みを行います。この場合は、まず大きさと色のみが読み込まれます。 *deferred* のデフォルト値は **FALSE** です。

libharu が libpng サポートを含めずにビルドされている場合は、 このメソッドをコールすると例外が発生します。

パラメータ

filename

PNG 画像ファイルのファイル名。

deferred

データを即時に読み込まない。デフォルトは **FALSE** です。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::loadJPEG()
- HaruDoc::loadRAW()

HaruDoc::loadRaw

(PECL haru:0.0.1)

HaruDoc::loadRaw — RAW 画像を読み込み、HaruImage のインスタンスを返す

説明

object **HaruDoc::loadRaw** (string `$filename` , int `$width` , int `$height` , int `$color_space`)

RAW 画像を読み込み、新しい HaruImage のインスタンスを返します。

パラメータ

filename

RAW 画像ファイルの名前。

width

画像の幅。

height

画像の高さ。

color_space

画像の色空間。以下のいずれかの値となります。

- **HaruDoc::CS_DEVICE_GRAY**
- **HaruDoc::CS_DEVICE_RGB**
- **HaruDoc::CS_DEVICE_CMYK**

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::loadJPEG()
- HaruDoc::loadPNG()

HaruDoc::loadTTC

(PECL haru:0.0.1)

HaruDoc::loadTTC — TTC ファイルから指定したインデックスのフォントを読み込む

説明

string **HaruDoc::loadTTC** (string `$fontfile` , int `$index` [, bool `$embed`])

TrueType コレクション (TTC) ファイルから、 指定したインデックスの TrueType フォントを読み込みます。

パラメータ

`fontfile`

TrueType コレクションファイル。

`index`

コレクションファイル内でのフォントのインデックス。

`embed`

TRUE を指定すると、フォントのグリフデータを PDF ファイルに埋め込みます。それ以外の場合はマトリックスデータのみを含めます。 デフォルトは **FALSE** です。

返り値

成功時に、読み込んだフォントの名前を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::loadTTF()
- HaruDoc::loadType1()

HaruDoc::loadTTF

(PECL haru:0.0.1)

HaruDoc::loadTTF — TTF フォントファイルを読み込む

説明

string **HaruDoc::loadTTF** (string `$fontfile` [, bool `$embed`])

指定した TTF ファイルを読み込み、(オプションで) そのデータをドキュメントに埋め込みます。

パラメータ

`fontfile`

読み込む TTF ファイル。

`embed`

TRUE を指定すると、フォントのグリフデータを PDF ファイルに埋め込みます。それ以外の場合はマトリックスデータのみを含めます。 デフォルトは **FALSE** です。

返り値

成功時に、読み込んだフォントの名前を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::loadTTC()
- HaruDoc::loadType1()

HaruDoc::loadType1

(PECL haru:0.0.1)

HaruDoc::loadType1 — Type1 フォントを読み込む

説明

string **HaruDoc::loadType1** (string `$afmfile` [, string `$pfmfile`])

Type1 フォントを指定したファイルから読み込み、PDF ドキュメントに登録します。

パラメータ

`afmfile`

AFM ファイルへのパス。

`pfmfile`

オプションで指定する、PFA/PFB ファイルへのパス。 省略した場合は、フォントのグリフデータのみを PDF ドキュメントに埋め込みます。

返り値

成功時に、読み込んだフォントの名前を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::loadTTC()
- HaruDoc::loadTTF()

HaruDoc::output

(PECL haru:0.0.1)

HaruDoc::output — ドキュメントデータを出力バッファに書き出す

説明

bool **HaruDoc::output** (void)

ドキュメントデータを標準出力に書き出します。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**save()**

HaruDoc::readFromStream

(PECL haru:0.0.1)

HaruDoc::readFromStream — データを一時ストリームから読み込む

説明

string **HaruDoc::readFromStream** (int $bytes)

一時ストリームのデータを返します。パラメータ *bytes* で、読み込むバイト数を指定します。 ストリームのデータがこれより少ない場合にも指定したバイト数が読み込まれます。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**saveToStream()**
- HaruDoc::**resetStream()**
- HaruDoc::**getStreamSize()**

HaruDoc::resetError

(PECL haru:0.0.1)

HaruDoc::resetError — ドキュメントハンドルのエラーの状態をリセットする

説明

bool **HaruDoc::resetError** (void)

いったんエラーコードが設定されると、入出力処理を含む 大半の操作が実行できなくなります。 エラーへの対処を済ませたあとで処理を続けたい場合は、 この関数を実行してエラー状態をリセットする必要があります。

返り値

常に成功し、**TRUE** を返します。

HaruDoc::resetStream

(PECL haru:0.0.1)

HaruDoc::resetStream — 一時ストリームを巻き戻す

説明

bool **HaruDoc::resetStream** (void)

ドキュメントの一時ストリームを巻き戻します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::saveToStream()
- HaruDoc::getStreamSize()
- HaruDoc::readFromStream()

HaruDoc::save

(PECL haru:0.0.1)

HaruDoc::save — ドキュメントを指定したファイルに保存する

説明

bool **HaruDoc::save** (string $file)

ドキュメントを、指定したファイルに保存します。

パラメータ

file

ドキュメントを保存するファイル名。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::output()

HaruDoc::saveToStream

(PECL haru:0.0.1)

HaruDoc::saveToStream — ドキュメントを一時ストリームに保存する

説明

bool **HaruDoc::saveToStream** (void)

ドキュメントのデータを一時ストリームに保存します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::resetStream()
- HaruDoc::getStreamSize()
- HaruDoc::readFromStream()

HaruDoc::setCompressionMode

(PECL haru:0.0.1)

HaruDoc::setCompressionMode — ドキュメントの圧縮モードを設定する

説明

bool **HaruDoc::setCompressionMode** (int $mode)

ドキュメントの圧縮モードを定義します。libharu が Zlib サポートなしでコンパイルされている場合は、この関数は常に HaruException をスローします。

パラメータ

mode

使用する圧縮モード。この値は、以下のフラグの組み合わせとなります。

- **HaruDoc::COMP_NONE** - 一切圧縮しません。
- **HaruDoc::COMP_TEXT** - テキストデータを圧縮します。
- **HaruDoc::COMP_IMAGE** - 画像データを圧縮します。
- **HaruDoc::COMP_METADATA** - その他のデータ (フォントや cmap) を圧縮します。
- **HaruDoc::COMP_ALL** - すべてのデータを圧縮します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::setCurrentEncoder

(PECL haru:0.0.1)

HaruDoc::setCurrentEncoder — ドキュメントの現在のエンコーダを設定する

説明

bool **HaruDoc::setCurrentEncoder** (string $encoding)

ドキュメントで現在使用しているエンコーダを定義します。

パラメータ

encoding

使用するエンコーディングの名前。使用できる値の一覧は 組み込みのエンコーディング を参照ください。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::setEncryptionMode

(PECL haru:0.0.1)

HaruDoc::setEncryptionMode — ドキュメントの暗号化モードを設定する

説明

bool **HaruDoc::setEncryptionMode** (int $mode [, int $key_len])

ドキュメントの暗号化モードを定義します。 先にパスワードを設定しないと、暗号化モードを設定することはできません。

パラメータ

mode

使用する暗号化モード。以下のいずれかとなります。

- **HaruDoc::ENCRYPT_R2** - "revision2" アルゴリズムを使用します。
- **HaruDoc::ENCRYPT_R3** - "revision3" アルゴリズムを使用します。これを使用すると、PDF のバージョンは 1.4 になります。

key_len

暗号化キーの長さを表すバイト数。このパラメータはオプションで、 モードが **HaruDoc::ENCRYPT_R3** の場合にのみ使用します。 デフォルトの値は 5 (40 ビット) です。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**setPassword()**
- HaruDoc::**setPermission()**

HaruDoc::setInfoAttr

(PECL haru:0.0.1)

HaruDoc::setInfoAttr — ドキュメントの情報属性を設定する

説明

bool **HaruDoc::setInfoAttr** (int `$type` , string `$info`)

情報属性を定義します。ドキュメントの現在のエンコーディングを使用します。

パラメータ

type

属性の型。以下のいずれかとなります。

- **HaruDoc::INFO_AUTHOR**
- **HaruDoc::INFO_CREATOR**
- **HaruDoc::INFO_TITLE**
- **HaruDoc::INFO_SUBJECT**
- **HaruDoc::INFO_KEYWORDS**

info

属性の値。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**setInfoDateAttr()**

HaruDoc::setInfoDateAttr

(PECL haru:0.0.1)

HaruDoc::setInfoDateAttr — ドキュメントの情報属性に日付と時刻を設定する

説明

bool **HaruDoc::setInfoDateAttr** (int `$type` , int `$year` , int `$month` , int `$day` , int `$hour` , int `$min` , int `$sec` , string `$ind` , int `$off_hour` , int `$off_min`)

ドキュメントの日付や時刻の属性を設定します。

パラメータ

`type`

属性の型。以下のいずれかとなります。

- **`HaruDoc::INFO_CREATION_DATE`**
- **`HaruDoc::INFO_MOD_DATE`**

`year`

`month`

1 から 12 までの値。

`day`

1 から 31、30、29 あるいは 28 までの値 (月によって異なります)。

`hour`

0 から 23 までの値。

`min`

0 から 59 までの値。

`sec`

0 から 59 までの値。

`ind`

UTC とタイムゾーンとの関係。""、" "、"+"、"-" および "Z" のいずれか。

`off_hour`

`ind` が " " あるいは "" 以外の場合は 0 から 23 までの値が有効。それ以外の場合はこのパラメータは無視されます。

`off_min`

`ind` が " " あるいは "" 以外の場合は 0 から 59 までの値が有効。それ以外の場合はこのパラメータは無視されます。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**setInfoAttr()**

HaruDoc::setOpenAction

(PECL haru:0.0.1)

HaruDoc::setOpenAction — ドキュメントを開いたときにどのページを表示するかを定義する

説明

bool **HaruDoc::setOpenAction** (object `$destination`)

ドキュメントを開いたときに、どのページを表示するかを定義します。

パラメータ

`destination`

有効な HaruDestination のインスタンス。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::setPageLayout

(PECL haru:0.0.1)

HaruDoc::setPageLayout — ページをどのように表示するかを設定する

説明

bool **HaruDoc::setPageLayout** (int `$layout`)

ページをどのように表示するかを定義します。

パラメータ

`layout`

以下の値を使用できます。

- **`HaruDoc::PAGE_LAYOUT_SINGLE`** - 単一ページのみを表示します。
- **`HaruDoc::PAGE_LAYOUT_ONE_COLUMN`** - ひとつのカラムでページを表示します。
- **`HaruDoc::PAGE_LAYOUT_TWO_COLUMN_LEFT`** - ふたつのカラムでページを表示します。最初のページが左側となります。
- **`HaruDoc::PAGE_LAYOUT_TWO_COLUMN_RIGHT`** - ふたつのカラムでページを表示します。最初のページが右側となります。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**getPageLayout()**

HaruDoc::setPageMode

(PECL haru:0.0.1)

HaruDoc::setPageMode — ドキュメントをどのように表示するかを設定する

説明

bool **HaruDoc::setPageMode** (int $mode)

ドキュメントをどのように表示するかを定義します。

パラメータ

mode

以下の値を使用できます。

- **HaruDoc::PAGE_MODE_USE_NONE** - アウトラインやサムネイルを表示しません。
- **HaruDoc::PAGE_MODE_USE_OUTLINE** - アウトラインペインを表示します。
- **HaruDoc::PAGE_MODE_USE_THUMBS** - サムネイルペインを表示します。
- **HaruDoc::PAGE_MODE_FULL_SCREEN** - フルスクリーンモードで表示します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**getPageMode()**

HaruDoc::setPagesConfiguration

(PECL haru:0.0.1)

HaruDoc::setPagesConfiguration — ページ群単位のページ数を設定する

説明

bool **HaruDoc::setPagesConfiguration** (int $page_per_pages)

デフォルトでは、ドキュメントはひとつの pages オブジェクトを持っており、 これがすべてのページのルートとなります。各ページのオブジェクトは、 すべてこのオブジェクトの配下に作成されます。ひとつの pages オブジェクトが保持できるページ数は最大で 8191 までなので、 ドキュメントの最大ページ数は 8191 となります。しかし、パラメータ *page_per_pages* を指定することで、 これを変更することができます。ルートオブジェクトは 8191 を超える pages (ページではない) オブジェクトを保持することができ、 それぞれの pages が 8191 のページを保持するのです。 これにより、ドキュメントの最大ページ数は 8191**page_per_pages* となります。

パラメータ

page_per_pages

pages オブジェクトが保持できるページ数。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruDoc::setPassword

(PECL haru:0.0.1)

HaruDoc::setPassword — ドキュメントに所有者パスワードおよびユーザパスワードを設定する

説明

bool **HaruDoc::setPassword** (string $owner_password , string $user_password)

ドキュメントに所有者パスワードおよびユーザパスワードを設定します。 これらのパスワードを設定すると、ドキュメントの内容が暗号化されます。

パラメータ

owner_password

所有者のパスワード。これはドキュメントの使用権限を変更することができます。 空のパスワードは認められません。所有者のパスワードをユーザのパスワードと同じものにすることはできません。

user_password

ユーザのパスワード。空でもかまいません。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::setEncryptionMode()
- HaruDoc::setPermission()

HaruDoc::setPermission

(PECL haru:0.0.1)

HaruDoc::setPermission — ドキュメントの使用権限を設定する

説明

bool **HaruDoc::setPermission** (int `$permission`)

ドキュメントの使用権限を定義します。

パラメータ

permission

これらのフラグを組み合わせた値となります。

- **HaruDoc::ENABLE_READ** - ユーザはドキュメントを読むことができます。
- **HaruDoc::ENABLE_PRINT** - ユーザはドキュメントを印刷することができます。
- **HaruDoc::ENABLE_EDIT_ALL** - ユーザはアノテーションやフォームフィールド以外の ドキュメントの内容を編集することができます。
- **HaruDoc::ENABLE_COPY** - ユーザはドキュメントのテキストや画像をコピーすることができます。
- **HaruDoc::ENABLE_EDIT** - user can add or modify ユーザはドキュメントにアノテーションやフォームフィールドを追加したり 編集したりすることができます。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::setPassword()
- HaruDoc::setEncryptionMode()

HaruDoc::useCNSEncodings

(PECL haru:0.0.1)

HaruDoc::useCNSEncodings — 簡体字中国語エンコーディングを有効にする

説明

bool **HaruDoc::useCNSEncodings** (void)

簡体字中国語エンコーディングを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**useCNSFonts()**

HaruDoc::useCNSFonts

(PECL haru:0.0.1)

HaruDoc::useCNSFonts — 組み込みの簡体字中国語フォントを有効にする

説明

bool **HaruDoc::useCNSFonts** (void)

組み込みの簡体字中国語フォントを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**useCNSEncodings()**

HaruDoc::useCNTEncodings

(PECL haru:0.0.1)

HaruDoc::useCNTEncodings — 繁体字中国語エンコーディングを有効にする

説明

bool **HaruDoc::useCNTEncodings** (void)

繁体字中国語エンコーディングを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**useCNTFonts()**

HaruDoc::useCNTFonts

(PECL haru:0.0.1)

HaruDoc::useCNTFonts — 組み込みの繁体字中国語フォントを有効にする

説明

bool **HaruDoc::useCNTFonts** (void)

組み込みの繁体字中国語フォントを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**useCNTEncodings()**

HaruDoc::useJPEncodings

(PECL haru:0.0.1)

HaruDoc::useJPEncodings — 日本語エンコーディングを有効にする

説明

bool **HaruDoc::useJPEncodings** (void)

日本語エンコーディングを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::**useJPFonts()**

HaruDoc::useJPFonts

(PECL haru:0.0.1)

HaruDoc::useJPFonts — 組み込みの日本語フォントを有効にする

説明

bool **HaruDoc::useJPFonts** (void)

組み込みの日本語フォントを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::useJPEncodings()

HaruDoc::useKREncodings

(PECL haru:0.0.1)

HaruDoc::useKREncodings — 韓国/朝鮮語エンコーディングを有効にする

説明

bool **HaruDoc::useKREncodings** (void)

韓国/朝鮮語エンコーディングを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::useKRFonts()

HaruDoc::useKRFonts

(PECL haru:0.0.1)

HaruDoc::useKRFonts — 組み込みの韓国/朝鮮語フォントを有効にする

説明

bool **HaruDoc::useKRFonts** (void)

組み込みの韓国/朝鮮語フォントを有効にします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::useKREncodings()

HaruEncoder::getByteType

(PECL haru:0.0.1)

HaruEncoder::getByteType — テキスト内のバイトの型を取得する

説明

int **HaruEncoder::getByteType** (string `$text` , int `$index`)

テキスト内の指定した位置のバイトの型を返します。 結果は以下のいずれかの値となります。

- **HaruEncoder::BYTE_TYPE_SINGLE** - シングルバイト文字。
- **HaruEncoder::BYTE_TYPE_LEAD** - マルチバイト文字の先頭バイト。
- **HaruEncoder::BYTE_TYPE_TRAIL** - マルチバイト文字の後続バイト。
- **HaruEncoder::BYTE_TYPE_UNKNOWN** - エンコーダが無効、あるいはバイトの型の取得に失敗。

パラメータ

text

テキスト。

index

テキスト内の位置。

エラー / 例外

エラー時に HaruException をスローします。

HaruEncoder::getType

(PECL haru:0.0.1)

HaruEncoder::getType — エンコーダの型を取得する

説明

int **HaruEncoder::getType** (void)

エンコーダの型を返します。結果は以下のいずれかの値となります。

- **HaruEncoder::TYPE_SINGLE_BYTE** - エンコーダはシングルバイト文字用です。
- **HaruEncoder::TYPE_DOUBLE_BYTE** - エンコーダはマルチバイト文字用です。
- **HaruEncoder::TYPE_UNINITIALIZED** - エンコーダが初期化されていません。
- **HaruEncoder::UNKNOWN** - エンコーダが無効です。

エラー / 例外

エラー時に HaruException をスローします。

HaruEncoder::getUnicode

(PECL haru:0.0.1)

HaruEncoder::getUnicode — 指定した文字を unicode に変換する

説明

int **HaruEncoder::getUnicode** (int `$character`)

指定した文字を unicode に変換します。

パラメータ

character

変換する文字のコード。

エラー / 例外

エラー時に HaruException をスローします。

HaruEncoder::getWritingMode

(PECL haru:0.0.1)

HaruEncoder::getWritingMode — エンコーダの書き込みモードを取得する

説明

int **HaruEncoder::getWritingMode** (void)

エンコーダの書き込みモードを返します。 結果の値は次のようになります。

- **HaruEncoder::WMODE_HORIZONTAL** - 横書きモード。
- **HaruEncoder::WMODE_VERTICAL** - 縦書きモード。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getAscent

(PECL haru:0.0.1)

HaruFont::getAscent — フォントの垂直 ascent を取得する

説明

int **HaruFont::getAscent** (void)

フォントの垂直 ascent (ベースラインの上側の高さ) を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getCapHeight

(PECL haru:0.0.1)

HaruFont::getCapHeight — 大文字のベースラインからの距離を取得する

説明

int **HaruFont::getCapHeight** (void)

大文字のベースラインからの距離を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getDescent

(PECL haru:0.0.1)

HaruFont::getDescent — フォントの垂直 descent を取得する

説明

int **HaruFont::getDescent** (void)

フォントの垂直 descent (ベースラインの下側の高さ) を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getEncodingName

(PECL haru:0.0.1)

HaruFont::getEncodingName — エンコーディング名を取得する

説明

string **HaruFont::getEncodingName** (void)

フォントエンコーディングの名前を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getFontName

(PECL haru:0.0.1)

HaruFont::getFontName — フォント名を取得する

説明

string **HaruFont::getFontName** (void)

フォントの名前を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getTextWidth

(PECL haru:0.0.1)

HaruFont::getTextWidth — テキスト全体の幅、文字数、単語数および空白の数を取得する

説明

array **HaruFont::getTextWidth** (string `$text`)

指定したテキストについて、 テキスト全体の幅、文字数、単語数および空白の数を返します。

パラメータ

`text`

調べたいテキスト。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getUnicodeWidth

(PECL haru:0.0.1)

HaruFont::getUnicodeWidth — フォントの文字の幅を取得する

説明

int **HaruFont::getUnicodeWidth** (int `$character`)

フォントの文字の幅を返します。

パラメータ

character

文字のコード。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::getXHeight

(PECL haru:0.0.1)

HaruFont::getXHeight — 小文字のベースラインからの距離を取得する

説明

int **HaruFont::getXHeight** (void)

小文字のベースラインからの距離を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruFont::measureText

(PECL haru:0.0.1)

HaruFont::measureText — 指定した幅に収めることのできる文字数を計算する

説明

int **HaruFont::measureText** (string `$text` , float `$width` , float `$font_size` , float `$char_space` , float `$word_space` [, bool `$word_wrap`])

指定した幅に収めることのできる文字数を返します。

パラメータ

text

幅にあわせたいテキスト。

width

テキストを代入するエリアの幅。

font_size

フォントの大きさ。

char_space

文字の間隔。

word_space

単語の間隔。

word_wrap

このパラメータを **TRUE** にすると、この関数はワードラップを "エミュレート" し、現在の単語の途中で範囲の最後に達してしまう場合は その単語を含めません。デフォルトは **FALSE** です。

エラー / 例外

エラー時に HaruException をスローします。

HaruImage::getBitsPerComponent

(PECL haru:0.0.1)

HaruImage::getBitsPerComponent — 画像の各色コンポーネントで使用するビット数を取得する

説明

int **HaruImage::getBitsPerComponent** (void)

画像の各色コンポーネントで使用するビット数を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruImage::getColorSpace

(PECL haru:0.0.1)

HaruImage::getColorSpace — 色空間の名前を取得する

説明

string **HaruImage::getColorSpace** (void)

画像の色空間の名前を返します。名前は以下のいずれかとなります。

- "DeviceGray"
- "DeviceRGB"
- "DeviceCMYK"
- "Indexed"

エラー / 例外

エラー時に HaruException をスローします。

HaruImage::getHeight

(PECL haru:0.0.1)

HaruImage::getHeight — 画像の高さを取得する

説明

int **HaruImage::getHeight** (void)

画像の高さを返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruImage::getSize

(PECL haru:0.0.1)

HaruImage::getSize — 画像の大きさを取得する

説明

array **HaruImage::getSize** (void)

二つの要素 width および height を持つ配列を返します。 それぞれ、画像の対応する部分の大きさが含まれます。

エラー / 例外

エラー時に HaruException をスローします。

HaruImage::getWidth

(PECL haru:0.0.1)

HaruImage::getWidth — 画像の幅を取得する

説明

int **HaruImage::getWidth** (void)

画像の幅を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruImage::setColorMask

(PECL haru:0.0.1)

HaruImage::setColorMask — 画像の色マスクを設定する

説明

bool **HaruImage::setColorMask** (int $rmin , int $rmax , int $gmin , int $gmax , int $bmin , int $bmax)

画像の透過色を、RGB 値の範囲を用いて定義します。 この範囲内の色が透過色として表示されます。 画像の色空間は RGB でなければなりません。

パラメータ

rmin

赤の最小値。0 から 255 までの値でなければなりません。

rmax

赤の最大値。0 から 255 までの値でなければなりません。

gmin

緑の最小値。0 から 255 までの値でなければなりません。

gmax

緑の最大値。0 から 255 までの値でなければなりません。

bmin

青の最小値。0 から 255 までの値でなければなりません。

bmax

青の最大値。0 から 255 までの値でなければなりません。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruImage::setMaskImage

(PECL haru:0.0.1)

HaruImage::setMaskImage — 画像マスクを設定する

説明

bool **HaruImage::setMaskImage** (object `$mask_image`)

画像マスクとして使用する画像を設定します。 1 ビットのグレイスケール画像である必要があります。

パラメータ

`mask_image`

有効な HaruImage のインスタンス。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruOutline::setDestination

(PECL haru:0.0.1)

HaruOutline::setDestination — アウトラインの対象を設定する

説明

bool **HaruOutline::setDestination** (object `$destination`)

アウトラインがクリックされたときにジャンプする先の 対象オブジェクトを設定します。

パラメータ

`destination`

有効な HaruDestination のインスタンス。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruOutline::setOpened

(PECL haru:0.0.1)

HaruOutline::setOpened — アウトラインの初期状態を設定する

説明

bool **HaruOutline::setOpened** (bool $opened)

アウトラインを最初に表示した際に、 このノードを開くか開かないかを定義します。

パラメータ

opened

TRUE は開いた状態、**FALSE** は閉じた状態。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::arc

(PECL haru:0.0.1)

HaruPage::arc — 現在のパスに弧を追加する

説明

bool **HaruPage::arc** (float $x , float $y , float $ray , float $ang1 , float $ang2)

現在のパスに弧を追加します。

パラメータ

x

中心の水平座標。

y

中心の垂直座標。

ray

弧の半径。

ang1

開始角。

ang2

終了角。*ang1* より大きくなければなりません。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::beginText

(PECL haru:0.0.1)

HaruPage::beginText — テキストオブジェクトを開始し、テキストの現在位置を (0,0) に設定する

説明

bool **HaruPage::beginText** (void)

新しいテキストオブジェクトを開始し、テキストの現在位置を (0,0) に設定します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::circle

(PECL haru:0.0.1)

HaruPage::circle — 現在のパスに円を追加する

説明

bool **HaruPage::circle** (float $x , float $y , float $ray)

現在のパスに円を追加します。

パラメータ

x

中心の水平座標。

y

中心の垂直座標。

ray

円の半径。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::closePath

(PECL haru:0.0.1)

HaruPage::closePath — 現在の位置からパスの開始位置に直線を追加する

説明

bool **HaruPage::closePath** (void)

現在の位置からパスの開始位置に直線を追加します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::concat

(PECL haru:0.0.1)

HaruPage::concat — 現在のページの変換行列と指定した行列を連結する

説明

bool **HaruPage::concat** (float $a , float $b , float $c , float $d , float $x , float $y)

現在のページの変換行列と指定した行列を連結します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::createDestination

(PECL haru:0.0.1)

HaruPage::createDestination — 新しい HaruDestination のインスタンスを作成する

説明

object **HaruPage::createDestination** (void)

新しい HaruDestination のインスタンスを作成し、 それを返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::createLinkAnnotation

(PECL haru:0.0.1)

HaruPage::createLinkAnnotation — 新しい HaruAnnotation のインスタンスを作成する

説明

object **HaruPage::createLinkAnnotation** (array $rectangle , object $destination)

新しい HaruAnnotation のインスタンスを作成し、 それを返します。

パラメータ

rectangle

クリック可能な範囲を表す 4 つの座標の配列。

destination

有効な HaruDestination のインスタンス。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::createTextAnnotation

(PECL haru:0.0.1)

HaruPage::createTextAnnotation — 新しい HaruAnnotation のインスタンスを作成する

説明

object **HaruPage::createTextAnnotation** (array `$rectangle` , string `$text` [, object `$encoder`])

新しい HaruAnnotation のインスタンスを作成し、 それを返します。

パラメータ

`rectangle`

アノテーションの範囲を表す 4 つの座標の配列。

`text`

表示するテキスト。

`encoder`

オプションで指定する HaruEncoder のインスタンス。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::createURLAnnotation

(PECL haru:0.0.1)

HaruPage::createURLAnnotation — 新しい HaruAnnotation のインスタンスを作成する

説明

object **HaruPage::createURLAnnotation** (array `$rectangle` , string `$url`)

新しい HaruAnnotation のインスタンスを作成し、 それを返します。

パラメータ

`rectangle`

クリック可能な範囲を表す 4 つの座標の配列。

`url`

開く URL。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::curveTo2

(PECL haru:0.0.1)

HaruPage::curveTo2 — ベジエ曲線を現在のパスに追加する

説明

bool **HaruPage::curveTo2** (float $x2 , float $y2 , float $x3 , float $y3)

ベジエ曲線を現在のパスに追加します。 現在位置および点 (x2, y2) をベジエ曲線の制御点とし、 現在位置を点 (x3, y3) に移動します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::curveTo3

(PECL haru:0.0.1)

HaruPage::curveTo3 — ベジエ曲線を現在のパスに追加する

説明

bool **HaruPage::curveTo3** (float $x1 , float $y1 , float $x3 , float $y3)

ベジエ曲線を現在のパスに追加します。 点 (x1, y1) および点 (x3, y3) をベジエ曲線の制御点とし、 現在位置を点 (x3, y3) に移動します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::curveTo

(PECL haru:0.0.1)

HaruPage::curveTo — ベジエ曲線を現在のパスに追加する

説明

bool **HaruPage::curveTo** (float $x1 , float $y1 , float $x2 , float $y2 , float $x3 , float $y3)

ベジエ曲線を現在のパスに追加します。 点 (x1, y1) および点 (x2, y2) をベジエ曲線の制御点とし、 現在位置を点 (x3, y3) に移動します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::drawImage

(PECL haru:0.0.1)

HaruPage::drawImage — 画像をページに表示する

説明

bool **HaruPage::drawImage** (object $image , float $x , float $y , float $width , float $height)

画像をページに表示します。

パラメータ

image

有効な HaruImage のインスタンス。

x

画像を表示する範囲の左端。

y

画像を表示する範囲の下端。

width

画像の範囲の幅。

height

画像の範囲の高さ。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::ellipse

(PECL haru:0.0.1)

HaruPage::ellipse — 楕円を現在のパスに追加する

説明

bool **HaruPage::ellipse** (float $x , float $y , float $xray , float $yray)

楕円を現在のパスに追加します。

パラメータ

x

中心の水平座標。

y

中心の垂直座標。

xray

楕円の x 方向の半径。

yray

楕円の y 方向の半径。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::endPath

(PECL haru:0.0.1)

HaruPage::endPath — 塗りつぶしや描画を行わずに現在のパスオブジェクトを終了する

説明

bool **HaruPage::endPath** (void)

塗りつぶしや描画を行わずに現在のパスオブジェクトを終了します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::endText

(PECL haru:0.0.1)

HaruPage::endText — 現在のテキストオブジェクトを終了する

説明

bool **HaruPage::endText** (void)

現在のテキストオブジェクトを終了します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::eofill

(PECL haru:0.0.1)

HaruPage::eofill — 奇偶規則を使用して現在のパスを塗りつぶす

説明

bool **HaruPage::eofill** (void)

奇偶規則を使用して現在のパスを塗りつぶします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::eoFillStroke

(PECL haru:0.0.1)

HaruPage::eoFillStroke — 奇偶規則を使用して現在のパスを塗りつぶす

説明

bool **HaruPage::eoFillStroke** ([bool `$close_path`])

奇偶規則を使用して現在のパスを塗りつぶします。

パラメータ

`close_path`

オプションのパラメータです。**TRUE** に設定すると、 この関数は現在のパスを閉じます。デフォルトは **FALSE** です。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::fill

(PECL haru:0.0.1)

HaruPage::fill — ノンゼロワインディング規則を使用して現在のパスを塗りつぶす

説明

bool **HaruPage::fill** (void)

ノンゼロワインディング規則を使用して現在のパスを塗りつぶします。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::fillStroke

(PECL haru:0.0.1)

HaruPage::fillStroke — ノンゼロワインディング規則を使用して現在のパスを塗りつぶす

説明

bool **HaruPage::fillStroke** ([bool `$close_path`])

ノンゼロワインディング規則を使用して現在のパスを塗りつぶします。

パラメータ

`close_path`

オプションのパラメータです。**TRUE** に設定すると、 この関数は現在のパスを閉じます。デフォルトは **FALSE** です。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getCharSpace

(PECL haru:0.0.1)

HaruPage::getCharSpace — 現在の文字間隔を取得する

説明

float **HaruPage::getCharSpace** (void)

現在の文字間隔を取得します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getCMYKFill

(PECL haru:0.0.1)

HaruPage::getCMYKFill — 現在の塗りつぶし色を取得する

説明

array **HaruPage::getCMYKFill** (void)

現在の塗りつぶし色を、4 つの要素 ("c", "m", "y" および "k") からなる配列で返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getCMYKStroke

(PECL haru:0.0.1)

HaruPage::getCMYKStroke — 現在の描画色を取得する

説明

array **HaruPage::getCMYKStroke** (void)

現在の描画色を、4 つの要素 ("c", "m", "y" および "k") からなる配列で返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getCurrentFont

(PECL haru:0.0.1)

HaruPage::getCurrentFont — 現在使用中のフォントを取得する

説明

object **HaruPage::getCurrentFont** (void)

現在使用中のフォントを現す HaruFont のインスタンスを取得します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getCurrentFontSize

(PECL haru:0.0.1)

HaruPage::getCurrentFontSize — 現在のフォントのサイズを取得する

説明

float **HaruPage::getCurrentFontSize** (void)

現在のフォントのサイズを返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getCurrentPos

(PECL haru:0.0.1)

HaruPage::getCurrentPos — パスの描画用の現在の位置を取得する

説明

array **HaruPage::getCurrentPos** (void)

パスの描画用に使用する現在位置を、二つの要素 "x" と "y" からなる配列で返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getCurrentTextPos

(PECL haru:0.0.1)

HaruPage::getCurrentTextPos — テキストの印字用の現在の位置を取得する

説明

array **HaruPage::getCurrentTextPos** (void)

テキストの印字用に使用する現在位置を、二つの要素 "x" と "y" からなる配列で返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getDash

(PECL haru:0.0.1)

HaruPage::getDash — 現在の破線のパターンを取得する

説明

array **HaruPage::getDash** (void)

現在の破線のパターンを、二つの要素 "pattern" と "phase" からなる配列で返します。破線のパターンが設定されていない場合は **FALSE** を返します。
破線のパターンについての詳細な情報は HaruPage::**setDash()** を参照ください。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setDash()**

HaruPage::getFillingColorSpace

(PECL haru:0.0.1)

HaruPage::getFillingColorSpace — 現在の塗りつぶしの色空間を取得する

説明

int **HaruPage::getFillingColorSpace** (void)

現在の塗りつぶしの色空間を返します。 結果は、以下のいずれかの値となります。

- **HaruDoc::CS_DEVICE_GRAY**
- **HaruDoc::CS_DEVICE_RGB**
- **HaruDoc::CS_DEVICE_CMYK**
- **HaruDoc::CS_CAL_GRAY**
- **HaruDoc::CS_CAL_RGB**
- **HaruDoc::CS_LAB**
- **HaruDoc::CS_ICC_BASED**
- **HaruDoc::CS_SEPARATION**
- **HaruDoc::CS_DEVICE_N**
- **HaruDoc::CS_INDEXED**
- **HaruDoc::CS_PATTERN**

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getFlatness

(PECL haru:0.0.1)

HaruPage::getFlatness — ページの平坦度を取得する

説明

float **HaruPage::getFlatness** (void)

ページの平坦度を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setFlatness()**

HaruPage::getGMode

(PECL haru:0.0.1)

HaruPage::getGMode — 現在のグラフィックスモードを取得する

説明

int **HaruPage::getGMode** (void)

現在のグラフィックスモードを返します。 結果は次のいずれかの値となります。

- **HaruPage::GMODE_PAGE_DESCRIPTION**
- **HaruPage::GMODE_TEXT_OBJECT**
- **HaruPage::GMODE_PATH_OBJECT**
- **HaruPage::GMODE_CLIPPING_PATH**
- **HaruPage::GMODE_SHADING**
- **HaruPage::GMODE_INLINE_IMAGE**
- **HaruPage::GMODE_EXTERNAL_OBJECT**

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::getGrayFill

(PECL haru:0.0.1)

HaruPage::getGrayFill — 現在の塗りつぶし色を取得する

説明

float **HaruPage::getGrayFill** (void)

現在の塗りつぶし色を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setGrayFill()**

HaruPage::getGrayStroke

(PECL haru:0.0.1)

HaruPage::getGrayStroke — 現在の描画色を取得する

説明

float **HaruPage::getGrayStroke** (void)

現在の塗りつぶし色を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setGrayStroke()**

HaruPage::getHeight

(PECL haru:0.0.1)

HaruPage::getHeight — ページの高さを取得する

説明

float **HaruPage::getHeight** (void)

ページの高さを返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setHeight()**

HaruPage::getHorizontalScaling

(PECL haru:0.0.1)

HaruPage::getHorizontalScaling — 現在の水平方向の拡大率を取得する

説明

float **HaruPage::getHorizontalScaling** (void)

現在の水平方向の拡大率を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setHorizontalScaling()**

HaruPage::getLineCap

(PECL haru:0.0.1)

HaruPage::getLineCap — 現在のパスの終端の形式を取得する

説明

int **HaruPage::getLineCap** (void)

現在のパスの終端の形式を返します。結果の値は以下のいずれかとなります。

- **HaruPage::BUTT_END** - パスの端点で、線を角型に打ち切ります。
- **HaruPage::ROUND_END** - パスの端点が、その点を中心とする半円となります。
- **HaruPage::PROJECTING_SCUARE_END** - パスの端点から、線幅の半分まで超えて線を続けます。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setLineCap()**

HaruPage::getLineJoin

(PECL haru:0.0.1)

HaruPage::getLineJoin — 現在のパスの角の形式を取得する

説明

int **HaruPage::getLineJoin** (void)

現在のパスの角の形式を返します。 結果の値は以下のいずれかとなります。

- **HaruPage::MITER_JOIN**
- **HaruPage::ROUND_JOIN**
- **HaruPage::BEVEL_JOIN**

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setLineJoin()**

HaruPage::getLineWidth

(PECL haru:0.0.1)

HaruPage::getLineWidth — 現在の線幅を取得する

説明

float **HaruPage::getLineWidth** (void)

現在の線幅を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setLineWidth()**

HaruPage::getMiterLimit

(PECL haru:0.0.1)

HaruPage::getMiterLimit — マイターリミットの値を取得する

説明

float **HaruPage::getMiterLimit** (void)

マイターリミットの値を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setMiterLimit()**

HaruPage::getRGBFill

(PECL haru:0.0.1)

HaruPage::getRGBFill — 現在の塗りつぶし色を取得する

説明

array **HaruPage::getRGBFill** (void)

現在の塗りつぶし色を、三つの要素 "r"、"g" および "b" からなる配列で返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setRGBFill()**

HaruPage::getRGBStroke

(PECL haru:0.0.1)

HaruPage::getRGBStroke — 現在の描画色を取得する

説明

array **HaruPage::getRGBStroke** (void)

現在の描画色を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setRGBStroke()**

HaruPage::getStrokingColorSpace

(PECL haru:0.0.1)

HaruPage::getStrokingColorSpace — 現在の描画の色空間を取得する

説明

int **HaruPage::getStrokingColorSpace** (void)

現在の描画の色空間を返します。 返り値の一覧は HaruPage::**getFillingColorSpace()** と同じです。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getFillingColorSpace()**

HaruPage::getTextLeading

(PECL haru:0.0.1)

HaruPage::getTextLeading — 現在の行間隔の値を取得する

説明

float **HaruPage::getTextLeading** (void)

現在の行間隔の値を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setTextLeading()**

HaruPage::getTextMatrix

(PECL haru:0.0.1)

HaruPage::getTextMatrix — そのページの現在のテキスト変換行列を取得する

説明

array **HaruPage::getTextMatrix** (void)

そのページの現在のテキスト変換行列を、 "a"、"b"、"c"、"d"、"x" および "y" の 6 つの要素からなる配列で返します。

エラー / 例外

エラー時に HaruException を返します。

参考

- HaruPage::**setTextMatrix()**

HaruPage::getTextRenderingMode

(PECL haru:0.0.1)

HaruPage::getTextRenderingMode — 現在のテキストのレンダリングモードを取得する

説明

int **HaruPage::getTextRenderingMode** (void)

現在のテキストのレンダリングモードを返します。 結果の値は次のいずれかとなります。

- **HaruPage::FILL**
- **HaruPage::STROKE**
- **HaruPage::FILL_THEN_STROKE**
- **HaruPage::INVISIBLE**
- **HaruPage::FILL_CLIPPING**
- **HaruPage::STROKE_CLIPPING**
- **HaruPage::FILL_STROKE_CLIPPING**
- **HaruPage::CLIPPING**

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setTextRenderingMode()**

HaruPage::getTextRise

(PECL haru:0.0.1)

HaruPage::getTextRise — 現在のテキストライズの値を取得する

説明

float **HaruPage::getTextRise** (void)

現在のテキストライズの値を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setTextRise()**

HaruPage::getTextWidth

(PECL haru:0.0.1)

HaruPage::getTextWidth — 現在のフォントサイズ、文字エンコーディングおよび単語間隔を使用してテキストの幅を取得する

説明

float **HaruPage::getTextWidth** (string `$text`)

現在のフォントサイズ、文字エンコーディングおよび単語間隔を使用して テキストの幅を返します。

パラメータ

`text`

調べたいテキスト。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**measureText()**

HaruPage::getTransMatrix

(PECL haru:0.0.1)

HaruPage::getTransMatrix — そのページの現在の変換行列を取得する

説明

array **HaruPage::getTransMatrix** (void)

そのページの現在の変換行列を、 "a"、"b"、"c"、"d"、"x" および "y" の 6 つの要素からなる配列で返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**concat()**

HaruPage::getWidth

(PECL haru:0.0.1)

HaruPage::getWidth — ページの幅を取得する

説明

float **HaruPage::getWidth** (void)

ページの幅を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setWidth()**

HaruPage::getWordSpace

(PECL haru:0.0.1)

HaruPage::getWordSpace — 現在の単語間隔の値を取得する

説明

float **HaruPage::getWordSpace** (void)

現在の単語間隔の値を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setWordSpace()**

HaruPage::lineTo

(PECL haru:0.0.1)

HaruPage::lineTo — 現在位置から指定した位置まで直線を描画する

説明

bool **HaruPage::lineTo** (float $x , float $y)

現在位置から指定した位置まで直線を描画します。

パラメータ

x

y

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::curveTo()
- HaruPage::curveTo2()
- HaruPage::curveTo3()

HaruPage::measureText

(PECL haru:0.0.1)

HaruPage::measureText — 指定した幅の中に配置できる文字の数を計算する

説明

int **HaruPage::measureText** (string `$text` , float `$width` [, bool `$wordwrap`])

指定した幅の中に配置できる文字の数を計算します。

パラメータ

`text`

調べたいテキスト。

`width`

テキストを配置する範囲の幅。

`wordwrap`

このパラメータを **TRUE** に設定すると、ワードラップ処理を "エミュレート" します。現在の単語の途中で範囲の最後に到達してしまう場合に その単語を含めません。デフォルトは **FALSE** です。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruFont::measureText()

HaruPage::moveTextPos

(PECL haru:0.0.1)

HaruPage::moveTextPos — テキストの位置を、指定したオフセットに移動する

説明

bool **HaruPage::moveTextPos** (float `$x` , float `$y` [, bool `$set_leading`])

テキストの位置を、指定したオフセットに移動します。 現在の行の開始位置が (`x1`, `y1`) の場合、次の行の開始位置は (`x1` + *`x`* , `y1` + *`y`*) となります。オプションのパラメータ *`set_leading`* が **TRUE** の場合は、テキストのリーディングが *`-y`* となります。*`set_leading`* のデフォルト値は **FALSE** です。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::moveToNextLine()

HaruPage::moveTo

(PECL haru:0.0.1)

HaruPage::moveTo — 新しい描画パスの開始位置を設定する

説明

bool **HaruPage::moveTo** (float $x , float $y)

新しい描画パスの開始位置を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::moveToNextLine

(PECL haru:0.0.1)

HaruPage::moveToNextLine — テキストの位置を次の行の行頭に移動する

説明

bool **HaruPage::moveToNextLine** (void)

テキストの位置を次の行の行頭に移動します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**moveTextPos()**

HaruPage::rectangle

(PECL haru:0.0.1)

HaruPage::rectangle — 現在のパスに矩形を追加する

説明

bool **HaruPage::rectangle** (float $x , float $y , float $width , float $height)

現在の描画パスに矩形を追加します。

パラメータ

x

矩形の左端。

y

矩形の下端。

width

矩形の幅。

height

矩形の高さ。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::setCharSpace

(PECL haru:0.0.1)

HaruPage::setCharSpace — ページの文字間隔を設定する

説明

bool **HaruPage::setCharSpace** (float $char_space)

ページの文字間隔を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getCharSpace()**

HaruPage::setCMYKFill

(PECL haru:0.0.1)

HaruPage::setCMYKFill — ページの塗りつぶし色を設定する

説明

bool **HaruPage::setCMYKFill** (float $c , float $m , float $y , float $k)

ページの塗りつぶし色を定義します。

パラメータ

c

m

y

k

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getCMYKFill()

HaruPage::setCMYKStroke

(PECL haru:0.0.1)

HaruPage::setCMYKStroke — ページの描画色を設定する

説明

bool **HaruPage::setCMYKStroke** (float $c , float $m , float $y , float $k)

ページの描画色を設定します。

パラメータ

c

m

y

k

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getCMYKStroke()

HaruPage::setDash

(PECL haru:0.0.1)

HaruPage::setDash — ページの破線パターンを設定する

説明

bool **HaruPage::setDash** (array $pattern , int $phase)

ページの破線パターンを定義します。

パラメータ

pattern

最大 8 要素からなる配列。そのページの破線で使用する 線と空白のパターンを指定します。

phase

パターンをどこから開始するか。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getDash()

HaruPage::setFlatness

(PECL haru:0.0.1)

HaruPage::setFlatness — ページの平坦度を設定する

説明

bool **HaruPage::setFlatness** (float `$flatness`)

ページの平坦度を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getFlatness()

HaruPage::setFontAndSize

(PECL haru:0.0.1)

HaruPage::setFontAndSize — ページのフォントおよびフォントサイズを設定する

説明

bool **HaruPage::setFontAndSize** (object `$font` , float `$size`)

ページの現在のフォントおよびフォントサイズを定義します。

パラメータ

`font`

有効な HaruFont のインスタンス。

`size`

フォントのサイズ。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruDoc::getFont()

HaruPage::setGrayFill

(PECL haru:0.0.1)

HaruPage::setGrayFill — ページの塗りつぶし色を設定する

説明

bool **HaruPage::setGrayFill** (float $value)

ページの塗りつぶし色を定義します。

パラメータ

value

0 から 1 までのグレイ値。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getGrayFill()**

HaruPage::setGrayStroke

(PECL haru:0.0.1)

HaruPage::setGrayStroke — ページの描画色を設定する

説明

bool **HaruPage::setGrayStroke** (float $value)

ページの描画色を定義します。

パラメータ

value

0 から 1 までのグレイ値。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getGrayStroke()**

HaruPage::setHeight

(PECL haru:0.0.1)

HaruPage::setHeight — ページの高さを設定する

説明

bool **HaruPage::setHeight** (float `$height`)

ページの高さを定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getHeight()**

HaruPage::setHorizontalScaling

(PECL haru:0.0.1)

HaruPage::setHorizontalScaling — ページの横方向の拡大率を設定する

説明

bool **HaruPage::setHorizontalScaling** (float `$scaling`)

ページ上でテキストを表示する際の水平方向の拡大率を定義します。 初期値は 100 です。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getHorizontalScaling()**

HaruPage::setLineCap

(PECL haru:0.0.1)

HaruPage::setLineCap — パスの終端の形式を設定する

説明

bool **HaruPage::setLineCap** (int `$cap`)

パスの終端に使用する形式を定義します。

パラメータ

cap

次のいずれかの値でなければなりません。

- **HaruPage::BUTT_END** - パスの端点で、線を角型に打ち切ります。
- **HaruPage::ROUND_END** - パスの端点が、その点を中心とする半円となります。
- **HaruPage::PROJECTING_SCUARE_END** - パスの端点から、線幅の半分まで超えて線を続けます。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getLineCap()

HaruPage::setLineJoin

(PECL haru:0.0.1)

HaruPage::setLineJoin — そのページのパスの角の形式を設定する

説明

bool **HaruPage::setLineJoin** (int `$join`)

そのページのパスの角の形式を定義します。

パラメータ

`join`

以下のいずれかの値でなければなりません。

- **HaruPage::MITER_JOIN**
- **HaruPage::ROUND_JOIN**
- **HaruPage::BEVEL_JOIN**

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getLineJoin()

HaruPage::setLineWidth

(PECL haru:0.0.1)

HaruPage::setLineWidth — ページの線幅を設定する

説明

bool **HaruPage::setLineWidth** (float `$width`)

ページの線幅を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getLineWidth()

HaruPage::setMiterLimit

(PECL haru:0.0.1)

HaruPage::setMiterLimit — ページのマイターリミットの現在値を設定する

説明

bool **HaruPage::setMiterLimit** (float `$limit`)

ページのマイターリミットの現在値を設定します。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getMiterLimit()

HaruPage::setRGBFill

(PECL haru:0.0.1)

HaruPage::setRGBFill — ページの塗りつぶし色を設定する

説明

bool **HaruPage::setRGBFill** (float `$r` , float `$g` , float `$b`)

ページの塗りつぶし色を定義します。すべての値は 0 から 1 までの間でなければなりません。

パラメータ

r

g

b

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::getRGBFill()

HaruPage::setRGBStroke

(PECL haru:0.0.1)

HaruPage::setRGBStroke — ページの描画色を設定する

説明

bool **HaruPage::setRGBStroke** (float $r , float $g , float $b)

ページの描画色を定義します。すべての値は 0 から 1 までの間でなければなりません。

パラメータ

r

g

b

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getRGBStroke()**

HaruPage::setRotate

(PECL haru:0.0.1)

HaruPage::setRotate — ページの回転角度を設定する

説明

bool **HaruPage::setRotate** (int $angle)

ページの回転角度を定義します。

パラメータ

angle

90 度の倍数である必要があります。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::setSize

(PECL haru:0.0.1)

HaruPage::setSize — ページのサイズおよび方向を設定する

説明

bool **HaruPage::setSize** (int $size , int $direction)

ページのサイズおよび方向を、定義済みのフォーマットに変更します。

パラメータ

size

次のいずれかの値でなければなりません。

- **HaruPage::SIZE_LETTER**
- **HaruPage::SIZE_LEGAL**
- **HaruPage::SIZE_A3**
- **HaruPage::SIZE_A4**
- **HaruPage::SIZE_A5**
- **HaruPage::SIZE_B4**
- **HaruPage::SIZE_B5**
- **HaruPage::SIZE_EXECUTIVE**
- **HaruPage::SIZE_US4x6**
- **HaruPage::SIZE_US4x8**
- **HaruPage::SIZE_US5x7**
- **HaruPage::SIZE_COMM10**

direction

次のいずれかの値でなければなりません。

- **HaruPage::PORTRAIT**
- **HaruPage::LANDSCAPE**

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**setWidth()**
- HaruPage::**setHeight()**

HaruPage::setSlideShow

(PECL haru:0.0.1)

HaruPage::setSlideShow — ページの移動方式を設定する

説明

bool **HaruPage::setSlideShow** (int `$type` , float `$disp_time` , float `$trans_time`)

ページの移動方式を設定します。

パラメータ

type

以下のいずれかの値でなければなりません。

- **HaruPage::TS_WIPE_RIGHT**
- **HaruPage::TS_WIPE_LEFT**
- **HaruPage::TS_WIPE_UP**
- **HaruPage::TS_WIPE_DOWN**
- **HaruPage::TS_BARN_DOORS_HORIZONTAL_OUT**
- **HaruPage::TS_BARN_DOORS_HORIZONTAL_IN**
- **HaruPage::TS_BARN_DOORS_VERTICAL_OUT**
- **HaruPage::TS_BARN_DOORS_VERTICAL_IN**
- **HaruPage::TS_BOX_OUT**
- **HaruPage::TS_BOX_IN**
- **HaruPage::TS_BLINDS_HORIZONTAL**
- **HaruPage::TS_BLINDS_VERTICAL**
- **HaruPage::TS_DISSOLVE**
- **HaruPage::TS_GLITTER_RIGHT**

- **HaruPage::TS_GLITTER_DOWN**
- **HaruPage::TS_GLITTER_TOP_LEFT_TO_BOTTOM_RIGHT**
- **HaruPage::TS_REPLACE**

disp_time

ページの表示時間を表す秒数。

trans_time

ページ移動処理を行う秒数。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::setTextLeading

(PECL haru:0.0.1)

HaruPage::setTextLeading — ページのテキストリーディング (行間隔) を設定する

説明

bool **HaruPage::setTextLeading** (float $text_leading)

ページの行間隔を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getTextLeading()**

HaruPage::setTextMatrix

(PECL haru:0.0.1)

HaruPage::setTextMatrix — そのページの現在のテキスト変換行列を設定する

説明

bool **HaruPage::setTextMatrix** (float $a , float $b , float $c , float $d , float $x , float $y)

そのページのテキスト変換行列を定義します。

パラメータ

a

b

c

d

x

y

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getTextMatrix()**

HaruPage::setTextRenderingMode

(PECL haru:0.0.1)

HaruPage::setTextRenderingMode — そのページの現在のテキストのレンダリングモードを設定する

説明

bool **HaruPage::setTextRenderingMode** (int $mode)

そのページのテキストのレンダリングモードを定義します。

パラメータ

mode

以下のいずれかの値でなければなりません。

- **HaruPage::FILL**
- **HaruPage::STROKE**
- **HaruPage::FILL_THEN_STROKE**
- **HaruPage::INVISIBLE**
- **HaruPage::FILL_CLIPPING**
- **HaruPage::STROKE_CLIPPING**
- **HaruPage::FILL_STROKE_CLIPPING**
- **HaruPage::CLIPPING**

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getTextRenderingMode()**

HaruPage::setTextRise

(PECL haru:0.0.1)

HaruPage::setTextRise — 現在のテキストライズの値を設定する

説明

bool **HaruPage::setTextRise** (float $rise)

現在のテキストライズの値を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getTextRise()**

HaruPage::setWidth

(PECL haru:0.0.1)

HaruPage::setWidth — ページの幅を設定する

説明

bool **HaruPage::setWidth** (float `$width`)

ページの幅を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getWidth()**

HaruPage::setWordSpace

(PECL haru:0.0.1)

HaruPage::setWordSpace — ページの単語間隔を設定する

説明

bool **HaruPage::setWordSpace** (float `$word_space`)

ページの単語間隔を定義します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**getWordSpace()**

HaruPage::showText

(PECL haru:0.0.1)

HaruPage::showText — ページの現在位置にテキストを表示する

説明

bool **HaruPage::showText** (string `$text`)

ページの現在位置にテキストを表示します。

パラメータ

`text`

表示させるテキスト。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**showTextNextLine()**
- HaruPage::**textOut()**

HaruPage::showTextNextLine

(PECL haru:0.0.1)

HaruPage::showTextNextLine — 現在位置を次の行の行頭に移動してテキストを表示する

説明

bool **HaruPage::showTextNextLine** (string `$text` [, float `$word_space` [, float `$char_space`]])

現在位置を次の行の行頭に移動してテキストを表示します。

パラメータ

`text`

表示させるテキスト。

`word_space`

単語の間隔。

`char_space`

文字の間隔。

返り値

成功した場合に **`TRUE`** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**showText()**
- HaruPage::**textOut()**

HaruPage::stroke

(PECL haru:0.0.1)

HaruPage::stroke — 現在のパスを描画する

説明

bool **HaruPage::stroke** ([bool `$close_path`])

現在のパスを描画します。

パラメータ

`close_path`

TRUE の場合は現在のパスを閉じます。デフォルトは **FALSE** です。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

HaruPage::textOut

(PECL haru:0.0.1)

HaruPage::textOut — テキストを指定した位置に表示する

説明

bool **HaruPage::textOut** (float `$x` , float `$y` , string `$text`)

テキストを指定した位置に表示します。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**showTextNextLine()**
- HaruPage::**showText()**

HaruPage::textRect

(PECL haru:0.0.1)

HaruPage::textRect — 指定した領域内にテキストを表示する

説明

bool **HaruPage::textRect** (float `$left` , float `$top` , float `$right` , float `$bottom` , string `$text` [, int `$align`])

指定した領域内にテキストを表示します。

パラメータ

`left`

テキストの範囲の左端。

`top`

テキストの範囲の上端。

right

テキストの範囲の右端。

bottom

テキストの範囲の下端。

text

表示させたいテキスト。

align

テキストの配置。次のいずれかの値でなければなりません。

- **HaruPage::TALING_LEFT**
- **HaruPage::TALING_RIGHT**
- **HaruPage::TALING_CENTER**
- **HaruPage::TALING_JUSTIFY**

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に HaruException をスローします。

参考

- HaruPage::**showTextNextLine()**
- HaruPage::**showText()**
- HaruPage::**textOut()**

目次

- HaruDoc::getFont — HaruFont のインスタンスを取得する
- HaruDoc::getInfoAttr — 指定したドキュメント属性の現在の値を取得する
- HaruDoc::getPageLayout — 現在のページレイアウトを取得する
- HaruDoc::getPageMode — 現在のページモードを取得する
- HaruDoc::getStreamSize — 一時ストリームの大きさを取得する
- HaruDoc::insertPage — 指定したページの直前に新しいページを挿入する
- HaruDoc::loadJPEG — JPEG 画像を読み込み、新しい HaruImage のインスタンスを返す
- HaruDoc::loadPNG — PNG 画像を読み込み、HaruImage のインスタンスを返す
- HaruDoc::loadRaw — RAW 画像を読み込み、HaruImage のインスタンスを返す
- HaruDoc::loadTTC — TTC ファイルから指定したインデックスのフォントを読み込む
- HaruDoc::loadTTF — TTF フォントファイルを読み込む
- HaruDoc::loadType1 — Type1 フォントを読み込む
- HaruDoc::output — ドキュメントデータを出力バッファに書き出す
- HaruDoc::readFromStream — データを一時ストリームから読み込む
- HaruDoc::resetError — ドキュメントハンドルのエラーの状態をリセットする
- HaruDoc::resetStream — 一時ストリームを巻き戻す
- HaruDoc::save — ドキュメントを指定したファイルに保存する
- HaruDoc::saveToStream — ドキュメントを一時ストリームに保存する
- HaruDoc::setCompressionMode — ドキュメントの圧縮モードを設定する
- HaruDoc::setCurrentEncoder — ドキュメントの現在のエンコーダを設定する
- HaruDoc::setEncryptionMode — ドキュメントの暗号化モードを設定する
- HaruDoc::setInfoAttr — ドキュメントの情報属性を設定する
- HaruDoc::setInfoDateAttr — ドキュメントの情報属性に日付と時刻を設定する
- HaruDoc::setOpenAction — ドキュメントを開いたときにどのページを表示するかを定義する
- HaruDoc::setPageLayout — ページをどのように表示するかを設定する
- HaruDoc::setPageMode — ドキュメントをどのように表示するかを設定する
- HaruDoc::setPagesConfiguration — ページ群単位のページ数を設定する
- HaruDoc::setPassword — ドキュメントに所有者パスワードおよびユーザパスワードを設定する
- HaruDoc::setPermission — ドキュメントの使用権限を設定する
- HaruDoc::useCNSEncodings — 簡体字中国語エンコーディングを有効にする
- HaruDoc::useCNSFonts — 組み込みの簡体字中国語フォントを有効にする
- HaruDoc::useCNTEncodings — 繁体字中国語エンコーディングを有効にする
- HaruDoc::useCNTFonts — 組み込みの繁体字中国語フォントを有効にする
- HaruDoc::useJPEncodings — 日本語エンコーディングを有効にする
- HaruDoc::useJPFonts — 組み込みの日本語フォントを有効にする
- HaruDoc::useKREncodings — 韓国/朝鮮語エンコーディングを有効にする
- HaruDoc::useKRFonts — 組み込みの韓国/朝鮮語フォントを有効にする
- HaruEncoder::getByteType — テキスト内のバイトの型を取得する
- HaruEncoder::getType — エンコーダの型を取得する
- HaruEncoder::getUnicode — 指定した文字を unicode に変換する
- HaruEncoder::getWritingMode — エンコーダの書き込みモードを取得する
- HaruFont::getAscent — フォントの垂直 ascent を取得する
- HaruFont::getCapHeight — 大文字のベースラインからの距離を取得する
- HaruFont::getDescent — フォントの垂直 descent を取得する
- HaruFont::getEncodingName — エンコーディング名を取得する
- HaruFont::getFontName — フォント名を取得する
- HaruFont::getTextWidth — テキスト全体の幅、文字数、単語数および空白の数を取得する
- HaruFont::getUnicodeWidth — フォントの文字の幅を取得する
- HaruFont::getXHeight — 小文字のベースラインからの距離を取得する
- HaruFont::measureText — 指定した幅に収めることのできる文字数を計算する
- HaruImage::getBitsPerComponent — 画像の各色コンポーネントで使用するビット数を取得する
- HaruImage::getColorSpace — 色空間の名前を取得する
- HaruImage::getHeight — 画像の高さを取得する
- HaruImage::getSize — 画像の大きさを取得する
- HaruImage::getWidth — 画像の幅を取得する
- HaruImage::setColorMask — 画像の色マスクを設定する
- HaruImage::setMaskImage — 画像マスクを設定する
- HaruOutline::setDestination — アウトラインの対象を設定する
- HaruOutline::setOpened — アウトラインの初期状態を設定する
- HaruPage::arc — 現在のパスに弧を追加する
- HaruPage::beginText — テキストオブジェクトを開始し、テキストの現在位置を (0,0) に設定する
- HaruPage::circle — 現在のパスに円を追加する
- HaruPage::closePath — 現在の位置からパスの開始位置に直線を追加する
- HaruPage::concat — 現在のページの変換行列と指定した行列を連結する

- HaruPage::createDestination — 新しい HaruDestination のインスタンスを作成する
- HaruPage::createLinkAnnotation — 新しい HaruAnnotation のインスタンスを作成する
- HaruPage::createTextAnnotation — 新しい HaruAnnotation のインスタンスを作成する
- HaruPage::createURLAnnotation — 新しい HaruAnnotation のインスタンスを作成する
- HaruPage::curveTo2 — ベジエ曲線を現在のパスに追加する
- HaruPage::curveTo3 — ベジエ曲線を現在のパスに追加する
- HaruPage::curveTo — ベジエ曲線を現在のパスに追加する
- HaruPage::drawImage — 画像をページに表示する
- HaruPage::ellipse — 楕円を現在のパスに追加する
- HaruPage::endPath — 塗りつぶしや描画を行わずに現在のパスオブジェクトを終了する
- HaruPage::endText — 現在のテキストオブジェクトを終了する
- HaruPage::eofill — 奇偶規則を使用して現在のパスを塗りつぶす
- HaruPage::eoFillStroke — 奇偶規則を使用して現在のパスを塗りつぶす
- HaruPage::fill — ノンゼロワインディング規則を使用して現在のパスを塗りつぶす
- HaruPage::fillStroke — ノンゼロワインディング規則を使用して現在のパスを塗りつぶす
- HaruPage::getCharSpace — 現在の文字間隔を取得する
- HaruPage::getCMYKFill — 現在の塗りつぶし色を取得する
- HaruPage::getCMYKStroke — 現在の描画色を取得する
- HaruPage::getCurrentFont — 現在使用中のフォントを取得する
- HaruPage::getCurrentFontSize — 現在のフォントのサイズを取得する
- HaruPage::getCurrentPos — パスの描画用の現在の位置を取得する
- HaruPage::getCurrentTextPos — テキストの印字用の現在の位置を取得する
- HaruPage::getDash — 現在の破線のパターンを取得する
- HaruPage::getFillingColorSpace — 現在の塗りつぶしの色空間を取得する
- HaruPage::getFlatness — ページの平坦度を取得する
- HaruPage::getGMode — 現在のグラフィックスモードを取得する
- HaruPage::getGrayFill — 現在の塗りつぶし色を取得する
- HaruPage::getGrayStroke — 現在の描画色を取得する
- HaruPage::getHeight — ページの高さを取得する
- HaruPage::getHorizontalScaling — 現在の水平方向の拡大率を取得する
- HaruPage::getLineCap — 現在のパスの終端の形式を取得する
- HaruPage::getLineJoin — 現在のパスの角の形式を取得する
- HaruPage::getLineWidth — 現在の線幅を取得する
- HaruPage::getMiterLimit — マイターリミットの値を取得する
- HaruPage::getRGBFill — 現在の塗りつぶし色を取得する
- HaruPage::getRGBStroke — 現在の描画色を取得する
- HaruPage::getStrokingColorSpace — 現在の描画の色空間を取得する
- HaruPage::getTextLeading — 現在の行間隔の値を取得する
- HaruPage::getTextMatrix — そのページの現在のテキスト変換行列を取得する
- HaruPage::getTextRenderingMode — 現在のテキストのレンダリングモードを取得する
- HaruPage::getTextRise — 現在のテキストライズの値を取得する
- HaruPage::getTextWidth — 現在のフォントサイズ、文字エンコーディングおよび単語間隔を使用してテキストの幅を取得する
- HaruPage::getTransMatrix — そのページの現在の変換行列を取得する
- HaruPage::getWidth — ページの幅を取得する
- HaruPage::getWordSpace — 現在の単語間隔の値を取得する
- HaruPage::lineTo — 現在位置から指定した位置まで直線を描画する
- HaruPage::measureText — 指定した幅の中に配置できる文字の数を計算する
- HaruPage::moveTextPos — テキストの位置を、指定したオフセットに移動する
- HaruPage::moveTo — 新しい描画パスの開始位置を設定する
- HaruPage::moveToNextLine — テキストの位置を次の行の行頭に移動する
- HaruPage::rectangle — 現在のパスに矩形を追加する
- HaruPage::setCharSpace — ページの文字間隔を設定する
- HaruPage::setCMYKFill — ページの塗りつぶし色を設定する
- HaruPage::setCMYKStroke — ページの描画色を設定する
- HaruPage::setDash — ページの破線パターンを設定する
- HaruPage::setFlatness — ページの平坦度を設定する
- HaruPage::setFontAndSize — ページのフォントおよびフォントサイズを設定する
- HaruPage::setGrayFill — ページの塗りつぶし色を設定する
- HaruPage::setGrayStroke — ページの描画色を設定する
- HaruPage::setHeight — ページの高さを設定する
- HaruPage::setHorizontalScaling — ページの横方向の拡大率を設定する
- HaruPage::setLineCap — パスの終端の形式を設定する
- HaruPage::setLineJoin — そのページのパスの角の形式を設定する
- HaruPage::setLineWidth — ページの線幅を設定する

- HaruPage::setMiterLimit — ページのマイターリミットの現在値を設定する
- HaruPage::setRGBFill — ページの塗りつぶし色を設定する
- HaruPage::setRGBStroke — ページの描画色を設定する
- HaruPage::setRotate — ページの回転角度を設定する
- HaruPage::setSize — ページのサイズおよび方向を設定する
- HaruPage::setSlideShow — ページの移動方式を設定する
- HaruPage::setTextLeading — ページのテキストリーディング (行間隔) を設定する
- HaruPage::setTextMatrix — そのページの現在のテキスト変換行列を設定する
- HaruPage::setTextRenderingMode — そのページの現在のテキストのレンダリングモードを設定する
- HaruPage::setTextRise — 現在のテキストライズの値を設定する
- HaruPage::setWidth — ページの幅を設定する
- HaruPage::setWordSpace — ページの単語間隔を設定する
- HaruPage::showText — ページの現在位置にテキストを表示する
- HaruPage::showTextNextLine — 現在位置を次の行の行頭に移動してテキストを表示する
- HaruPage::stroke — 現在のパスを描画する
- HaruPage::textOut — テキストを指定した位置に表示する
- HaruPage::textRect — 指定した領域内にテキストを表示する

ハッシュ関数

導入

メッセージダイジェスト (ハッシュ) エンジンです。さまざまなハッシュ アルゴリズムを使用して、任意の長さのメッセージに対する 直接的あるいは段階的な処理を可能とします。

要件

ハッシュ拡張モジュールは外部のライブラリを必要とせず、PHP 5.1.2 ではデフォルトで有効となります。configure 時に --disable-hash を指定することで明示的に無効にすることも可能です。それ以前の バージョンの PHP にハッシュ拡張モジュールを組み込むには、 » PECL モジュール をインストールします。

リソース型

この拡張モジュールでは、ハッシュコンテキストリソースを定義しています。これは hash_init() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

HASH_HMAC (integer)
hash_init() のオプションのフラグです。 現在のハッシュコンテキストに対して HMAC digest-keying アルゴリズムが適用されることを示します。

---

hash_algos

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_algos — 登録されているハッシュアルゴリズムの一覧を返す

説明

array **hash_algos** (void)

返り値

サポートされているハッシュアルゴリズムの一覧を、数値添字の 配列として返します。

例

Example#1 hash_algos() の例

PHP 5.1.2 では、**hash_algos()** は 以下のようなアルゴリズム名の一覧を返します。

```
<?php
print_r(hash_algos());
?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => md4
     [1] => md5
     [2] => sha1
     [3] => sha256
     [4] => sha384
     [5] => sha512
     [6] => ripemd128
     [7] => ripemd160
     [8] => whirlpool
     [9] => tiger128,3
     [10] => tiger160,3
     [11] => tiger192,3
     [12] => tiger128,4
     [13] => tiger160,4
     [14] => tiger192,4
     [15] => snefru
     [16] => gost
     [17] => adler32
     [18] => crc32
     [19] => crc32b
     [20] => haval128,3
     [21] => haval160,3
     [22] => haval192,3
     [23] => haval224,3
     [24] => haval256,3
     [25] => haval128,4
     [26] => haval160,4
     [27] => haval192,4
     [28] => haval224,4
     [29] => haval256,4
     [30] => haval128,5
     [31] => haval160,5
     [32] => haval192,5
     [33] => haval224,5
     [34] => haval256,5
)
```

---

hash_file

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_file — ファイルの内容から、ハッシュ値を生成する

説明

string **hash_file** (string `$algo` , string `$filename` [, bool `$raw_output`])

パラメータ

`algo`

選択したアルゴリズムの名前 (すなわち "md5"、"sha256"、"haval160,4" など…)。

`filename`

ハッシュ対象となるファイルの位置を示す URL。 fopen ラッパーをサポートしています。

`raw_output`

TRUE を設定すると、生のバイナリデータを出力します。 デフォルト (**FALSE**) の場合は小文字の 16 進数値となります。

返り値

`raw_output` が true に設定されていない場合は、 メッセージダイジェストの計算結果を小文字の 16 進数値形式の文字列で 返します。もし true に設定されていた場合は、メッセージダイジェストが そのままのバイナリ形式で返されます。

例

Example#1 hash_file() の使用例

```php
<?php
/* ハッシュ値を計算するファイルを作成します */
file_put_contents('example.txt', 'The quick brown fox jumped over the lazy dog.');

echo hash_file('md5', 'example.txt');
?>
```

上の例の出力は以下となります。

```
5c6ffbdd40d9556b73a21e63c3e0e904
```

参考

- hash()
- hash_hmac_file()
- hash_update_file()

---

hash_final

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_final — 段階的なハッシュ処理を終了し、出来上がったダイジェストを返す

説明

string **hash_final** (resource `$context` [, bool `$raw_output`])

パラメータ

`context`

hash_init() が返すハッシュコンテキスト。

`raw_output`

TRUE を設定すると、生のバイナリデータを出力します。 デフォルト (**FALSE**) の場合は小文字の 16 進数値となります。

返り値

raw_output が true に設定されていない場合は、 メッセージダイジェストの計算結果を小文字の 16 進数値形式の文字列で 返します。もし true に設定されていた場合は、メッセージダイジェストが そのままのバイナリ形式で返されます。

例

Example#1 hash_final() の例

```php
<?php
$ctx = hash_init('sha1');
hash_update($ctx, 'The quick brown fox jumped over the lazy dog.');
echo hash_final($ctx);
?>
```

上の例の出力は以下となります。

```
c0854fb9fb03c41cce3802cb0d220529e6eef94e
```

参考

- hash_init()
- hash_update()
- hash_update_stream()
- hash_update_file()

---

hash_hmac_file

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_hmac_file — HMAC 方式を使用して、指定されたファイルの内容からハッシュ値を生成する

説明

string **hash_hmac_file** (string $algo , string $filename , string $key [, bool $raw_output])

パラメータ

algo

選択したアルゴリズムの名前 (すなわち "md5"、"sha256"、"haval160,4" など…)。

filename

ハッシュ対象となるファイルの位置を示す URL。 fopen ラッパーをサポートしています。

key

HMAC 方式でのメッセージダイジェストを生成するために使用する 共有の秘密鍵。

raw_output

TRUE を設定すると、生のバイナリデータを出力します。 デフォルト (**FALSE**) の場合は小文字の 16 進数値となります。

返り値

raw_output が true に設定されていない場合は、 メッセージダイジェストの計算結果を小文字の 16 進数値形式の文字列で 返します。もし true に設定されていた場合は、メッセージダイジェストが そのままのバイナリ形式で返されます。

例

Example#1 hash_hmac_file() の例

```
<?php
/* ハッシュ値を計算するファイルを作成します */
file_put_contents('example.txt', 'The quick brown fox jumped over the lazy dog.');

echo hash_hmac_file('md5', 'example.txt', 'secret');
?>
```

上の例の出力は以下となります。

```
7eb2b5c37443418fc77c136dd20e859c
```

参考

- hash_hmac()
- hash_file()

---

hash_hmac

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_hmac — HMAC 方式を使用してハッシュ値を生成する

説明

string **hash_hmac** (string $algo , string $data , string $key [, bool $raw_output])

パラメータ

algo

選択したアルゴリズムの名前 (すなわち "md5"、"sha256"、"haval160,4" など…)。

data

ハッシュするメッセージ。

key

HMAC 方式でのメッセージダイジェストを生成するために使用する 共有の秘密鍵。

raw_output

TRUE を設定すると、生のバイナリデータを出力します。 デフォルト (**FALSE**) の場合は小文字の 16 進数値となります。

返り値

raw_output が true に設定されていない場合は、 メッセージダイジェストの計算結果を小文字の 16 進数値形式の文字列で 返します。もし true に設定されていた場合は、メッセージダイジェストが そのままのバイナリ形式で返されます。

例

Example#1 hash_hmac() の例

```
<?php
echo hash_hmac('ripemd160', 'The quick brown fox jumped over the lazy dog.', 'secret');
?>
```

上の例の出力は以下となります。

```
b8e7ae12510bdfb1812e463a7f086122cf37e4f7
```

参考

- hash()
- hash_init()
- hash_hmac_file()

---

hash_init

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_init — 段階的なハッシュコンテキストを初期化する

説明

resource **hash_init** (string `$algo` [, int `$options`], string `$key`)

パラメータ

algo

選択したアルゴリズムの名前 (すなわち "md5"、"sha256"、"haval160,4" など…)。

options

ハッシュ生成の際に使用するオプションで、現在は、ただひとつ **HASH_HMAC** のみをサポートしています。これが 指定された場合、*key* を 必ず指定しなければなりません。

key

options に **HASH_HMAC** が指定された場合に、HMAC 形式のハッシュで使用される共有の秘密鍵を 設定する必要があります。

返り値

hash_update()、 hash_update_stream()、hash_update_file() および hash_final() で使用するハッシュコンテキストリソースを 返します。

例

Example#1 段階的なハッシュの例

```
<?php
$ctx = hash_init('md5');
hash_update($ctx, 'The quick brown fox ');
hash_update($ctx, 'jumped over the lazy dog.');
echo hash_final($ctx);
?>
```

上の例の出力は以下となります。

```
5c6ffbdd40d9556b73a21e63c3e0e904
```

参考

- hash()
- hash_file()
- hash_hmac()
- hash_hmac_file()

---

hash_update_file

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_update_file — アクティブなハッシュコンテキストに、ファイルから データを投入する

説明

bool **hash_update_file** (resource `$context` , string `$filename` [, resource `$context`])

パラメータ

`context`

hash_init() が返すハッシュコンテキスト。

`filename`

ハッシュ対象となるファイルの位置を示す URL。 fopen ラッパーをサポートしています。

`context`

stream_context_create() が返す ストリームコンテキスト。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- hash_init()
- hash_update()
- hash_update_stream()
- hash_final()
- hash()
- hash_file()

---

hash_update_stream

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_update_stream — アクティブなハッシュコンテキストに、オープンしているストリームから データを投入する

説明

int **hash_update_stream** (resource `$context` , resource `$handle` [, int `$length`])

パラメータ

`context`

hash_init() が返すハッシュコンテキスト。

`handle`

ストリーム作成用の関数が返す、オープンしているファイルハンドル。

length

handle からハッシュコンテキストにコピーする 最大文字数。

返り値

handle からハッシュコンテキストに追加された 実際のバイト数を返します。

例

Example#1 hash_update_stream() の例

```
<?php
$fp = tmpfile();
fwrite($fp, 'The quick brown fox jumped over the lazy dog.');
rewind($fp);

$ctx = hash_init('md5');
hash_update_stream($ctx, $fp);
echo hash_final($ctx);
?>
```

上の例の出力は以下となります。

```
5c6ffbdd40d9556b73a21e63c3e0e904
```

参考

- hash_init()
- hash_update()
- hash_final()
- hash()
- hash_file()

---

hash_update

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash_update — アクティブなハッシュコンテキストにデータを投入する

説明

bool **hash_update** (resource `$context` , string `$data`)

パラメータ

context

hash_init() が返すハッシュコンテキスト。

data

ハッシュダイジェストに含めるメッセージ。

返り値

TRUE を返します。

参考

- hash_init()
- hash_update_file()
- hash_update_stream()
- hash_final()

---

hash

(PHP 5 >= 5.1.2, PECL hash:1.1-1.5)

hash — ハッシュ値 (メッセージダイジェスト) を生成する

説明

string **hash** (string $algo , string $data [, bool $raw_output])

パラメータ

algo

選択したアルゴリズムの名前 (すなわち "md5"、"sha256"、"haval160,4" など…)。

data

ハッシュするメッセージ。

raw_output

TRUE を設定すると、生のバイナリデータを出力します。 デフォルト (**FALSE**) の場合は小文字の 16 進数値となります。

返り値

raw_output が true に設定されていない場合は、 メッセージダイジェストの計算結果を小文字の 16 進数値形式の文字列で 返します。もし true に設定されていた場合は、メッセージダイジェストが そのままのバイナリ形式で返されます。

例

Example#1 hash() の例

```
<?php
echo hash('ripemd160', 'The quick brown fox jumped over the lazy dog.');
?>
```

上の例の出力は以下となります。

```
ec457d0a974c48d5685a7efa03d137dc8bbde7e3
```

参考

- hash_file()
- hash_hmac()
- hash_init()

目次

HTTP

導入

この HTTP 拡張モジュールの狙いは、PHP アプリケーションのために便利で強力な機能を提供することです。

HTTP の URL、日付、リダイレクト、ヘッダおよびメッセージを 使いやすくし、クライアントの希望する言語および文字セットの ネゴシエーション手

段を提供します。 また、任意のデータを送信する際にキャッシュやリジュームの機能を もたせることができます。

CURL サポート込みでビルドされている場合は、 強力なリクエスト機能を提供します。PHP 5 以降では、 複数のリクエストを平行して実行することができます。

このマニュアルでは、API リファレンスに加えて インストールや設定の方法、定義済みのグローバル定数などを 以下の節で説明しています。

- インストール
- 設定
- グローバル定数
- リソース型

クラス

以下のクラスが定義されています。この拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

HttpResponse クラスは PHP v5.1 以降が必要です。 その他のクラスは PHP v5.0 以降で使用できます。

> **注意**: しかし、PHP v5.0 ではいくつか使用できないメソッドがあることに注意しましょう。

- HttpMessage
- HttpQueryString
- HttpDeflateStream
- HttpInflateStream
- HttpRequest
- HttpRequestPool
- HttpResponse

関数

組み込みの関数

このページですでに取り上げられた組み込みの HTTP 関連関数については ネットワーク関数 に説明があります。

以下の関数は、HTTP モジュールを必要としません。 header()、headers_list()、headers_sent()、 setcookie() そして setrawcookie()。

キャッシュ

- http_cache_etag()
- http_cache_last_modified()

エンコーディング

- http_chunked_decode()
- http_deflate()
- http_inflate()

雑多な関数

- http_build_cookie()
- http_date()
- http_get_request_body_stream()
- http_get_request_body()
- http_get_request_headers()
- http_match_etag()
- http_match_modified()
- http_match_request_header()
- http_support()

ネゴシエーション

- http_negotiate_charset()
- http_negotiate_content_type()
- http_negotiate_language()

出力ハンドラ

- ob_deflatehandler()
- ob_etaghandler()
- ob_inflatehandler()

パーサ

- http_parse_cookie()
- http_parse_headers()
- http_parse_message()
- http_parse_params()

リクエスト

- http_get()
- http_head()
- http_post_data()
- http_post_fields()
- http_put_data()
- http_put_file()
- http_put_stream()
- http_request_body_encode()
- http_request_method_exists()
- http_request_method_name()
- http_request_method_register()
- http_request_method_unregister()
- http_request()

レスポンス

- http_redirect()
- http_send_content_disposition()
- http_send_content_type()
- http_send_data()
- http_send_file()
- http_send_last_modified()
- http_send_status()
- http_send_stream()
- http_throttle()

URL

- http_build_str()
- http_build_url()

持続的なハンドル

- http_persistent_handles_count()
- http_persistent_handles_ident()
- http_persistent_handles_clean()

インストール

(No version information available, might be only in CVS)

インストール — HTTP 拡張モジュールのインストール

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/pecl_http.

> **注意**: この拡張モジュールの正式な名前は *pecl_http* です。

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

Windows 上でのインストールのための要件

Windows でこの拡張モジュールを使用するには、さらに hash, iconv そして SPL をロードする必要があります。

その他のプラットフォーム上でのインストールのための要件

この拡張モジュールを » libcurl サポートつきでビルドしないと、 *request* 機能 (*--with-http-curl-requests*) を使用することができません。ライブラリのバージョンは *v7.12.3* 以降でなければなりません。

*圧縮された*レスポンスデータの送受信を有効にするには、 この拡張モジュールを » zlib サポートつきでビルドする必要があります (*--with-http-zlib-compression*)。 ライブラリのバージョンは *v1.2.2* 以降でなければなりません。

*Content type の推測*を使用するには、この拡張モジュールを » libmagic サポートつきでビルドする必要があります (*--with-http-magic-mime*)。

設定

(No version information available, might be only in CVS)

設定 — http モジュールの設定ディレクティブ

実行時設定

php.ini の設定により動作が変化します。

HTTP 設定オプション

名前	デフォルト	変更の可否	説明
http.etag.mode	"MD5"	PHP_INI_ALL	ETag の生成に使用するハッシュアルゴリズム。MD5、SHA1 および CRC32 が常に使用可能です。hash 拡張モジュールが有効な場合は、この拡張モジュールがサポートする 任意のハッシュアルゴリズムを使用可能です。
http.log.cache	""	PHP_INI_ALL	ログファイルへのパス (あるいはストリームラッパーの url)。 キャッシュにヒットした場合にここに書き込みます。
http.log.redirect	""	PHP_INI_ALL	ログファイルへのパス (あるいはストリームラッパーの url)。 リダイレクトした場合にここに書き込みます。
http.log.not_found	""	PHP_INI_ALL	ログファイルへのパス (あるいはストリームラッパーの url)。 "file not found" エラーが発生した場合にここに書き込みます。
http.log.allowed_methods	""	PHP_INI_ALL	ログファイルへのパス (あるいはストリームラッパーの url)。 "allowed methods" に違反した場合にここに書き込みます。
http.log.composite	""	PHP_INI_ALL	ログファイルへのパス (あるいはストリームラッパーの url)。 すべてのイベントをここに書き込みます。
http.request.methods.allowed	""	PHP_INI_ALL	許可するリクエストメソッド。ここに挙げた以外のリクエストメソッドを クライアントが発行した場合は、ステータス "405 Method not allowed" で PHP が終了します。 "終了" の意味については、INI 設定 http.force_exit を参照ください。
http.request.methods.custom	""	PHP_INI_PERDIR¦PHP_INI_SYSTEM	独自のリクエストメソッド。非標準のリクエストメソッドを使用したい場合は、 これを INI 設定あるいは http_request_method_register() で指定します。
http.request.datashare.cookie	"0"	PHP_INI_SYSTEM	グローバルの HttpRequestDataShare がデフォルトでクッキー情報を共有するかどうか。
http.request.datashare.dns	"1"	PHP_INI_SYSTEM	グローバルの HttpRequestDataShare がデフォルトで名前解決情報を共有するかどうか。
http.request.datashare.ssl	"0"	PHP_INI_SYSTEM	グローバルの HttpRequestDataShare がデフォルトで SSL セッション情報を共有するかどうか。 これは、まだ libcurl で実装されていません。
http.request.datashare.connect	"0"	PHP_INI_SYSTEM	グローバルの HttpRequestDataShare がデフォルトで接続情報を共有するかどうか。 これは、まだ libcurl で実装されていません。
http.persistent.handles.limit	"-1"	PHP_INI_SYSTEM	持続させるハンドルの最大数。
http.persistent.handles.ident	"GLOBAL"	PHP_INI_ALL	持続ハンドルの ident。
http.send.inflate.start_auto	"0"	PHP_INI_PERDIR¦PHP_INI_SYSTEM	inflate 出力ハンドラを自動的に開始するかどうか。
http.send.inflate.start_flags	"0"	PHP_INI_ALL	inflate 出力ハンドラの初期化設定。
http.send.deflate.start_auto	"0"	PHP_INI_PERDIR¦PHP_INI_SYSTEM	deflate 出力ハンドラを自動的に開始するかどうか。
http.send.deflate.start_flags	"0"	PHP_INI_ALL	deflate 出力ハンドラの初期化設定。 deflate 定数 を参照ください。
http.send.not_found_404	"1"	PHP_INI_ALL	http_send_file() が指定したファイルを見つけられなかった場合に 自動的に "404 Not found" で終了するかどうか。 "終了" の意味については、INI 設定 http.force_exit を参照ください。
http.only_exceptions	"0"	PHP_INI_ALL	すべての notices/warnings/errors を例外としてスローするかどうか。
http.force_exit	"1"	PHP_INI_ALL	"exits with a status of..." が発生するたびに、 通常はスクリプトの実行を終了します。このオプションを無効にすると、廃棄用 (dev/null) の出力ハンドラを開始し、 スクリプトをそのまま続行します。

リソース

(No version information available, might be only in CVS)

リソース — HTTP 拡張モジュールが作成するリソース

リソース型

この拡張モジュールではストリームリソースを定義しています。これは http_get_request_body_stream() および (その後の) HttpResponse::getStream() が返すものです。

定数

(No version information available, might be only in CVS)

定数 — http モジュールの定義済みの定数

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

http_support() で使用する定数

HTTP_SUPPORT (integer)
: この定数を問い合わせると、常に **TRUE** を返します。

HTTP_SUPPORT_REQUESTS (integer)
: HTTP リクエストをサポートしているかどうか、 つまり libcurl のサポート込みでコンパイルされているかどうか。

HTTP_SUPPORT_MAGICMIME (integer)
: HTTP メッセージにおける Content-Type の推測をサポートしているかどうか、 つまり libmagic のサポート込みでコンパイルされているかどうか。

HTTP_SUPPORT_ENCODINGS (integer)
: zlib エンコーディングをサポートしているかどうか、 つまり libz のサポート込みでコンパイルされているかどうか。

HTTP_SUPPORT_SSLREQUESTS (integer)
: SSL 越しの HTTP リクエストの発行をサポートしているかどうか、 つまり、リンクしている libcurl に SSL サポートが組み込まれているかどうか。

http_parse_params() で使用する定数

HTTP_PARAMS_ALLOW_COMMA (integer)
: 区切り文字として、セミコロンのほかにカンマも許可します。

HTTP_PARAMS_ALLOW_FAILURE (integer)
: エラーが発生した後もパースを続行します。

HTTP_PARAMS_RAISE_ERROR (integer)
: パースエラー時に PHP の警告を発生します。

HTTP_PARAMS_DEFAULT (integer)
: 上の三つすべての論理和。

http_parse_cookie() およびその返り値で使用する定数

HTTP_COOKIE_PARSE_RAW (integer)
: 値を urldecode しません。

HTTP_COOKIE_SECURE (integer)
: クッキーのパラメータリストに "secure" があるかどうか。

HTTP_COOKIE_HTTPONLY (integer)
: クッキーのパラメータリストに "httpOnly" があるかどうか。

http_deflate() および HttpDeflateStream で使用する定数

HTTP_DEFLATE_LEVEL_DEF (integer)
HTTP_DEFLATE_LEVEL_MIN (integer)
HTTP_DEFLATE_LEVEL_MAX (integer)
HTTP_DEFLATE_TYPE_ZLIB (integer)
HTTP_DEFLATE_TYPE_GZIP (integer)
HTTP_DEFLATE_TYPE_RAW (integer)
HTTP_DEFLATE_STRATEGY_DEF (integer)
HTTP_DEFLATE_STRATEGY_FILT (integer)
HTTP_DEFLATE_STRATEGY_HUFF (integer)

HTTP_DEFLATE_STRATEGY_RLE (integer)
HTTP_DEFLATE_STRATEGY_FIXED (integer)

HttpDeflateStream および HttpInflateStream で使用する定数
HTTP_ENCODING_STREAM_FLUSH_NONE (integer)
フラッシュしません。
HTTP_ENCODING_STREAM_FLUSH_SYNC (integer)
同期フラッシュのみを行います。
HTTP_ENCODING_STREAM_FLUSH_FULL (integer)
データの完全なフラッシュを行います。

エラー報告および例外で使用する定数
HTTP_E_RUNTIME (integer)
実行時エラー。
HTTP_E_INVALID_PARAM (integer)
無効なパラメータが渡されました。
HTTP_E_HEADER (integer)
header() あるいは同等の処理に失敗しました。
HTTP_E_MALFORMED_HEADERS (integer)
HTTP ヘッダのパースエラー。
HTTP_E_REQUEST_METHOD (integer)
未知の/無効な リクエストメソッドです。
HTTP_E_MESSAGE_TYPE (integer)
操作とメッセージの型が一致しません。
HTTP_E_ENCODING (integer)
エンコード/デコード エラー。
HTTP_E_REQUEST (integer)
リクエストに失敗しました。
HTTP_E_REQUEST_POOL (integer)
リクエストプールに失敗しました。
HTTP_E_SOCKET (integer)
ソケットの例外。
HTTP_E_RESPONSE (integer)
レスポンスに失敗しました。
HTTP_E_URL (integer)
URL が無効です。
HTTP_E_QUERYSTRING (integer)
クエリ文字列の操作に失敗しました。

HttpMessage で使用する定数
HTTP_MSG_NONE (integer)
メッセージに型が指定されていません。
HTTP_MSG_REQUEST (integer)
リクエスト型のメッセージ。
HTTP_MSG_RESPONSE (integer)
レスポンス型のメッセージ。

HttpQueryString で使用する定数
HTTP_QUERYSTRING_TYPE_BOOL (integer)
HTTP_QUERYSTRING_TYPE_INT (integer)
HTTP_QUERYSTRING_TYPE_FLOAT (integer)
HTTP_QUERYSTRING_TYPE_STRING (integer)
HTTP_QUERYSTRING_TYPE_ARRAY (integer)
HTTP_QUERYSTRING_TYPE_OBJECT (integer)

httpauthtype リクエストのオプション で使用する定数
HTTP_AUTH_BASIC (integer)
ベーシック認証を使用します。
HTTP_AUTH_DIGEST (integer)
ダイジェスト認証を使用します。
HTTP_AUTH_NTLM (integer)
"NTLM" 認証を使用します。
HTTP_AUTH_GSSNEG (integer)
"GSS-NEGOTIATE" 認証を使用します。
HTTP_AUTH_ANY (integer)
任意の認証スキームを試みます。

HTTP *protocol* バージョン リクエストのオプション で使用する定数

HTTP_VERSION_ANY (integer)
HTTP プロトコルのバージョンを指定しません。

HTTP_VERSION_1_0 (integer)
HTTP バージョン 1.0。

HTTP_VERSION_1_1 (integer)
HTTP バージョン 1.1。

SSL *protocol* 型およびバージョン リクエストのオプション で使用する定数

HTTP_SSL_VERSION_ANY (integer)
SSL プロトコルのバージョンを指定しません。

HTTP_SSL_VERSION_TLSv1 (integer)
TLSv1 のみを使用します。

HTTP_SSL_VERSION_SSLv3 (integer)
SSLv3 のみを使用します。

HTTP_SSL_VERSION_SSLv2 (integer)
SSLv2 のみを使用します。

***proxytype* リクエストのオプション で使用する定数**

HTTP_PROXY_SOCKS4 (integer)
SOCKS4 型のプロキシ。

HTTP_PROXY_SOCKS5 (integer)
SOCKS5 型のプロキシ。

HTTP_PROXY_HTTP (integer)
標準の HTTP プロキシ。

***ipresolve* リクエストのオプション で使用する定数**

HTTP_IPRESOLVE_V4 (integer)
IPv4 のみで名前解決を行います。

HTTP_IPRESOLVE_V6 (integer)
IPv6 のみで名前解決を行います。

HTTP_IPRESOLVE_ANY (integer)
任意の IP メカニズムで名前解決を行います。

定義済みの HTTP リクエストメソッド定数

HTTP_METH_GET (integer)
HTTP_METH_HEAD (integer)
HTTP_METH_POST (integer)
HTTP_METH_PUT (integer)
HTTP_METH_DELETE (integer)
HTTP_METH_OPTIONS (integer)
HTTP_METH_TRACE (integer)
HTTP_METH_CONNECT (integer)
HTTP_METH_PROPFIND (integer)
HTTP_METH_PROPPATCH (integer)
HTTP_METH_MKCOL (integer)
HTTP_METH_COPY (integer)
HTTP_METH_MOVE (integer)
HTTP_METH_LOCK (integer)
HTTP_METH_UNLOCK (integer)
HTTP_METH_VERSION_CONTROL (integer)
HTTP_METH_REPORT (integer)
HTTP_METH_CHECKOUT (integer)
HTTP_METH_CHECKIN (integer)
HTTP_METH_UNCHECKOUT (integer)
HTTP_METH_MKWORKSPACE (integer)
HTTP_METH_UPDATE (integer)
HTTP_METH_LABEL (integer)
HTTP_METH_MERGE (integer)
HTTP_METH_BASELINE_CONTROL (integer)
HTTP_METH_MKACTIVITY (integer)
HTTP_METH_ACL (integer)

http_redirect() で使用する定数

HTTP_REDIRECT (integer)
適切なリダイレクト方式を判断します。

HTTP_REDIRECT_PERM (integer)
永続的なリダイレクト (*301 Moved permanently*)。
HTTP_REDIRECT_FOUND (integer)
標準のリダイレクト (*302 Found*)。

> **注意**: RFC 1945 および RFC 2068 にて、 リダイレクトされたリクエストのメソッドをクライアントで変更してはいけないと規定されています。 しかし、既存のほとんどのユーザエージェントの実装は、 302 をまるで 303 のレスポンスであるかのように扱い、 元のリクエストメソッドが何であるかにかかわらず Location フィールドの値に GET リクエストを実行します。 ステータスコード 303 および 307 が追加されたのは、 クライアント側に期待する反応をサーバ側で指定できるようにするためです。

HTTP_REDIRECT_POST (integer)
POST リクエストが適切なリダイレクト (*303 See other*)。
HTTP_REDIRECT_PROXY (integer)
プロキシリダイレクト (*305 Use proxy*)。
HTTP_REDIRECT_TEMP (integer)
一時的なリダイレクト (*307 Temporary Redirect*)。

http_build_url() で使用する定数
HTTP_URL_REPLACE (integer)
最初の URL のすべての部分を、二番目の URL がある場合にそれで置き換えます。
HTTP_URL_JOIN_PATH (integer)
相対パスを連結します。
HTTP_URL_JOIN_QUERY (integer)
クエリ文字列を連結します。
HTTP_URL_STRIP_USER (integer)
認証ユーザに関する情報を取り除きます。
HTTP_URL_STRIP_PASS (integer)
認証パスワードに関する情報を取り除きます。
HTTP_URL_STRIP_AUTH (integer)
すべての認証情報を取り除きます。
HTTP_URL_STRIP_PORT (integer)
明示的なポート番号の指定を取り除きます。
HTTP_URL_STRIP_PATH (integer)
完全パスを取り除きます。
HTTP_URL_STRIP_QUERY (integer)
クエリ文字列を取り除きます。
HTTP_URL_STRIP_FRAGMENT (integer)
フラグメント (# 以降) を取り除きます。
HTTP_URL_STRIP_ALL (integer)
スキームとホスト以外のすべての部分を取り除きます。

リクエストのオプション

(No version information available, might be only in CVS)

リクエストのオプション — HttpRequest クラスおよびリクエスト関数で使用するオプション

概要

タイムアウトに関連するオプション
timeout (integer)
リクエスト全体が完了するまでの秒数。
connecttimeout (integer)
名前解決を含む接続にかける秒数。
dns_cache_timeout (integer)
dns キャッシュエントリのタイムアウト秒数。

url に関連するオプション
url (string)
リクエスト url。
port (integer)
url で指定したものとは別のポートを使用する場合のポート。
redirect (integer)
リダイレクトをどれだけ追いかけるか。デフォルトは 0。

unrestrictedauth (bool)
　　別のホストにリダイレクトする際に認証情報を引き継ぐかどうか。
referer (string)
　　リファラとして送信する url。

クッキーに関連するオプション
encodecookies (bool)
　　独自のクッキーを送信する前に urlencode() するかどうか。
cookies (array)
　　クッキーの内容を表す *array("cookie" => "value")* 形式の連想配列。
cookiestore (string)
　　クッキーを保存するファイルへのパス。
cookiesession (bool)
　　TRUE の場合は、cookiestore からセッションクッキーを読み込まない。

ヘッダに関連するオプション
useragent (string)
　　ユーザエージェントとして送信する内容。デフォルトは *PECL::HTTP/x.y.z (PHP/x.y.z)*。 送信したくない場合は、明示的に空文字列を設定します。
lastmodified (int)
　　If-(Un)Modified-Since ヘッダ用のタイムスタンプ。
etag (string)
　　If-(None-)Match ヘッダ用のクォートした etag。
headers (array)
　　独自のヘッダを指定する、 *array("header" => "value")* 形式の連想配列。

認証に関連するオプション
httpauth (string)
　　"user:pass" 形式の http 認証情報。
httpauthtype (int)
　　HTTP 認証方式の定数。
(array)

プロキシに関連するオプション
proxyhost (string)
　　"host[:port]" 形式のプロキシホスト。
proxyport (int)
　　proxyhost で指定したものとは別のポートを使用する場合のプロキシポート。
proxytype (int)
　　HTTP プロキシ形式の定数。
proxyauth (string)
　　"user:pass" 形式のプロキシ認証情報。
proxyauthtype (int)
　　HTTP 認証形式の定数。

転送に関連するオプション
compress (bool)
　　gzip/deflate エンコードされたレスポンスを受け入れるかどうか。
resume (int)
　　サーバがサポートしている場合 (レスポンスコード 206) に、指定したバイトオフセットからダウンロードを開始する。
range (array)
　　配列の配列で、それぞれ二つの integer を含み。 ダウンロードする範囲を指定する。 サーバがサポートしており (レスポンスコード 206)、resume オプションが空の場合にのみ有効。

制限に関連するオプション
maxfilesize (integer)
　　ダウンロードできるファイルサイズの最大値。 リクエストされたエンティティのサイズが取得できない場合 (動的なページで分割して転送している場合など) には無意味。
low_speed_limit (int)
　　リクエストが成功するために必要な最低限の転送速度。
low_speed_time (int)
　　リクエストが成功するために、 *low_speed_limit* 以上で転送しなければならない時間。
max_send_speed (int)
　　最大の送信速度。バイト毎秒。
max_recv_speed (int)
　　最大の受信速度。バイト毎秒。

コールバックオプション
onprogress (callback)
　　進捗状況のコールバック。

ネットワークオプション
interface (string)
　　送信用のネットワークインターフェイス (ifname、ip あるいは hostname)。
portrange (array)
　　二つの整数で指定する、送信用のポートの範囲。

SSL オプション
ssl (array)

　　注意: SSL のオプションは、オプション名 "ssl" の配列として指定します。

　　cert (string)
　　　　証明書へのパス。
　　certtype (string)
　　　　証明書の形式。
　　certpasswd (string)
　　　　証明書のパスワード。
　　key (string)
　　　　鍵へのパス。
　　keytype (string)
　　　　鍵の形式。
　　keypasswd (string)
　　　　鍵のパスワード。
　　engine (string)
　　　　使用する ssl エンジン。
　　version (int)
　　　　使用する ssl バージョン。
　　verifypeer (bool)
　　　　相手側を検証するかどうか。
　　verifyhost (bool)
　　　　ホストを検証するかどうか。
　　cipher_list (string)
　　　　利用できる暗号形式の一覧。
　　cainfo (string)
　　capath (string)
　　random_file (string)
　　egdsocket (string)

HttpMessage

(No version information available, might be only in CVS)

HttpMessage — HTTP メッセージクラス

```
class HttpMessage implements Iterator, Countable, Serializable
```

クラスのメンバ

プロパティ

インスタンスプロパティ

アクセス範囲	型	名前	説明
protected	int	type	メッセージの型
protected	string	body	メッセージの本文
protected	float	httpVersion	HTTP プロトコルのバージョン
protected	array	headers	メッセージのヘッダ
protected	string	requestMethod	リクエストメソッドの名前
protected	requestUrl	string	リクエスト URL
protected	int	responseCode	レスポンスコード
protected	string	responseStatus	レスポンスステータスのメッセージ
protected	HttpMessage	parentMessage	親メッセージへの参照

注意: これらのデフォルトプロパティについては、参照としてアクセスすることはできません。 また、配列のキー/値 方式の表記を使用することもできませんし、 インクリメント/デクリメント 操作を行うこともできません。

定義済み定数

型	名前	説明
int	TYPE_NONE	メッセージは特定の型ではありません
int	TYPE_REQUEST	メッセージは、リクエスト型の HTTP メッセージです
int	TYPE_RESPONSE	メッセージは、レスポンス型の HTTP メッセージです

メソッド

- HttpMessage::construct()
- HttpMessage::factory()
- HttpMessage::fromEnv()
- HttpMessage::fromString()
- HttpMessage::toString()
- HttpMessage::toMessageTypeObject()
- HttpMessage::guessContentType()
- HttpMessage::detach()
- HttpMessage::prepend()
- HttpMessage::reverse()
- HttpMessage::send()
- HttpMessage::getParentMessage()
- HttpMessage::getType()
- HttpMessage::setType()
- HttpMessage::getHttpVersion()
- HttpMessage::setHttpVersion()
- HttpMessage::getHeaders()
- HttpMessage::getHeader()
- HttpMessage::addHeaders()
- HttpMessage::setHeaders()
- HttpMessage::getBody()
- HttpMessage::setBody()
- HttpMessage::getRequestMethod()
- HttpMessage::setRequestMethod()
- HttpMessage::getRequestUrl()
- HttpMessage::setRequestUrl()
- HttpMessage::getResponseCode()
- HttpMessage::setResponseCode()
- HttpMessage::getResponseStatus()
- HttpMessage::setResponseStatus()

HttpMessage::__construct

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::__construct — HttpMessage のコンストラクタ

説明

public void **HttpMessage::__construct** ([string `$message`])

新しい HttpMessage オブジェクトのインスタンスを作成します。

作成されたオブジェクトが表すのは、渡された文字列の *直近*のメッセージです。 それ以前のメッセージが存在する場合、そこにアクセスするには HttpMessage::**getParentMessage()** を使用します。

パラメータ

`message`

単一の、あるいは複数の連続した HTTP メッセージ。

エラー / 例外

HttpMalformedHeaderException をスローします。

HttpMessage::factory

(PECL pecl_http:1.4.0-1.5.5)

HttpMessage::factory — 文字列から HttpMessage を作成する

説明

static public HttpMessage **HttpMessage::factory** ([string `$raw_message` [, string `$class_name = 'HttpMessage'`]])

文字列から HttpMessage オブジェクトを作成します。

パラメータ

`raw_message`

単一の、あるいは連続した複数の HTTP メッセージ。

`class_name`

HttpMessage を継承したクラス。

返り値

成功した場合は HttpMessage オブジェクト、失敗した場合は NULL を返します。

エラー / 例外

HttpMalformedHeadersException をスローします。

参考

- **HttpMessage::fromEnv()**.

HttpMessage::fromEnv

(PECL pecl_http:1.5.0-1.5.5)

HttpMessage::fromEnv — 環境から HttpMessage を作成する

説明

static public HttpMessage **HttpMessage::fromEnv** (int `$message_type` [, string `$class_name = 'HttpMessage'`])

スクリプトの環境から HttpMessage オブジェクトを作成します。

パラメータ

message_type

メッセージの型。HttpMessage の型の定数 を参照ください。

class_name

HttpMessage を継承したクラス。

返り値

成功した場合は HttpMessage オブジェクト、失敗した場合は NULL を返します。

参考

- HttpMessage::factory().

HttpMessage::fromString

(PECL pecl_http:0.10.0-1.3.3)

HttpMessage::fromString — 文字列から HttpMessage を作成する

説明

static public HttpMessage **HttpMessage::fromString** ([string `$raw_message` [, string `$class_name = 'HttpMessage'`]])

文字列から HttpMessage オブジェクトを作成します。

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

パラメータ

raw_message

単一の、あるいは複数の連続した HTTP メッセージ。

class_name

HttpMessage を継承したクラス。

返り値

成功した場合は HttpMessage オブジェクト、失敗した場合は NULL を返します。

エラー / 例外

HttpMalformedHeadersException をスローします。

参考

- HttpMessage::factory().

HttpMessage::toString

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::toString — 文字列表現を取得する

説明

public string **HttpMessage::toString** ([bool `$include_parent = FALSE`])

メッセージの文字列表現を取得します。

パラメータ

include_parent

返される文字列に親メッセージを含めるかどうかを指定します。

返り値

メッセージを文字列として返します。

HttpMessage::toMessageTypeObject

(PECL pecl_http:0.22.0-1.5.5)

HttpMessage::toMessageTypeObject — メッセージの型に応じた HTTP オブジェクトを作成する

説明

public HttpRequest¦HttpResponse **HttpMessage::toMessageTypeObject** (void)

メッセージの型に応じたオブジェクトを作成します。

パラメータ

返り値

成功した場合に HttpRequest オブジェクトあるいは HttpResponse オブジェクト、失敗した場合に NULL を返します。

エラー / 例外

HttpRuntimeException、HttpMessageTypeException、HttpHeaderException をスローします。

HttpMessage::guessContentType

(PECL pecl_http:1.0.0-1.5.5)

HttpMessage::guessContentType — content type を推測する

説明

public string **HttpMessage::guessContentType** (string `$magic_file` [, int `$magic_mode = MAGIC_MIME`])

メッセージ本文の content type を、libmagic を使用して推測します。

パラメータ

magic_file

使用する magic.mime データベース。

magic_mode

libmagic のフラグ。

返り値

成功した場合は推測した content type、 失敗した場合は FALSE を返します。

エラー / 例外

HttpRuntimeException、HttpInvalidParamException をスローします。

HttpMessage::detach

(PECL pecl_http:0.22.0-1.5.5)

HttpMessage::detach — HttpMessage をデタッチする

説明

public HttpMessage **HttpMessage::detach** (void)

HttpMessage オブジェクトの複製を返します。 これは親メッセージからデタッチされたものです。

パラメータ

返り値

デタッチされた、HttpMessage オブジェクトの複製を返します。

HttpMessage::prepend

(PECL pecl_http:0.22.0-1.5.5)

HttpMessage::prepend — メッセージを先頭に追加する

説明

public void **HttpMessage::prepend** (HttpMessage `$message` [, bool `$top = TRUE`])

HTTP メッセージの先頭に、メッセージを追加します。

パラメータ

`message`

先頭に追加する HttpMessage オブジェクト。

`top`

メッセージ全体の先頭に追加するか、このメッセージの先頭に追加するのかを指定します。

エラー / 例外

message が同一のメッセージチェイン内にある場合に、 HttpInvalidParamException をスローします。

HttpMessage::reverse

(PECL pecl_http:0.23.0-1.5.5)

HttpMessage::reverse — メッセージチェインを逆順にする

説明

public HttpMessage **HttpMessage::reverse** (void)

メッセージチェインを逆順に並べなおします。

返り値

最上位の HttpMessage オブジェクトを返します。

HttpMessage::send

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::send — メッセージを送信する

説明

public bool **HttpMessage::send** (void)

レスポンスあるいはリクエストのいずれかの型に応じたメッセージを送信します。

このメソッドの機能は、HttpRequest および HttpResponse に比べて限定されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpMessage::getParentMessage

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::getParentMessage — 親メッセージを取得する

説明

public HttpMessage **HttpMessage::getParentMessage** (void)

親メッセージを取得します。

返り値

親 HttpMessage オブジェクトを返します。

エラー / 例外

HttpRuntimeException をスローします。

HttpMessage::getType

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::getType — メッセージの型を取得する

説明

public int **HttpMessage::getType** (void)

メッセージの型を取得します。 **HTTP_MSG_NONE**、**HTTP_MSG_REQUEST** あるいは **HTTP_MSG_RESPONSE** のいずれかです。

返り値

HttpMessage::**TYPE** を返します。

HttpMessage::setType

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::setType — メッセージの型を設定する

説明

public void **HttpMessage::setType** (int $type)

メッセージの型を設定します。 **HTTP_MSG_NONE**、**HTTP_MSG_REQUEST** あるいは **HTTP_MSG_RESPONSE** のいずれかとなります。

パラメータ

type

HttpMessage::TYPE

HttpMessage::getHttpVersion

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::getHttpVersion — HTTP バージョンを取得する

説明

public string **HttpMessage::getHttpVersion** (void)

メッセージの HTTP プロトコルバージョンを取得します。

返り値

HTTP プロトコルのバージョンを文字列で返します。

HttpMessage::setHttpVersion

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::setHttpVersion — HTTP バージョンを設定する

説明

public bool **HttpMessage::setHttpVersion** (string `$version`)

メッセージの HTTP プロトコルバージョンを設定します。

パラメータ

`version`

HTTP のプロトコルバージョン。

返り値

成功した場合に TRUE、あるいは指定したバージョンが範囲外 (1.0/1.1 以外) の場合に FALSE を返します。

HttpMessage::getHeaders

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::getHeaders — メッセージのヘッダを取得する

説明

public array **HttpMessage::getHeaders** (void)

メッセージのヘッダを取得します。

返り値

メッセージの HTTP ヘッダを含む連想配列を返します。

HttpMessage::getHeader

(PECL pecl_http:1.1.0-1.5.5)

HttpMessage::getHeader — ヘッダを取得する

説明

public string **HttpMessage::getHeader** (string $header)

メッセージのヘッダを取得します。

パラメータ

header

ヘッダの名前。

返り値

成功した場合はヘッダの値、ヘッダが存在しない場合は NULL を返します。

HttpMessage::addHeaders

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::addHeaders — ヘッダを追加する

説明

public void **HttpMessage::addHeaders** (array $headers [, bool $append = FALSE])

ヘッダを追加します。append が true の場合は同名のヘッダが複数登録されます。 そうでない場合は、同名のヘッダは上書きされます。

パラメータ

headers

既存のメッセージのヘッダに追加する HTTP ヘッダを含む連想配列。

append

true の場合、もし追加しようとしているヘッダが既に存在すれば、 それを配列に変換して両方の値を含むようにします。false の場合は、 既存のヘッダの値が新しい値で上書きされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpMessage::setHeaders

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::setHeaders — ヘッダを設定する

説明

public void **HttpMessage::setHeaders** (array $headers)

新しいヘッダを設定します。

パラメータ

headers

新しい HTTP ヘッダを含む連想配列。メッセージの既存の HTTP ヘッダは、すべて上書きされます。

HttpMessage::getBody

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::getBody — メッセージの本文を取得する

説明

public string **HttpMessage::getBody** (void)

パースされた HttpMessage の本文を取得します。

返り値

メッセージの本文を文字列で返します。

HttpMessage::setBody

(PECL pecl_http:0.14.0-1.5.5)

HttpMessage::setBody — メッセージの本文を設定する

説明

public void **HttpMessage::setBody** (string `$body`)

HttpMessage の本文を設定します。

> **注意**: それに応じてヘッダを適切に更新することを忘れないようにしましょう。

パラメータ

`body`

新しいメッセージ本文。

HttpMessage::getRequestMethod

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::getRequestMethod — リクエストメソッドを取得する

説明

public string **HttpMessage::getRequestMethod** (void)

メッセージのリクエストメソッドを取得します。

返り値

成功した場合はリクエストメソッド、メッセージの型が HttpMessage::**`TYPE_REQUEST`** でない場合は FALSE を返します。

HttpMessage::setRequestMethod

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::setRequestMethod — リクエストメソッドを設定する

説明

public bool **HttpMessage::setRequestMethod** (string `$method`)

HTTP メッセージのリクエストメソッドを設定します。

パラメータ

`method`

リクエストメソッドの名前。

返り値

成功した場合は TRUE、メッセージの型が HttpMessage::**TYPE_REQUEST** でない場合、あるいは無効なリクエストメソッドが指定された場合は FALSE を返します。

HttpMessage::getRequestUrl

(PECL pecl_http:0.21.0-1.5.5)

HttpMessage::getRequestUrl — リクエスト URL を取得する

説明

public string **HttpMessage::getRequestUrl** (void)

メッセージのリクエスト URL を取得します。

パラメータ

返り値

成功した場合はリクエスト URL、メッセージの型が HttpMessage::**TYPE_REQUEST** でない場合は FALSE を返します。

HttpMessage::setRequestUrl

(PECL pecl_http:0.21.0-1.5.5)

HttpMessage::setRequestUrl — リクエスト URL を設定する

説明

public bool **HttpMessage::setRequestUrl** (string `$url`)

HTTP メッセージのリクエスト URL を設定します。

パラメータ

`url`

リクエスト URL。

返り値

成功した場合は TRUE、メッセージの型が HttpMessage::**TYPE_REQUEST** でない場合、あるいは指定した URL が空の場合に FALSE を返します。

HttpMessage::getResponseCode

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::getResponseCode — レスポンスコードを取得する

説明

public int **HttpMessage::getResponseCode** (void)

メッセージのレスポンスコードを取得します。

返り値

メッセージの型が HttpMessage::**TYPE_RESPONSE** の場合は HTTP レスポンスコード、それ以外の場合は FALSE を返します。

HttpMessage::setResponseCode

(PECL pecl_http:0.10.0-1.5.5)

HttpMessage::setResponseCode — レスポンスコードを設定する

説明

public bool **HttpMessage::setResponseCode** (int $code)

HTTP レスポンスメッセージのレスポンスコードを設定します。

パラメータ

code

HTTP レスポンスコード。

返り値

成功した場合は TRUE、メッセージの型が HttpMessage::**TYPE_RESPONSE** でない場合、あるいはレスポンスコードが範囲外 (100-510 以外) の場合に FALSE を返します。

HttpMessage::getResponseStatus

(PECL pecl_http:0.23.0-1.5.5)

HttpMessage::getResponseStatus — レスポンスのステータスを取得する

説明

public string **HttpMessage::getResponseStatus** (void)

メッセージのレスポンスステータス (レスポンスコードの後に続く文字列) を取得します。

返り値

メッセージの型が HttpMessage::**TYPE_RESPONSE** の場合に HTTP レスポンスステータス、それ以外の場合に FALSE を返します。

HttpMessage::setResponseStatus

(PECL pecl_http:0.23.0-1.5.5)

HttpMessage::setResponseStatus — レスポンスのステータスを設定する

説明

public bool **HttpMessage::setResponseStatus** (string $status)

HTTP メッセージのレスポンスステータス (レスポンスコードの後に続く文字列) を設定します。

パラメータ

status

レスポンスステータスを表す文字列。

返り値

成功した場合に TRUE、メッセージの型が HttpMessage::**TYPE_RESPONSE** でない場合に FALSE を返します。

HttpQueryString

(No version information available, might be only in CVS)

HttpQueryString — HTTP クエリ文字列クラス

```
class HttpQueryString implements Serializable, ArrayAccess
```

クラスのメンバ

プロパティ

インスタンスプロパティ

アクセス範囲	型	名前	説明
private	array	queryArray	クエリパラメータ
private	string	queryString	シリアライズしたクエリパラメータ

静的なプロパティ

アクセス範囲	型	名前	説明
private	array	instance	シングルトン

定義済み定数

型	名前	説明
int	TYPE_BOOL	クエリパラメータを bool 型で取得する
int	TYPE_INT	クエリパラメータを int 型で取得する
int	TYPE_FLOAT	クエリパラメータを float 型で取得する
int	TYPE_STRING	クエリパラメータを string 型で取得する
int	TYPE_ARRAY	クエリパラメータを array 型で取得する
int	TYPE_OBJECT	クエリパラメータを object 型で取得する

メソッド

HttpQueryString::__construct

(PECL pecl_http:0.22.0-1.5.5)

HttpQueryString::__construct — HttpQueryString のコンストラクタ

説明

final public void **HttpQueryString::__construct** ([bool `$global = TRUE` [, mixed `$add`]])

新しい HttpQueryString オブジェクトのインスタンスを作成します。

global が TRUE の場合は、 $_GET および $_SERVER['QUERY_STRING'] を操作して変更します。

パラメータ

`global`

$_GET および $_SERVER['QUERY_STRING'] を操作するかどうか

`add`

追加の、あるいは最初のクエリ文字列パラメータ。

エラー / 例外

HttpRuntimeException をスローします。

HttpQueryString::singleton

(PECL pecl_http:0.25.0-1.5.5)

HttpQueryString::singleton — HttpQueryString のシングルトン

説明

static public HttpQueryString **HttpQueryString::singleton** ([bool `$global = TRUE`])

シングルトンインスタンス (グローバル設定によって異なります) を取得します。

パラメータ

`global`

$_GET および $_SERVER['QUERY_STRING'] を操作するかどうか。

返り値

グローバル設定に応じて、常に同一の HttpQueryString インスタンスを返します。

エラー / 例外

HttpRuntimeException をスローします。

HttpQueryString::get

(PECL pecl_http:0.22.0-1.5.5)

HttpQueryString::get — クエリ文字列 (の一部) を取得する

説明

public mixed **HttpQueryString::get** ([string `$key` [, mixed `$type = 0` [, mixed `$defval = NULL` [, bool `$delete = FALSE`]]]])

クエリ文字列 (の一部) を取得します。

パラメータ type には HttpQueryString::TYPE_* 定数のいずれか、 あるいは型を省略したものを指定します。例えば "b" は bool、"i" は int、"f" は float、 "s" は string、"a" は array そして "o" は stdClass オブジェクトです。

パラメータ

`key`

取得するクエリ文字列パラメータのキー。

`type`

変数の型。

`defval`

キーが存在しない場合のデフォルトの値。

`delete`

クエリ文字列からキー・値の組み合わせを削除するかどうか。

返り値

クエリ文字列パラメータの値を返します。キーが指定されない場合は、 クエリ文字列全体を返します。キーが存在しない場合は defval を返します。

HttpQueryString::mod

(PECL pecl_http:1.1.0-1.5.5)

HttpQueryString::mod — クエリ文字列の複製を変更する

説明

public HttpQueryString **HttpQueryString::mod** (mixed $params)

クエリ文字列オブジェクトを複製し、指定したパラメータをその複製に設定します。

パラメータ

params

追加するクエリ文字列パラメータ。

返り値

新しい HttpQueryString オブジェクトを返します。

HttpQueryString::set

(PECL pecl_http:0.22.0-1.5.5)

HttpQueryString::set — クエリ文字列パラメータを設定する

説明

public string **HttpQueryString::set** (mixed $params)

クエリ文字列エントリを設定します。**NULL** を指定すると、変数を削除します。

パラメータ

params

追加するクエリ文字列パラメータ。

返り値

現在のクエリ文字列を返します。

HttpQueryString::toArray

(PECL pecl_http:0.22.0-1.5.5)

HttpQueryString::toArray — クエリ文字列を配列で取得する

説明

public array **HttpQueryString::toArray** (void)

クエリ文字列を、連想配列として取得します。

返り値

クエリ文字列を連想配列として返します。

HttpQueryString::toString

(PECL pecl_http:0.22.0-1.5.5)

HttpQueryString::toString — クエリ文字列を取得する

説明

public string **HttpQueryString::toString** (void)

クエリ文字列を取得します。

パラメータ

返り値

クエリ文字列を文字列として返します。

HttpQueryString::xlate

(PECL pecl_http:0.25.0-1.5.5)

HttpQueryString::xlate — クエリ文字列の文字セットを変更する

説明

public bool **HttpQueryString::xlate** (string `$ie` , string `$oe`)

クエリ文字列のエンコーディングを、ie から oe に変換します。

警告

UTF-16 のように NUL バイトを含められる文字セットは、使用しないでください。

> **注意**: このメソッドを使用するには、ext/iconv が有効になっている必要があります。

パラメータ

ie

入力エンコーディング。

oe

出力エンコーディング。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpDeflateStream

(No version information available, might be only in CVS)

HttpDeflateStream — HTTP 圧縮ストリームクラス

```
class  HttpDeflateStream
```

クラスのメンバ

定義済み定数

型	名前	説明
int	TYPE_GZIP	gzip エンコーディング
int	TYPE_ZLIB	zlib 圧縮エンコーディング

型	名前	説明
int	TYPE_RAW	raw 圧縮エンコーディング
int	LEVEL_DEF	デフォルトの圧縮レベル
int	LEVEL_MIN	最小限の圧縮レベル
int	LEVEL_MAX	最大限の圧縮レベル
int	STRATEGY_DEF	デフォルトの方式
int	STRATEGY_FILT	フィルタ方式
int	STRATEGY_HUFF	ハフマン方式
int	STRATEGY_RLE	RLE 方式
int	STRATEGY_FIXED	fixed 方式
int	FLUSH_NONE	フラッシュを強制しない
int	FLUSH_SYNC	フラッシュを同期させる
int	FLUSH_FULL	完全にフラッシュする

メソッド

- HttpDeflateStream::**factory()**
- HttpDeflateStream::**__construct()**
- HttpDeflateStream::**update()**
- HttpDeflateStream::flush()
- HttpDeflateStream::**finish()**

例

Example#1 HttpDeflateStream の例

```
<?php
$stream = new HttpDeflateStream(
    HttpDeflateStream::TYPE_GZIP |
    HttpDeflateStream::LEVEL_MAX |
    HttpDeflateStream::FLUSH_SYNC);

echo $stream->update($data);
echo $stream->finish();
?>
```

HttpDeflateStream::factory

(PECL pecl_http:1.4.0-1.5.5)

HttpDeflateStream::factory — HttpDeflateStream クラスのファクトリ

説明

public HttpDeflateStream **HttpDeflateStream::factory** ([int `$flags = 0` [, string `$class_name = 'HttpDeflateStream'`]])

新しい HttpDeflateStream オブジェクトのインスタンスを作成します。

flags に指定する内容については 圧縮ストリームの定数の表 を参照ください。

パラメータ

flags

初期化フラグ。

class_name

HttpDeflateStream のサブクラスの名前。

参考

- HttpDeflateStream::**__construct()**

HttpDeflateStream::__construct

(PECL pecl_http:0.21.0-1.5.5)

HttpDeflateStream::__construct — HttpDeflateStream クラスのコンストラクタ

説明

public void **HttpDeflateStream::__construct** ([int `$flags = 0`])

新しい HttpDeflateStream クラスのインスタンスを作成します。

`flags` に指定する内容については 圧縮ストリームの定数の表 を参照ください。

パラメータ

`flags`

初期化フラグ。

参考

- HttpDeflateStream::factory()

HttpDeflateStream::update

(PECL pecl_http:0.21.0-1.5.5)

HttpDeflateStream::update — 圧縮ストリームを更新する

説明

public string **HttpDeflateStream::update** (string `$data`)

圧縮ストリームに、さらにデータを渡します。

パラメータ

`data`

圧縮するデータ。

返り値

成功した場合は圧縮したデータ、失敗した場合は **`FALSE`** を返します。

HttpDeflateStream::flush

(PECL pecl_http:0.21.0-1.5.5)

HttpDeflateStream::flush — 圧縮ストリームをフラッシュする

説明

public string **HttpDeflateStream::flush** ([string `$data`])

圧縮ストリームをフラッシュします。

パラメータ

`data`

さらに圧縮するデータ。

返り値

成功した場合は、圧縮されたデータを文字列で返します。 失敗した場合は **FALSE** を返します。

HttpDeflateStream::finish

(PECL pecl_http:0.21.0-1.5.5)

HttpDeflateStream::finish — 圧縮ストリームを終了する

説明

public string **HttpDeflateStream::finish** ([string $data])

圧縮ストリームを終了します。圧縮ストリームは、 終了した後でも再開することが可能です。

パラメータ

data

圧縮するデータ。

返り値

圧縮されたデータの最後の部分を返します。

HttpInflateStream

(No version information available, might be only in CVS)

HttpInflateStream — HTTP 展開ストリーム

```
class HttpInflateStream
```

クラスのメンバ

定数

型	名前	説明
int	FLUSH_NONE	フラッシュを強制しない
int	FLUSH_SYNC	フラッシュを同期させる
int	FLUSH_FULL	完全にフラッシュする

注意: フラッシュは、展開ストリームにおいては通常何の意味も持ちません。

メソッド

- HttpInflateStream::**factory()**
- HttpInflateStream::**__construct()**
- HttpInflateStream::**update()**
- HttpInflateStream::flush()
- HttpInflateStream::**finish()**

例

Example#1 HttpInflateStream の例

```
<?php
$stream = new HttpInflateStream;
echo $stream->update($data);
echo $stream->finish();
?>
```

HttpInflateStream::factory

(PECL pecl_http:1.4.0-1.5.5)

HttpInflateStream::factory — HttpInflateStream クラスのファクトリ

説明

public HttpInflateStream **HttpInflateStream::factory** ([int `$flags = 0` [, string `$class_name = 'HttpInflateStream'`]])

新しい HttpInflateStream オブジェクトのインスタンスを作成します。

flags に指定する内容については 展開ストリームの定数の表 を参照ください。

パラメータ

flags

初期化フラグ。

class_name

HttpInflateStream のサブクラスの名前。

参考

- HttpInflateStream::__construct()

HttpInflateStream::__construct

(PECL pecl_http:1.0.0-1.5.5)

HttpInflateStream::__construct — HttpInflateStream クラスのコンストラクタ

説明

public void **HttpInflateStream::__construct** ([int `$flags = 0`])

新しい HttpInflateStream オブジェクトのインスタンスを作成します。

flags に指定する内容については 展開ストリームの定数の表 を参照ください。

パラメータ

flags

展開フラグ。

参考

- HttpInflateStream::factory()

HttpInflateStream::update

(PECL pecl_http:0.21.0-1.5.5)

HttpInflateStream::update — 展開ストリームを更新する

説明

public string **HttpInflateStream::update** (string `$data`)

展開ストリームに、さらにデータを渡します。

パラメータ

data

展開するデータ。

返り値

成功した場合は展開したデータ、失敗した場合は **FALSE** を返します。

HttpInflateStream::flush

(PECL pecl_http:0.21.0-1.5.5)

HttpInflateStream::flush — 展開ストリームをフラッシュする

説明

public string **HttpInflateStream::flush** ([string `$data`])

展開ストリームをフラッシュします。

> **注意**: 展開ストリームにおいては、フラッシュは通常は何の意味もありません。

パラメータ

`data`

> さらに展開するデータ。

返り値

成功した場合は、展開されたデータを文字列で返します。 失敗した場合は **FALSE** を返します。

HttpInflateStream::finish

(PECL pecl_http:0.21.0-1.5.5)

HttpInflateStream::finish — 展開ストリームを終了する

説明

public string **HttpInflateStream::finish** ([string `$data`])

展開ストリームを終了します。展開ストリームは、 終了した後でも再開することが可能です。

パラメータ

`data`

> 展開するデータ。

返り値

展開されたデータの最後の部分を返します。

HttpRequest

(No version information available, might be only in CVS)

HttpRequest — HTTP リクエストクラス

```
class HttpRequest
```

クラスのメンバ

プロパティ

インスタンスプロパティ

アクセス範囲	型	名前	説明
protected	array	options	リクエストを設定するオプション。リクエストのオプション を参照ください。
protected	array	postFields	フォームのデータ。 *array("フィールド名" => "フィールドの値")*
protected	array	postFiles	アップロードするファイル。 *array(array("name" => "image", "file" => "/home/u/images/u.png", "type" => "image*
protected	array	responseInfo	リクエスト/レスポンスについての (統計上の) 情報。リクエスト/レスポンス の情報 を参照ください。
protected	HttpMessage	responseMessage	レスポンスメッセージ。
protected	integer	responseCode	レスポンスコードを表す数値。
protected	string	responseStatus	レスポンスのステータスを表すリテラル文字列。
protected	integer	method	使用するリクエストメソッド。
protected	string	url	リクエスト url。
protected	string	contentType	生の post リクエストで使用する content type。
protected	string	rawPostData	生の post データ。
protected	string	queryData	クエリパラメータ。
protected	string	putFile	PUT リクエストでアップロードするファイル。
protected	string	putData	PUT リクエストでアップロードする生のデータ。
protected	HttpMessage	history	履歴の記録が有効な場合の、リクエスト/レスポンス全体の履歴。
public	boolean	recordHistory	履歴を記録するかどうか。

定義済み定数

型	名前	説明
integer	METH_GET	GET リクエストメソッド。
integer	METH_HEAD	HEAD リクエストメソッド。
integer	METH_POST	POST リクエストメソッド。
integer	METH_PUT	PUT リクエストメソッド。
integer	METH_DELETE	DELETE リクエストメソッド。
integer	METH_OPTIONS	OPTIONS リクエストメソッド。
integer	METH_TRACE	TRACE リクエストメソッド。
integer	METH_CONNECT	CONNECT リクエストメソッド。
integer	METH_PROPFIND	PROPFIND リクエストメソッド。
integer	METH_PROPPATCH	PROPPATCH リクエストメソッド。
integer	METH_MKCOL	MKCOL リクエストメソッド。
integer	METH_COPY	COPY リクエストメソッド。
integer	METH_MOVE	MOVE リクエストメソッド。
integer	METH_LOCK	LOCK リクエストメソッド。
integer	METH_UNLOCK	UNLOCK リクエストメソッド。
integer	METH_VERSION_CONTROL	VERSION-CONTROL リクエストメソッド。
integer	METH_REPORT	REPORT リクエストメソッド。
integer	METH_CHECKOUT	CHECKOUT リクエストメソッド。
integer	METH_CHECKIN	CHECKIN リクエストメソッド。
integer	METH_UNCHECKOUT	UNCHECKOUT リクエストメソッド。
integer	METH_MKWORKSPACE	MKWORKSPACE リクエストメソッド。
integer	METH_UPDATE	UPDATE リクエストメソッド。
integer	METH_LABEL	LABEL リクエストメソッド。
integer	METH_MERGE	MERGE リクエストメソッド。

型	名前	説明
integer	METH_BASELINE_CONTROL	BASELINE-CONTROL リクエストメソッド。
integer	METH_MKACTIVITY	MKACTIVITY リクエストメソッド。
integer	METH_ACL	ACL リクエストメソッド。
integer	VERSION_1_0	HTTP プロトコル バージョン 1.0。
integer	VERSION_1_1	HTTP プロトコル バージョン 1.1。
integer	VERSION_ANY	任意の HTTP プロトコルバージョン。
integer	AUTH_BASIC	ベーシック認証。
integer	AUTH_DIGEST	ダイジェスト認証。
integer	AUTH_NTLM	NTLM 認証。
integer	AUTH_GSSNEG	GSS ネゴシエート認証。
integer	AUTH_ANY	任意の認証。
integer	PROXY_SOCKS4	SOCKS v4 プロキシ。
integer	PROXY_SOCKS5	SOCKS v5 プロキシ。
integer	PROXY_HTTP	HTTP プロキシ。
integer	SSL_VERSION_TLSv1	TLS v1 を使用します。
integer	SSL_VERSION_SSLv2	SSL v2 を使用します。
integer	SSL_VERSION_SSLv3	SSL v3 を使用します。
integer	SSL_VERSION_ANY	任意の SSL/TLS メソッドを使用します。
integer	IPRESOLVE_V4	IPv4 での解決のみを行います。
integer	IPRESOLVE_V6	IPv6 での解決のみを行います。
integer	IPRESOLVE_ANY	任意の方法で解決を行います。

メソッド

- HttpRequest::**construct()**
- HttpRequest::**getOptions()**
- HttpRequest::**setOptions()**
- HttpRequest::**getSslOptions()**
- HttpRequest::**addSslOptions()**
- HttpRequest::**setSslOptions()**
- HttpRequest::**getUrl()**
- HttpRequest::**setUrl()**
- HttpRequest::**getMethod()**
- HttpRequest::**setMethod()**
- HttpRequest::**enableCookies()**
- HttpRequest::**resetCookies()**
- HttpRequest::**getCookies()**
- HttpRequest::**addCookies()**
- HttpRequest::**setCookies()**
- HttpRequest::**getHeaders()**
- HttpRequest::**addHeaders()**
- HttpRequest::**setHeaders()**
- HttpRequest::**getContentType()**
- HttpRequest::**setContentType()**
- HttpRequest::**getQueryData()**
- HttpRequest::**addQueryData()**
- HttpRequest::**setQueryData()**
- HttpRequest::**getPostFields()**
- HttpRequest::**addPostFields()**
- HttpRequest::**setPostFields()**
- HttpRequest::**getPostFiles()**
- HttpRequest::**addPostFile()**
- HttpRequest::**setPostFiles()**
- HttpRequest::**getRawPostData()**
- HttpRequest::**addRawPostData()**
- HttpRequest::**setRawPostData()**
- HttpRequest::**getPutData()**

- HttpRequest::addPutData()
- HttpRequest::setPutData()
- HttpRequest::getPutFile()
- HttpRequest::setPutFile()
- HttpRequest::send()
- HttpRequest::getRequestMessage()
- HttpRequest::getRawRequestMessage()
- HttpRequest::getResponseMessage()
- HttpRequest::getRawResponseMessage()
- HttpRequest::getResponseCode()
- HttpRequest::getResponseStatus()
- HttpRequest::getResponseHeader()
- HttpRequest::getResponseCookies()
- HttpRequest::getResponseBody()
- HttpRequest::getResponseData()
- HttpRequest::getResponseInfo()
- HttpRequest::clearHistory()
- HttpRequest::getHistory()

HttpRequest::addCookies

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::addCookies — クッキーを追加する

説明

public bool **HttpRequest::addCookies** (array `$cookies`)

独自のクッキーを追加します。

> **注意**: リクエストのオプション *encodecookies* は、クッキーの値を urlencode() すべきかどうかを制御します。
>
> **注意**: すべてのリクエストメソッドについて有効です。

パラメータ

cookies

> 追加するクッキーの 名前/値 の組み合わせを含む連想配列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 HttpRequest::addCookies() の例

```
<?php
$r = new HttpRequest;
$r->addCookies(
     array(
         "cookie_name"  => "cookie  value",
     )
);
?>
```

参考

- HttpRequest::setCookies()

HttpRequest::addHeaders

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::addHeaders — ヘッダを追加する

説明

public bool **HttpRequest::addHeaders** (array `$headers`)

リクエストヘッダの 名前/値 の組み合わせを追加します。

パラメータ

`headers`

追加するヘッダの 名前/値 の組み合わせを含む連想配列。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

HttpRequest::addPostFields

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::addPostFields — POST フィールドを追加する

説明

public bool **HttpRequest::addPostFields** (array `$post_data`)

POST データエントリを追加します。既存のデータは、 同名のエントリを指定されない限りはそのまま残ります。

POST およびカスタムリクエストについてのみ影響します。

パラメータ

`post_data`

POST フィールドを含む連想配列。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

HttpRequest::addPostFile

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::addPostFile — POST ファイルを追加する

説明

public bool **HttpRequest::addPostFile** (string `$name` , string `$file` [, string `$content_type = 'application/x-octetstream'`])

ファイルを POST リクエストに追加します。事前に設定されているファイルは変更されません。

POST およびカスタムリクエストについてのみ影響します。 生の POST データには使用できません。

パラメータ

`name`

フォーム要素の名前。

`file`

ファイルへのパス。

`content_type`

ファイルの content type。

返り値

成功した場合に TRUE を返します。 content type にプライマリパート、セカンダリパートが含まれていない場合に FALSE を返します。

HttpRequest::addPutData

(PECL pecl_http:0.25.0-1.5.5)

HttpRequest::addPutData — PUT データを追加する

説明

public bool **HttpRequest::addPutData** (string `$put_data`)

PUT データを追加します。事前に設定されている PUT データはそのまま残ります。

PUT リクエストについてのみ影響します。

パラメータ

`put_data`

連結するデータ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::addQueryData

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::addQueryData — クエリデータを追加する

説明

public bool **HttpRequest::addQueryData** (array `$query_params`)

クエリパラメータのリストにパラメータを追加します。 既存のデータはそのまま残ります。

あらゆる型のリクエストに影響します。

パラメータ

`query_params`

追加するクエリフィールドを含む連想配列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::addRawPostData

(PECL pecl_http:0.14.0-1.4.1)

HttpRequest::addRawPostData — 生の POST データを追加する

説明

public bool **HttpRequest::addRawPostData** (string `$raw_post_data`)

生の POST データを追加します。既存のデータはそのまま変更されません。

POST およびカスタムリクエストについてのみ影響します。

パラメータ

raw_post_data

連結する生の POST データ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::addSslOptions

(PECL pecl_http:0.12.0-1.5.5)

HttpRequest::addSslOptions — SSL オプションを追加する

説明

public bool **HttpRequest::addSslOptions** (array `$options`)

追加の SSL オプションを設定します。

パラメータ

options

追加の SSL オプションを含む連想配列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::clearHistory

(PECL pecl_http:0.15.0-1.5.5)

HttpRequest::clearHistory — 履歴を消去する

説明

public void **HttpRequest::clearHistory** (void)

すべての履歴メッセージを消去します。

注意: 履歴が記録されるのは *recordHistory* が有効な場合のみです。

HttpRequest::__construct

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::__construct — HttpRequest のコンストラクタ

説明

public void **HttpRequest::__construct** ([string `$url` [, int `$request_method = HTTP_METH_GET` [, array `$options`]]])

新しい HttpRequest オブジェクトのインスタンスを作成します。

パラメータ

url

リクエスト対象の URL。

request_method

使用するリクエストメソッド。

options

リクエストオプションの連想配列。

エラー / 例外

HttpException をスローします。

HttpRequest::enableCookies

(PECL pecl_http:1.0.0-1.5.5)

HttpRequest::enableCookies — クッキーを有効にする

説明

public bool **HttpRequest::enableCookies** (void)

受け取ったクッキーを自動送信するようにします。

注意: いずれにせよ、独自に設定されたクッキーが送信されることに注意しましょう。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::getContentType

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getContentType — content type を取得する

説明

public string **HttpRequest::getContentType** (void)

事前に設定された content type を取得します。

返り値

事前に設定された content type を文字列で返します。

HttpRequest::getCookies

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getCookies — クッキーを取得する

説明

public array **HttpRequest::getCookies** (void)

事前に設定されたクッキーを取得します。

返り値

事前に設定されたクッキーを含む連想配列を返します。

HttpRequest::getHeaders

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getHeaders — ヘッダを取得する

説明

public array **HttpRequest::getHeaders** (void)

事前に設定されたリクエストヘッダを取得します。

返り値

現在設定されているすべてのヘッダを含む連想配列を返します。

HttpRequest::getHistory

(PECL pecl_http:0.15.0-1.5.5)

HttpRequest::getHistory — 履歴を取得する

説明

public HttpMessage **HttpRequest::getHistory** (void)

すべての送信済みリクエストと受信したレスポンスを、 HttpMessage オブジェクトとして取得します。

履歴を記録したい場合は、インスタンス変数 HttpRequest::*recordHistory* を TRUE に設定します。

返されるオブジェクトは、直近に受け取ったレスポンスを参照しています。 それ以前に送信したリクエストや受信したレスポンスにアクセスするには **HttpMessage::getParentMessage()()** を使用します。

返り値

リクエスト/レスポンス の履歴を表す HttpMessage オブジェクトを返します。

エラー / 例外

HttpRuntimeException をスローします。

HttpRequest::getMethod

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getMethod — メソッドを取得する

説明

public int **HttpRequest::getMethod** (void)

事前に設定されたリクエストメソッドを取得します。

返り値

現在設定されているリクエストメソッドを返します。

HttpRequest::getOptions

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getOptions — オプションを取得する

説明

public array **HttpRequest::getOptions** (void)

現在設定されているオプションを取得します。

返り値

現在設定されているオプションを含む連想配列を返します。

HttpRequest::getPostFields

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getPostFields — POST フィールドを取得する

説明

public array **HttpRequest::getPostFields** (void)

事前に設定された POST データを取得します。

パラメータ

返り値

現在設定されている POST フィールドを連想配列で返します。

HttpRequest::getPostFiles

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getPostFiles — POST ファイルを取得する

説明

public array **HttpRequest::getPostFiles** (void)

事前に追加された POST ファイルをすべて取得します。

返り値

現在設定されている POST ファイルを含む配列を返します。

HttpRequest::getPutData

(PECL pecl_http:0.25.0-1.5.5)

HttpRequest::getPutData — PUT データを取得する

説明

public string **HttpRequest::getPutData** (void)

事前に設定された PUT データを取得します。

返り値

現在設定されている PUT データを文字列で返します。

HttpRequest::getPutFile

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getPutFile — PUT ファイルを取得する

説明

public string **HttpRequest::getPutFile** (void)

事前に設定された PUT ファイルを取得します。

返り値

現在設定されている PUT ファイルへのパスを含む文字列を返します。

HttpRequest::getQueryData

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getQueryData — クエリデータを取得する

説明

public string **HttpRequest::getQueryData** (void)

現在のクエリデータを、urlencode された形式の文字列で取得します。

返り値

urlencode されたクエリを含む文字列を返します。

HttpRequest::getRawPostData

(PECL pecl_http:0.14.0-1.4.1)

HttpRequest::getRawPostData — 生の POST データを取得する

説明

public string **HttpRequest::getRawPostData** (void)

事前に設定された生の POST データを取得します。

パラメータ

返り値

現在設定されている生の POST 一他を含む文字列を返します。

HttpRequest::getRawRequestMessage

(PECL pecl_http:0.21.0-1.5.5)

HttpRequest::getRawRequestMessage — 名前のリクエストメッセージを取得する

説明

public string **HttpRequest::getRawRequestMessage** (void)

送信された HTTP メッセージを取得します。

パラメータ

返り値

HttpMessage を文字列形式で返します。

HttpRequest::getRawResponseMessage

(PECL pecl_http:0.21.0-1.5.5)

HttpRequest::getRawResponseMessage — 生のレスポンスメッセージを取得する

説明

public string **HttpRequest::getRawResponseMessage** (void)

HTTP レスポンスの全体を取得します。

パラメータ

返り値

ウェブサーバからの完全なレスポンスを返します。 ここには、文字列形式のヘッダも含まれます。

HttpRequest::getRequestMessage

(PECL pecl_http:0.11.0-1.5.5)

HttpRequest::getRequestMessage — リクエストメッセージを取得する

説明

public HttpMessage **HttpRequest::getRequestMessage** (void)

送信された HTTP メッセージを取得します。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。このリクエストにおいて その前に送信したリクエストの内容にアクセスするには **HttpMessage::getParentMessage()** を使用します。

> **注意**: 内部的なリクエストメッセージは不変であることに注意しましょう。 つまり、**HttpRequest::getRequestMessage()** で受け取るリクエストメッセージは、同一リクエストにおいては 常に同じになるということです。これは、リクエストオブジェクトに 何か変更を加えた場合でも同じです。

返り値

送信されたリクエストを表す HttpMessage オブジェクトを返します。

エラー / 例外

HttpMalformedHeadersException、HttpEncodingException をスローします。

HttpRequest::getResponseBody

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getResponseBody — レスポンスの本文を取得する

説明

public string **HttpRequest::getResponseBody** (void)

リクエストが送信された後に、レスポンスの本文を取得します。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。

パラメータ

返り値

レスポンスの本文を含む文字列を返します。

HttpRequest::getResponseCode

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getResponseCode — レスポンスコードを取得する

説明

public int **HttpRequest::getResponseCode** (void)

リクエストが送信された後に、レスポンスコードを取得します。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。

返り値

レスポンスコードを整数値で返します。

HttpRequest::getResponseCookies

(PECL pecl_http:0.23.0-1.5.5)

HttpRequest::getResponseCookies — レスポンスのクッキーを取得する

説明

public array **HttpRequest::getResponseCookies** ([int `$flags` [, array `$allowed_extras`]])

リクエストが送信された後に、レスポンスのクッキーを取得します。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。

パラメータ

`flags`

　http_parse_cookie() のフラグ。

`allowed_extras`

　キーを、クッキー名ではなく追加情報として扱うようにする。

返り値

http_parse_cookie() が返すのと同様に、 stdClass オブジェクトの配列を返します。

HttpRequest::getResponseData

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getResponseData — レスポンスデータを取得する

説明

public array **HttpRequest::getResponseData** (void)

リクエストが送信された後に、すべてのレスポンスデータを取得します。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。

返り値

キー "headers"、"body" を持つ連想配列を返します。 "headers" にはすべてのレスポンスヘッダが連想配列として含まれ、 "body" にはレスポンス本文が文字列で含まれます。

HttpRequest::getResponseHeader

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getResponseHeader — レスポンスヘッダを取得する

説明

public mixed **HttpRequest::getResponseHeader** ([string $name])

リクエストが送信された後に、レスポンスヘッダを取得します。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。

パラメータ

name

読み込むヘッダ。空の場合はすべてのレスポンスヘッダが返されます。

返り値

name を指定した場合は、それに一致する名前のヘッダの値を文字列で返します。 失敗した場合は FALSE を返します。name を指定しなかった場合は、 すべてのレスポンスヘッダを含む連想配列を返します。

HttpRequest::getResponseInfo

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getResponseInfo — レスポンスの情報を取得する

説明

public mixed **HttpRequest::getResponseInfo** ([string $name])

リクエストが送信された後に、レスポンスの情報を取得します。

どのような 情報 が返されるのかについては、 http_get() を参照ください。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。

パラメータ

name

読み込む情報。空だったり省略したりした場合は、 取得可能なすべての銃尾法を含む連想配列が返されます。

返り値

name を指定した場合は、それに一致する名前の情報をスカラ値で返します。 失敗した場合は FALSE を返します。name を指定しなかった場合は、取得可能なすべての情報を含む連想配列を返します。

HttpRequest::getResponseMessage

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getResponseMessage — レスポンスメッセージを取得する

説明

public HttpMessage **HttpRequest::getResponseMessage** (void)

リクエストが送信された後に、完全なレスポンスを HttpMessage オブジェクトとして取得します。

リダイレクトが許可されており、複数のレスポンスを受信した場合は、 最後に受信したレスポンスを参照します。このリクエストにおいて その前に送信したリクエストの内容にアクセスするには **HttpMessage::getParentMessage()** を使用します。

返り値

レスポンスを表す HttpMessage オブジェクトを返します。

エラー / 例外

HttpException、HttpRuntimeException をスローします。

HttpRequest::getResponseStatus

(PECL pecl_http:0.23.0-1.5.5)

HttpRequest::getResponseStatus — レスポンスのステータスを取得する

説明

public string **HttpRequest::getResponseStatus** (void)

メッセージが送信された後に、レスポンスのステータス (レスポンスコードの後に続く文字列) を取得します。

返り値

レスポンスステータスのテキストを文字列で返します。

HttpRequest::getSslOptions

(No version information available, might be only in CVS)

HttpRequest::getSslOptions — ssl オプションを取得する

説明

public array **HttpRequest::getSslOptions** (void)

事前に設定された SSL オプションを取得します。

返り値

事前に設定された SSL オプションを含む連想配列を返します。

HttpRequest::getUrl

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::getUrl — url を取得する

説明

public string **HttpRequest::getUrl** (void)

事前に設定されたリクエスト URL を取得します。

返り値

現在設定されている url を文字列で返します。

HttpRequest::resetCookies

(PECL pecl_http:1.0.0-1.5.5)

HttpRequest::resetCookies — クッキーをリセットする

説明

public bool **HttpRequest::resetCookies** ([bool `$session_only = FALSE`])

自動的に受信/送信されたクッキーを、リセットします。

> **注意**: 独自に設定したクッキーには影響を及ぼさないことに注意しましょう。

パラメータ

`session_only`

> セッションクッキーのみをリセットするかどうかを指定します (libcurl >= v7.15.4 あるいは libcurl >= v7.14.1 が必要です)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::send

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::send — リクエストを送信する

説明

public HttpMessage **HttpRequest::send** (void)

HTTP リクエストを送信します。

> **注意**: 例外がスローされた場合でも、少なくとも部分的には転送に成功しているかもしれません。 そこで、さまざまな HttpRequest::getResponse*() メソッドを使用して 返り値を調べるようにしましょう。

返り値

受信したレスポンスを HttpMessage オブジェクトとして返します。

エラー / 例外

HttpRuntimeException、HttpRequestException、 HttpMalformedHeaderException、HttpEncodingException をスローします。

例

Example#1 GET example

```
<?php
$r = new HttpRequest('http://example.com/feed.rss', HttpRequest::METH_GET);
$r->setOptions(array('lastmodified' => filemtime('local.rss')));
$r->addQueryData(array('category' => 3));
try {
    $r->send();
    if ($r->getResponseCode() == 200) {
        file_put_contents('local.rss', $r->getResponseBody());
    }
} catch (HttpException $ex) {
    echo $ex;
}
?>
```

Example#2 POST example

```
<?php
$r = new HttpRequest('http://example.com/form.php', HttpRequest::METH_POST);
$r->setOptions(array('cookies' => array('lang' => 'de')));
$r->addPostFields(array('user' => 'mike', 'pass' => 's3c!r3t'));
$r->addPostFile('image', 'profile.jpg', 'image/jpeg');
try {
    echo $r->send()->getBody();
} catch (HttpException $ex) {
    echo $ex;
}
?>
```

HttpRequest::setContentType

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setContentType — content type を設定する

説明

public bool **HttpRequest::setContentType** (string `$content_type`)

post リクエストの content type を設定します。

パラメータ

`content_type`

リクエストの content type (primary/secondary)。

返り値

成功した場合に TRUE を返します。 content type にプライマリパート、セカンダリパートが含まれていない場合に FALSE を返します。

HttpRequest::setCookies

(PECL pecl_http:0.12.0-1.5.5)

HttpRequest::setCookies — クッキーを設定する

説明

public bool **HttpRequest::setCookies** ([array `$cookies`])

独自のクッキーを設定します。

パラメータ

`cookies`

クッキーの 名前/値 の組み合わせを含む連想配列。 空の配列を渡したり省略したりした場合は、 これまでに設定されているクッキーがすべて削除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setHeaders

(PECL pecl_http:0.12.0-1.5.5)

HttpRequest::setHeaders — ヘッダを設定する

説明

public bool **HttpRequest::setHeaders** ([array `$headers`])

リクエストヘッダの 名前/値 の組み合わせを設定します。

パラメータ

`headers`

ヘッダの 名前/値 の組み合わせを含む連想配列。 空の配列を渡したり省略したりした場合は、 これまでに設定されているヘッダがすべて削除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setMethod

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setMethod — メソッドを設定する

説明

public bool **HttpRequest::setMethod** (int `$request_method`)

リクエストメソッドを設定します。

パラメータ

`request_method`

使用するリクエストメソッド。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setOptions

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setOptions — オプションを設定する

説明

public bool **HttpRequest::setOptions** ([array `$options`])

使用するリクエストオプションを設定します。

リクエストのオプション を参照ください。

パラメータ

`options`

連想配列で、その値が現在のリクエストオプションを上書きします。 空の配列を渡したり省略したりした場合は、 HttpRequest オブジェクトのオプションがリセットされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setPostFields

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setPostFields — POST フィールドを設定する

説明

public bool **HttpRequest::setPostFields** (array `$post_data`)

POST データエントリを設定します。事前に設定されている POST データが上書きされます。

POST およびカスタムリクエストについてのみ影響します。

パラメータ

`post_data`

POST フィールドを含む連想配列。 空の配列を渡した場合は、 POST データが削除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setPostFiles

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setPostFiles — POST ファイルを設定する

説明

public bool **HttpRequest::setPostFiles** (array `$post_files`)

POST するファイルを設定します。事前に設定されているファイルを上書きします。

POST およびカスタムリクエストについてのみ影響します。 生の POST データには使用できません。

パラメータ

`post_files`

POST するファイルを含む配列。 空の配列を渡した場合は、設定が削除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setPutData

(PECL pecl_http:0.25.0-1.5.5)

HttpRequest::setPutData — PUT データを設定する

説明

public bool **HttpRequest::setPutData** ([string `$put_data`])

PUT データを追加します。事前に設定されている PUT データを上書きします。

PUT リクエストについてのみ影響します。

各リクエストについて、PUT データあるいは PUT ファイルの どちらか一方のみが使用されます。PUT データのほうが優先順位が高いので、 たとえ PUT ファイルが設定されていたとしても PUT データがあればそちらを使用します。

パラメータ

`put_data`

アップロードするデータ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setPutFile

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setPutFile — PUT ファイルを設定する

説明

public bool **HttpRequest::setPutFile** ([string `$file`])

PUT するファイルを設定します。 PUT リクエストについてのみ影響します。

パラメータ

file

送信するファイルへのパス。 空の文字列を渡したり省略したりした場合は、 設定が削除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setQueryData

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setQueryData — クエリデータを設定する

説明

public bool **HttpRequest::setQueryData** (mixed `$query_data`)

使用する URL クエリパラメータを設定します。 事前に設定されているクエリパラメータを上書きします。

あらゆる型のリクエストに影響します。

パラメータ

query_data

エンコード済みのクエリ文字列、 あるいはエンコードするクエリフィールドを含む連想配列。 空のデータを渡した場合は、 クエリデータの設定が削除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setRawPostData

(PECL pecl_http:0.14.0-1.4.1)

HttpRequest::setRawPostData — 生の POST データを設定する

説明

public bool **HttpRequest::setRawPostData** ([string `$raw_post_data`])

生の POST データを追加します。事前に設定されているデータは上書きされます。 content type を指定することを忘れないでください。 POST およびカスタムリクエストについてのみ影響します。

各リクエストについて、POST フィールドあるいは生の POST データの どちらか一方のみが使用されます。生の POST データのほうが優先順位が高いので、 たとえ POST フィールドが設定されていたとしても 生の POST データがあればそちらを使用します。

パラメータ

raw_post_data

生の POST データ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpRequest::setSslOptions

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setSslOptions — SSL オプションを設定する

説明

public bool **HttpRequest::setSslOptions** ([array `$options`])

SSL オプションを設定します。

パラメータ

`options`

SSL オプションを含む連想配列。 空の配列を渡したり省略したりした場合は、 SSL オプションの設定が削除されます。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

HttpRequest::setUrl

(PECL pecl_http:0.10.0-1.5.5)

HttpRequest::setUrl — URL を設定する

説明

public bool **HttpRequest::setUrl** (string `$url`)

リクエスト URL を設定します。

パラメータ

`url`

リクエスト URL。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

HttpRequestPool

(No version information available, might be only in CVS)

HttpRequestPool — HTTP リクエストプールクラス

```
class HttpRequest implements Iterator, Countable
```

クラスのメンバ

プロパティ

HttpRequestPool クラスのプロパティはありません。

定義済み定数

HttpRequestPool クラスの定数はありません。

メソッド

- HttpRequestPool::**construct()**
- HttpRequestPool::**destruct()**
- HttpRequestPool::**attach()**
- HttpRequestPool::**detach()**

HttpRequestPool::attach

(PECL pecl_http:0.10.0-1.5.5)

HttpRequestPool::attach — HttpRequest をアタッチする

説明

public bool **HttpRequestPool::attach** (HttpRequest `$request`)

HttpRequest オブジェクトを、この HttpRequestPool にアタッチします。

警告

アタッチする前に、すべてのオプションを設定しておいてください!

パラメータ

`request`

まだどの HttpRequestPool オブジェクトにもアタッチされていない HttpRequest オブジェクト。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

エラー / 例外

HttpInvalidParamException、HttpRequestException、 HttpRequestPoolException、HttpEncodingException をスローします。

HttpRequestPool::__construct

(PECL pecl_http:0.10.0-1.5.5)

HttpRequestPool::__construct — HttpRequestPool のコンストラクタ

説明

void **HttpRequestPool::__construct** ([HttpRequest `$request`])

新しい HttpRequestPool オブジェクトのインスタンスを作成します。 HttpRequestPool は、複数の HttpRequest を平行して送信することができます。

事実上無限にオプションのパラメータを指定することができます。 各パラメータは HttpRequest オブジェクトへの参照です。

パラメータ

`request`

アタッチする HttpRequest オブジェクト。

エラー / 例外

HttpRequestPoolException (HttpRequestException、HttpInvalidParamException) をスローします。

例

Example#1 HttpRequestPool の例

```
<?php
```

```
try {
    $pool = new HttpRequestPool(
        new HttpRequest('http://www.google.com/', HttpRequest::METH_HEAD),
        new HttpRequest('http://www.php.net/', HttpRequest::METH_HEAD)
    );
    $pool->send();
    foreach($pool as $request) {
        printf("%s is %s (%d)¥n",
            $request->getUrl(),
            $request->getResponseCode() ? 'alive' : 'not alive',
            $request->getResponseCode()
        );
    }
} catch (HttpException $e) {
    echo $e;
}
?>
```

HttpRequestPool::__destruct

(PECL pecl_http:0.10.0-1.5.5)

HttpRequestPool::__destruct — HttpRequestPool のデストラクタ

説明

void **HttpRequestPool::__destruct** (void)

HttpRequestPool オブジェクトの後始末をします。

HttpRequestPool::detach

(PECL pecl_http:0.10.0-1.5.5)

HttpRequestPool::detach — HttpRequest をデタッチする

説明

bool **HttpRequestPool::detach** (HttpRequest `$request`)

この HttpRequestPool から、HttpRequest オブジェクトをデタッチします。

パラメータ

`request`

この HttpRequestPool オブジェクトにアタッチされている HttpRequest オブジェクト。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

エラー / 例外

HttpInvalidParamException、HttpRequestPoolException をスローします。

HttpRequestPool::getAttachedRequests

(PECL pecl_http:0.16.0-1.5.5)

HttpRequestPool::getAttachedRequests — アタッチされているリクエストを取得する

説明

array **HttpRequestPool::getAttachedRequests** (void)

アタッチされている HttpRequest オブジェクトを取得します。

パラメータ

返り値

現在アタッチされているすべての HttpRequest オブジェクトを含む配列を返します。

HttpRequestPool::getFinishedRequests

(PECL pecl_http:0.16.0-1.5.5)

HttpRequestPool::getFinishedRequests — 終了したリクエストを取得する

説明

array **HttpRequestPool::getFinishedRequests** (void)

アタッチされている HttpRequest オブジェクトのうち、 すでに処理を終えた後のものを取得します。

パラメータ

返り値

現在アタッチされているすべての HttpRequest オブジェクトのうち、すでに処理を終えたものを含む配列を返します。

HttpRequestPool::reset

(PECL pecl_http:0.10.0-1.5.5)

HttpRequestPool::reset — リクエストプールをリセットする

説明

void **HttpRequestPool::reset** (void)

アタッチされているすべての HttpRequest オブジェクトを デタッチします。

HttpRequestPool::send

(PECL pecl_http:0.10.0-1.5.5)

HttpRequestPool::send — すべてのリクエストを送信する

説明

bool **HttpRequestPool::send** (void)

アタッチされているすべての HttpRequest オブジェクトを、平行して送信します。

パラメータ

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

HttpRequestPoolException (HttpSocketException、HttpRequestException、HttpMalformedHeaderException) をスローします。

HttpRequestPool::socketPerform

(PECL pecl_http:0.15.0-1.5.5)

HttpRequestPool::socketPerform — ソケットアクションを実行する

説明

protected bool **HttpRequestPool::socketPerform** (void)

各リクエストがトランザクションを終了するまでの間、TRUE を返します。

返り値

各リクエストがトランザクションを終了するまでの間、TRUE を返します。

例

Example#1 HttpRequestPool::socketPerform() の例

```
<?php
class MyPool extends HttpRequestPool
{
     public function send()
    {
         while ($this->socketPerform()) {
             if (!$this->socketSelect()) {
                 throw new HttpSocketExcpetion;
             }
         }
    }

     protected final function socketPerform()
    {
         $result = parent::socketPerform();
          foreach ($this->getFinishedRequests() as $r) {
             $this->detach($r);
            // 終了したリクエストのレスポンスを処理します。
         }
          return $result;
    }
}
?>
```

HttpRequestPool::socketSelect

(PECL pecl_http:0.10.0-1.5.5)

HttpRequestPool::socketSelect — ソケットの選択を実行する

説明

protected bool **HttpRequestPool::socketSelect** (void)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

HttpResponse

(No version information available, might be only in CVS)

HttpResponse — HTTP レスポンスクラス

```
static class HttpResponse
```

クラスのメンバ

プロパティ

静的なプロパティ

アクセス範囲	型	名前	説明
protected	boolean	cache	レスポンスのキャッシュを試みるかどうか。
protected	boolean	gzip	送信するエンティティをその場で gzip するかどうか。
protected	string	eTag	生成された、あるいは独自の ETag。
protected	integer	lastModified	生成された、あるいは独自の最終更新タイムスタンプ。
protected	string	cacheControl	*Cache-Control* の設定。
protected	string	contentType	送信したエンティティの *Content-Type*。
protected	string	contentDisposition	送信したエンティティの *Content-Disposition*。
protected	integer	bufferSize	抑止処理に使用するチャンクバッファサイズ。
protected	double	throttleDelay	抑止処理の遅延秒数。

定義済み定数

型	名前	説明
integer	REDIRECT	適切なリダイレクト方法を判別する。
integer	REDIRECT_PERM	完全なリダイレクト (*301 Moved permanently*)。
integer	REDIRECT_FOUND	標準のリダイレクト (*302 Found*)。
integer	REDIRECT_POST	POST リクエストに対するリダイレクト (*303 See other*)。
integer	REDIRECT_PROXY	プロキシリダイレクト (*305 Use proxy*)。
integer	REDIRECT_TEMP	一時的なリダイレクト (*307 Temporary Redirect*)。

メソッド

- HttpResponse::getBufferSize()
- HttpResponse::setBufferSize()
- HttpResponse::getCacheControl()
- HttpResponse::setCacheControl()
- HttpResponse::getCache()
- HttpResponse::setCache()
- HttpResponse::getContentDisposition()
- HttpResponse::setContentDisposition()
- HttpResponse::getContentType()
- HttpResponse::setContentType()
- HttpResponse::getData()
- HttpResponse::setData()
- HttpResponse::getETag()
- HttpResponse::setETag()
- HttpResponse::getFile()
- HttpResponse::setFile()
- HttpResponse::getGzip()
- HttpResponse::setGzip()
- HttpResponse::getHeader()
- HttpResponse::setHeader()
- HttpResponse::getLastModified()
- HttpResponse::setLastModified()
- HttpResponse::getStream()
- HttpResponse::setStream()
- HttpResponse::getThrottleDelay()
- HttpResponse::setThrottleDelay()
- HttpResponse::redirect()
- HttpResponse::capture()
- HttpResponse::send()
- HttpResponse::guessContentType()
- HttpResponse::redirect()
- HttpResponse::status()
- HttpResponse::getRequestHeaders()
- HttpResponse::getRequestBody()
- HttpResponse::getRequestBodyStream()

HttpResponse::capture

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::capture — スクリプトの出力を取り込む

説明

static void **HttpResponse::capture** (void)

スクリプトの出力を取り込みます。

例

Example#1 HttpResponse::capture() の例

```
<?php
HttpResponse::setCache(true);
HttpResponse::capture();
// この後にスクリプトが続きます
?>
```

参考

- HttpResponse::send()

HttpResponse::getBufferSize

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getBufferSize — バッファサイズを取得する

説明

static int **HttpResponse::getBufferSize** (void)

現在のバッファのサイズを取得します。

返り値

現在のバッファのサイズをバイト単位で返します。

参考

- HttpResponse::setBufferSize()
- HttpResponse::getThrottleDelay()
- HttpResponse::setThrottleDelay()

HttpResponse::getCacheControl

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getCacheControl — cache control を取得する

説明

static string **HttpResponse::getCacheControl** (void)

現在の *Cache-Control* ヘッダの設定を取得します。

返り値

現在の cache control の設定を、 ヘッダとして送信されるのと同様の形式の文字列で返します。

参考

- HttpResponse::setCacheControl()
- HttpResponse::setCache()
- HttpResponse::getCachel()

HttpResponse::getCache

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getCache — キャッシュを取得する

説明

static bool **HttpResponse::getCache** (void)

現在のキャッシュ設定を取得します。

返り値

キャッシュを行おうとしている場合に **TRUE**、 そうでない場合に **FALSE** を返します。

参考

- HttpResponse::setCacheControl()
- HttpResponse::getCacheControl()
- HttpResponse::setCache()

HttpResponse::getContentDisposition

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getContentDisposition — content disposition を取得する

説明

static string **HttpResponse::getContentDisposition** (void)

現在の *Content-Disposition* の設定を取得します。

返り値

現在の content disposition の設定を、 ヘッダとして送信されるのと同様の形式の文字列で返します。

参考

- HttpResponse::setContentDisposition()
- HttpResponse::getContentType()
- HttpResponse::setContentType()

HttpResponse::getContentType

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getContentType — content type を取得する

説明

static string **HttpResponse::getContentType** (void)

現在の *Content-Type* ヘッダの設定を取得します。

返り値

現在設定されている content type を文字列で返します。

参考

- HttpResponse::getContentDisposition()
- HttpResponse::setContentDisposition()
- HttpResponse::setContentType()
- HttpResponse::guessContentType()

HttpResponse::getData

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getData — データを取得する

説明

static string **HttpResponse::getData** (void)

事前に設定された、送信されようとしているデータを取得します。

返り値

事前に設定された、送信されようとしているデータを含む文字列を返します。

参考

- HttpResponse::setData()
- HttpResponse::getFile()
- HttpResponse::setFile()
- HttpResponse::getStream()
- HttpResponse::setStream()

HttpResponse::getETag

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getETag — ETag を取得する

説明

static string **HttpResponse::getETag** (void)

予測される、あるいは事前に設定された独自の *ETag* を取得します。

返り値

予測される、あるいは事前に設定された独自の *ETag* を、クォートされた文字列で返します。

参考

- HttpResponse::getLastModified()
- HttpResponse::setLastModified()
- HttpResponse::setETag()

HttpResponse::getFile

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getFile — ファイルを取得する

説明

static string **HttpResponse::getFile** (void)

事前に設定された、送信されようとしているファイルを取得します。

返り値

事前に設定された、送信されようとしているファイルを文字列で返します。

参考

- HttpResponse::getData()
- HttpResponse::setData()
- HttpResponse::setFile()
- HttpResponse::getStream()
- HttpResponse::setStream()

HttpResponse::getGzip

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getGzip — gzip を取得する

説明

static bool **HttpResponse::getGzip** (void)

現在の gzip 圧縮設定を取得します。

返り値

GZip 圧縮が有効な場合に **TRUE**、そうでない場合に **FALSE** を返します。

参考

- HttpResponse::setGzip()

HttpResponse::getHeader

(PECL pecl_http:0.12.0-1.5.5)

HttpResponse::getHeader — ヘッダを取得する

説明

static mixed **HttpResponse::getHeader** ([string $name])

送信されようとしているヘッダを取得します。

> **注意**: これは、以下の SAPI では期待通りに動作しないかもしれません。 Apache2 (PHP < 5.1.3 と組み合わせた場合)。

パラメータ

name

読み込むヘッダの名前を指定します。 空の文字列を渡されたり省略されたりした場合は、 すべてのヘッダを含む連想配列が返されます。

返り値

指定した名前のヘッダが存在する場合にはそのヘッダの値を含む文字列、 失敗した場合には **FALSE**、 あるいはすべてのヘッダを含む連想配列を返します。

参考

- HttpResponse::setHeader()

HttpResponse::getLastModified

(PECL pecl_http:0.12.0-1.5.5)

HttpResponse::getLastModified — 最終更新日時を取得する

説明

static int **HttpResponse::getLastModified** (void)

予測される、あるいは事前に設定された *Last-Modified* 日付を取得します。

返り値

予測される、あるいは事前に設定された Unix タイムスタンプを返します。

参考

- HttpResponse::setLastModified()
- HttpResponse::getETag()
- HttpResponse::setETag()

HttpResponse::getStream

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getStream — ストリームを取得する

説明

static resource **HttpResponse::getStream** (void)

事前に設定された、送信されようとしているリソースを取得します。

パラメータ

返り値

事前に設定されたリソースを返します。

参考

- HttpResponse::getData()
- HttpResponse::setData()
- HttpResponse::getFile()
- HttpResponse::setFile()
- HttpResponse::setStream()

HttpResponse::getThrottleDelay

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::getThrottleDelay — throttle delay を取得する

説明

static double **HttpResponse::getThrottleDelay** (void)

現在の throttle delay を取得します。

返り値

秒単位の throttle delay を表す double 値を返します。

参考

- HttpResponse::getBufferSize()
- HttpResponse::setBufferSize()
- HttpResponse::setThrottleDelay()

HttpResponse::getRequestBody

(No version information available, might be only in CVS)

HttpResponse::getRequestBody — リクエストの本文を取得する

説明

static string **HttpResponse::getRequestBody** (void)

この関数は次の関数のエイリアスです。 http_get_request_body()

HttpResponse::getRequestBodyStream

(No version information available, might be only in CVS)

HttpResponse::getRequestBodyStream — リクエストの本文をストリームとして取得する

説明

static resource **HttpResponse::getRequestBodyStream** (void)

この関数は次の関数のエイリアスです。 http_get_request_body_stream()

HttpResponse::getRequestHeaders

(No version information available, might be only in CVS)

HttpResponse::getRequestHeaders — リクエストのヘッダを取得する

説明

static array **HttpResponse::getRequestHeaders** (void)

この関数は次の関数のエイリアスです。 http_get_request_headers()

HttpResponse::guessContentType

(PECL pecl_http:0.13.0-1.5.5)

HttpResponse::guessContentType — content type を推測する

説明

static string **HttpResponse::guessContentType** (string `$magic_file` [, int `$magic_mode=MAGIC_MIME`])

指定された本文の content type を、libmagic を使用して推測します。

推測に成功すると、その結果の *Content-Type* が自動的にレスポンスの *Content-Type* として設定されます。

パラメータ

`magic_file`

使用する magic.mime データベース。

`magic_mode`

libmagic のフラグ。

返り値

成功した場合は推測した content type、 失敗した場合は **FALSE** を返します。

エラー / 例外

HttpRuntimeException、HttpInvalidParamException をスローします。

参考

- HttpResponse::getContentType()
- HttpResponse::setContentType()

HttpResponse::redirect

(No version information available, might be only in CVS)

HttpResponse::redirect — リダイレクトする

説明

static void **HttpResponse::redirect** ([string `$url` [, array `$params` [, bool `$session = FALSE` [, int `$status`]]]])

この関数は次の関数のエイリアスです。 http_redirect()

HttpResponse::send

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::send — レスポンスを送信する

説明

static bool **HttpResponse::send** ([bool `$clean_ob = TRUE`])

エンティティを送信します。

キャッシュ処理に成功すると PHP の処理を終了します。そして、 INI 設定 http.log.cache が設定されている場合にはログにエントリを書き出します。"終了" の意味については、INI 設定 http.force_exit を参照ください。

パラメータ

clean_ob

事前に開始されている出力ハンドラとそのバッファを すべて破壊するかどうか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 HttpResponse::send() の例

```
<?php
HttpResponse::setCache(true);
HttpResponse::setContentType('application/pdf');
HttpResponse::setContentDisposition("$user.pdf", false);
HttpResponse::setFile('sheet.pdf');
HttpResponse::send();
?>
```

参考

- HttpResponse::capture()

HttpResponse::setBufferSize

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setBufferSize — バッファサイズを設定する

説明

static bool **HttpResponse::setBufferSize** (int $bytes)

抑止機構の送信バッファの大きさを設定します。

> **注意**: 基本的な抑止機構を提供します。 これにより、現在のプロセスがエンティティを完全に送信できるようになるでしょう。
>
> **注意**: これは、以下の SAPI では期待通りに動作しないかもしれません。 FastCGI。

パラメータ

bytes

> バイト単位で指定した、バッファの大きさ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getBufferSize()
- HttpResponse::getThrottleDelay()
- HttpResponse::setThrottleDelay()

HttpResponse::setCacheControl

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setCacheControl — cache control を設定する

説明

static bool **HttpResponse::setCacheControl** (string $control [, int $max_age = 0 [, bool $must_revalidate = TRUE]])

Cache-Control ヘッダを定義します。 通常は *private* または *public* です。

パラメータ

control

> キャッシュコントロール設定。

max_age

> max-age の秒数。これは、クライアント側での キャッシュエントリの有効期限を表します。

must_revalidate

> クライアント側でのキャッシュエントリの再検証を、 リクエストのたびに行うかどうかを指定します。

返り値

成功した場合に **TRUE**、control が *public*、 *private* あるいは *no-cache* のいずれかでない場合に **FALSE** を返します。

参考

- HttpResponse::getCacheControl()
- HttpResponse::setCache()
- HttpResponse::getCache()

HttpResponse::setCache

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setCache — キャッシュを設定する

説明

static bool **HttpResponse::setCache** (bool `$cache`)

エンティティのキャッシュを試みるかどうかを設定します。

これは、必要なキャッシュヘッダを設定したうえで、クライアントの *If-Modified-Since* および *If-None-Match* ヘッダを調べます。これらのヘッダのいずれかにマッチした場合、 ステータスコード *304 Not Modified* が発行されます。

> **注意**: セッションを使用している場合は、session.cache_limiter の設定を "no-cache" 以外の適切な値にしておきましょう!

パラメータ

`cache`

キャッシュを試みるかどうか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::setCacheControl()
- HttpResponse::getCacheControl()
- HttpResponse::getCache()

HttpResponse::setContentDisposition

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setContentDisposition — content disposition を設定する

説明

static bool **HttpResponse::setContentDisposition** (string `$filename` [, bool `$inline = FALSE`])

Content-Disposition を設定します。 *Content-Disposition* ヘッダは、 実際に送信するデータがファイルなどであり、それをクライアント/ユーザに "保存" させたい (ブラウザのポップアップ "名前を付けて保存..." を出させたい) 場合に便利です。

パラメータ

`filename`

"名前を付けて保存..." ダイアログで表示するファイル名。

`inline`

これを true に設定し、かつユーザエージェントがその content type の処理方法を知っている場合は、ポップアップウィンドウが表示されません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getContentDisposition()
- HttpResponse::getContentType()
- HttpResponse::setContentType()

HttpResponse::setContentType

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setContentType — content type を設定する

説明

static bool **HttpResponse::setContentType** (string `$content_type`)

送信されるエンティティの *Content-Type* を設定します。

パラメータ

`content_type`

送信されるエンティティの content type (primary/secondary)。

返り値

成功した場合に **TRUE** を返します。 content type にプライマリパート、セカンダリパートが含まれていない場合に **FALSE** を返します。

参考

- HttpResponse::getContentDisposition()
- HttpResponse::setContentDisposition()
- HttpResponse::getContentType()
- HttpResponse::guessContentType()

HttpResponse::setData

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setData — データを設定する

説明

static bool **HttpResponse::setData** (mixed `$data`)

送信されるデータを設定します。

注意: 以前に計算したり定義したりしていた *ETag* や *Last-Modified* は、再計算/再定義されます。

パラメータ

`data`

送信するデータ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getData()
- HttpResponse::getFile()
- HttpResponse::setFile()
- HttpResponse::getStream()
- HttpResponse::setStream()

HttpResponse::setETag

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setETag — ETag を設定する

説明

static bool **HttpResponse::setETag** (string `$etag`)

独自の *ETag* を設定します。 これが何を意味するのかをわかっている人以外は使用しないでください。

パラメータ

`etag`

ETag のパラメータに使用する、クォートされていない文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getLastModified()
- HttpResponse::setLastModified()
- HttpResponse::getETag()

HttpResponse::setFile

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setFile — ファイルを設定する

説明

static bool **HttpResponse::setFile** (string `$file`)

送信されるファイルを設定します。

注意: 以前に計算したり定義したりしていた *ETag* や *Last-Modified* は、再計算/再定義されます。

パラメータ

`file`

送信するファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getData()
- HttpResponse::setData()
- HttpResponse::getFile()
- HttpResponse::getStream()
- HttpResponse::setStream()

HttpResponse::setGzip

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setGzip — gzip を設定する

説明

static bool **HttpResponse::setGzip** (bool `$gzip`)

送信されるエンティティを、その場で gzip 圧縮する処理を有効にします。

パラメータ

gzip

GZip 圧縮を有効にするかどうか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getGzip()

HttpResponse::setHeader

(PECL pecl_http:0.12.0-1.5.5)

HttpResponse::setHeader — ヘッダを設定する

説明

static bool **HttpResponse::setHeader** (string `$name` [, mixed `$value` [, bool `$replace = TRUE`]])

HTTP ヘッダを送信します。

パラメータ

name

ヘッダの名前。

value

ヘッダの値。設定しなかった場合は、 この名前のヘッダは送信されません。

replace

既存のヘッダを置き換えるかどうか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getHeader()

HttpResponse::setLastModified

(PECL pecl_http:0.12.0-1.5.5)

HttpResponse::setLastModified — 最終更新日時を設定する

説明

static bool **HttpResponse::setLastModified** (int `$timestamp`)

独自の *Last-Modified* 日時を設定します。

パラメータ

timestamp

送信されるエンティティの最終更新日時を表す Unix タイムスタンプ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getLastModified()
- HttpResponse::getETag()
- HttpResponse::setETag()

HttpResponse::setStream

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setStream — ストリームを設定する

説明

static bool **HttpResponse::setStream** (resource `$stream`)

送信されるリソースを設定します。

> **注意**: 以前に計算したり定義したりしていた *ETag* や *Last-Modified* は、再計算/再定義されます。

パラメータ

`stream`

> 送信するデータの読み込み元となる、 既にオープンしているストリーム。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getData()
- HttpResponse::setData()
- HttpResponse::getFile()
- HttpResponse::setFile()
- HttpResponse::getStream()

HttpResponse::setThrottleDelay

(PECL pecl_http:0.10.0-1.5.5)

HttpResponse::setThrottleDelay — throttle delay を設定する

説明

static bool **HttpResponse::setThrottleDelay** (float `$seconds`)

throttle delay を設定します。

> **注意**: 基本的な抑止機構を提供します。 これにより、現在のプロセスがエンティティを完全に送信できるようになるでしょう。

> **注意**: これは、以下の SAPI では期待通りに動作しないかもしれません。 FastCGI.

パラメータ

`seconds`

> 各チャンクが送信された後で処理を待機する秒数。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- HttpResponse::getBufferSize()
- HttpResponse::setBufferSize()
- HttpResponse::getThrottleDelay()

HttpResponse::status

(No version information available, might be only in CVS)

HttpResponse::status — HTTP レスポンスステータスを送信する

説明

static bool **HttpResponse::status** (int `$status`)

この関数は次の関数のエイリアスです。 http_send_status().

---

http_cache_etag

(PECL pecl_http:0.1.0-1.5.5)

http_cache_etag — ETag でキャッシュする

説明

bool **http_cache_etag** ([string `$etag`])

送信されるエンティティを、その *ETag* を基準にしてキャッシュしようとします。 ETag は、指定されたものを使用するか、あるいは INI 設定 http.etag.mode で指定したハッシュアルゴリズムにより生成したものを使用します。

クライアントの *If-None-Match* ヘッダがこの ETag と一致した場合は、その本文はクライアント側でキャッシュされていると判断し、 ステータスコード *304 Not Modified* を発行します。

INI 設定 http.log.cache が設定されており、キャッシュの試みが成功した場合は、 キャッシュログにエントリが書き込まれます。

> **注意**: この関数は、 http_send_data() や http_send_file() そして http_send_stream() と組み合わせて使用しすることもあります。

この関数を http_send_*() API の外部で使用すると、 ob_etaghandler() を助けます。

パラメータ

`etag`

独自の *ETag*。

返り値

失敗した場合は **FALSE** を返します。成功した場合は *終了* します。 エンティティがキャッシュされた場合は、*304 Not Modified* となります。 "終了" の意味については、INI 設定 http.force_exit を参照ください。

例

Example#1 http_cache_etag() の例

```
<?php
http_cache_etag();
http_send_data("data");
?>
```

参考

- http_cache_last_modified()
- ob_etaghandler()
- http_match_etag()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

http_cache_last_modified

(PECL pecl_http:0.1.0-1.5.5)

http_cache_last_modified — 最終更新日時でキャッシュする

説明

bool **http_cache_last_modified** ([int `$timestamp_or_expires`])

送信されるエンティティを、その最終更新日時を基準にしてキャッシュしようとします。

引数に 0 より大きい値が指定された場合、それは timestamp として扱われ、最終更新日時として送信されます。0 または省略された場合は、 現在の時刻を *Last-Modified* 日時として送信します。負の数を指定した場合は、 それは有効期限を表す秒数として扱われます。つまり、 リクエストされた最終更新日時がこの範囲内にない場合は、 *Last-Modified* ヘッダが更新されて実際の本文が送信されます。

INI 設定 http.log.cache が設定されており、キャッシュの試みが成功した場合は、 キャッシュログにエントリが書き込まれます。

> **注意**: この関数は、 http_send_data() や http_send_file() そして http_send_stream() と組み合わせて使用しすることもあります。

パラメータ

`timestamp_or_expires`

Unix タイムスタンプ。

返り値

失敗した場合は **FALSE** を返します。成功した場合は *終了* します。 エンティティがキャッシュされた場合は、*304 Not Modified* となります。 "終了" の意味については、INI 設定 http.force_exit を参照ください。

例

Example#1 http_cache_last_modified() の例

5 秒間キャッシュします。

```
<?php
http_cache_last_modified(-5);
printf("%s¥n", http_date());
?>
```

参考

- http_cache_etag()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_chunked_decode

(PECL pecl_http:0.1.0-1.5.5)

http_chunked_decode — chunked-encoded データをデコードする

説明

string **http_chunked_decode** (string `$encoded`)

HTTP-chunked エンコードされた文字列をデコードします。

パラメータ

`encoded`

Chunked encode された文字列。

返り値

成功した場合はデコードした文字列、失敗した場合は **FALSE** を返します。

例

Example#1 http_chunked_decode() の例

```
<?php
$string = "".
    "05¥r¥n".
    "this ¥r¥n".
    "07¥r¥n".
    "string ¥r¥n".
    "12¥r¥n".
    "is chunked encoded¥r¥n".
    "01¥n¥r¥n".
    "00";
echo http_chunked_decode($string);
?>
```

上の例の出力は以下となります。

```
this string is chunked encoded
```

---

http_deflate

(PECL pecl_http:0.15.0-1.5.5)

http_deflate — データを圧縮する

説明

string **http_deflate** (string `$data` [, int `$flags = 0`])

gzip、 *zlib* 別名 *deflate* あるいは *raw deflate* エンコーディングを使用してデータを圧縮します。

flags に指定できる値については、 圧縮定数の表 を参照ください。

パラメータ

`data`

エンコードされるデータを含む文字列。

`flags`

圧縮オプション。

返り値

成功した場合はエンコードした文字列、失敗した場合は NULL を返します。

参考

- http_inflate()
- HttpDeflateStream

---

http_inflate

(PECL pecl_http:0.15.0-1.5.5)

http_inflate — データを展開する

説明

string **http_inflate** (string `$data`)

gzip、*deflate* 別名 *zlib* あるいは *raw deflate* エンコーディングで圧縮されたデータを展開します。

パラメータ

data

圧縮されたデータを含む文字列。

返り値

成功した場合はデコードした文字列、失敗した場合は NULL を返します。

参考

- http_deflate()
- HttpInflateStream
-

---

http_get_request_body_stream

(PECL pecl_http:0.22.0-1.5.5)

http_get_request_body_stream — リクエストの本文をストリームとして取得する

説明

resource **http_get_request_body_stream** (void)

ストリームを作成し、生のリクエスト本文 (例: POST あるいは PUT データ) を読み込めるようにします。

リクエストメソッドが POST 以外の場合は、この関数は一度しか使用できません。

パラメータ

返り値

成功した場合は生のリクエスト本文をストリームで返します。 失敗した場合は NULL を返します。

参考

- http_get_request_body()
- http_get_request_headers()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_get_request_body

(PECL pecl_http:0.10.0-1.5.5)

http_get_request_body — リクエストの本文を文字列として取得する

説明

string **http_get_request_body** (void)

生のリクエスト本文 (例: POST あるいは PUT データ) を取得します。

リクエストメソッドが POST 以外の場合は、 http_get_request_body_stream() の後にこの関数を使用することはできません。

パラメータ

返り値

成功した場合は生のリクエスト本文を文字列で返します。 失敗した場合は NULL を返します。

参考

- http_get_request_body_stream()
- http_get_request_headers()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_get_request_headers

(PECL pecl_http:0.10.0-1.5.5)

http_get_request_headers — リクエストヘッダを配列として取得する

説明

array **http_get_request_headers** (void)

受け取った HTTP ヘッダの一覧を取得します。

パラメータ

返り値

受け取ったリクエストヘッダを連想配列で返します。

参考

- http_get_request_body()
- http_get_request_body_stream()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_date

(PECL pecl_http:0.1.0-1.5.5)

http_date — HTTP の RFC に準拠した日付を作成する

説明

string **http_date** ([int `$timestamp`])

RFC 1123 に準拠した、妥当な形式の HTTP 日付を作成します。 *Wed, 22 Dec 2004 11:34:47 GMT* のような形式になります。

パラメータ

`timestamp`

Unix タイムスタンプ。省略した場合は現在時刻。

返り値

HTTP の日付を文字列で返します。

参考

- date()

---

http_support

(PECL pecl_http:0.15.0-1.5.5)

http_support — 組み込みの HTTP サポートを調べる

説明

int **http_support** ([int `$feature = 0`])

外部ライブラリを必要とする機能について調べます。

引数 *`feature`* に指定可能な値については サポートする機能の定数の表 を参照ください。

パラメータ

feature

調査する機能。

返り値

指定された機能がサポートされているかどうかを表す integer 値、 あるいは *feature* を省略した場合はサポートされる全機能のビットマスクを返します。

例

Example#1 http_support() の例

```
<?php
if (!http_support(HTTP_SUPPORT_REQUESTS)) {
    die("Need HTTP request support!¥n");
}
?>
```

---

http_match_etag

(PECL pecl_http:0.1.0-1.5.5)

http_match_etag — ETag を比較する

説明

bool **http_match_etag** (string `$etag` [, bool `$for_range = FALSE`])

指定された *ETag* を、クライアントの HTTP ヘッダ *If-Match* あるいは *If-None-Match* と比較します。

パラメータ

etag

比較する *ETag*。

for_range

TRUE に設定すると、通常は HTTP の範囲を検証するために使用されるヘッダが チェックされます。

返り値

ETag が一致する場合やヘッダにアスタリスク ("*") が含まれる場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- **http_match_last_modified()**
- http_match_request_header()
- http_cache_etag()
- http_cache_last_modified()
- ob_etaghandler()

---

http_match_modified

(PECL pecl_http:0.1.0-1.5.5)

http_match_modified — 最終更新日時を比較する

説明

bool **http_match_modified** ([int `$timestamp` [, bool `$for_range = FALSE`]])

指定された Unix タイムスタンプを、クライアントの HTTP ヘッダ *If-Modified-Since* あるいは *If-Unmodified-Since* と比較します。

パラメータ

timestamp

Unix タイムスタンプ。省略した場合は現在の時刻が使用されます。

for_range

TRUE に設定すると、通常は HTTP の範囲を検証するために使用されるヘッダが チェックされます。

返り値

timestamp がヘッダより前の時刻を表す場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

参考

- http_match_etag()
- http_match_request_header()
- http_cache_etag()
- http_cache_last_modified()

---

http_match_request_header

(PECL pecl_http:0.10.0-1.5.5)

http_match_request_header — 任意のヘッダを比較する

説明

bool **http_match_request_header** (string `$header` , string `$value` [, bool `$match_case = FALSE`])

受け取った HTTP ヘッダを比較します。

パラメータ

header

ヘッダの名前 (大文字小文字は区別しません)。

value

ヘッダの値と比較する値。

match_case

値を比較する際に大文字小文字を区別するかどうか。

返り値

ヘッダの値が一致する場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- http_match_etag()
- **http_match_last_modified()**

---

http_build_cookie

(PECL pecl_http:1.2.0-1.5.5)

http_build_cookie — クッキー文字列を作成する

説明

string **http_build_cookie** (array `$cookie`)

http_parse_cookie() が返すような形式の配列/オブジェクトから、クッキー文字列を作成します。

パラメータ

cookie

http_parse_cookie() が返すような形式の、クッキーのリスト。

返り値

クッキーを文字列で返します。

参考

- http_parse_cookie()

---

http_negotiate_charset

(PECL pecl_http:0.1.0-1.5.5)

http_negotiate_charset — クライアントが希望している文字セットを選択する

説明

string **http_negotiate_charset** (array `$supported` [, array `&$result`])

この関数は、クライアントが希望している文字セットを *Accept-Charset* HTTP ヘッダに基づいて選択します。 qualifier も理解し、qualifier の指定されていない文字セットが最優先されます。

パラメータ

`supported`

サポートされる文字セットを値にもつ配列。

`result`

選択した結果がこの配列に含まれます。

返り値

選択された文字セット、あるいは一致するものがなかった場合はデフォルトの文字セット (配列の最初のエントリ) を返します。

例

Example#1 http_negotiate_charset() の使用法

```
<?php
$charsets  =  array(
        'iso-8859-1', //  デフォルト
        'iso-8859-2',
        'iso-8859-15',
        'utf-8'
);

$pref  =  http_negotiate_charset($charsets, $result);

if (strcmp($pref, 'iso-8859-1')) {
        iconv_set_encoding('internal_encoding', 'iso-8859-1');
        iconv_set_encoding('output_encoding', $pref);
        ob_start('ob_iconv_handler');
}

print_r($result);
?>
```

---

http_negotiate_content_type

(PECL pecl_http:0.19.0-1.5.5)

http_negotiate_content_type — クライアントが希望している content type を選択する

説明

string **http_negotiate_content_type** (array `$supported` [, array `&$result`])

この関数は、クライアントが希望している content type を *Accept* HTTP ヘッダに基づいて選択します。 qualifier も理解し、qualifier の指定されていない content type が最優先されます。

パラメータ

supported

サポートする content type を値にもつ配列。

result

選択した結果がこの配列に含まれます。

返り値

選択された content type、あるいは一致するものがなかった場合はデフォルトの content type (配列の最初のエントリ) を返します。

例

Example#1 http_negotiate_content_type() の使用例

```
<?php
$content_types  =  array('application/xhtml+xml', 'text/html');
http_send_content_type(http_negotiate_content_type($content_types));
?>
```

---

http_negotiate_language

(PECL pecl_http:0.1.0-1.5.5)

http_negotiate_language — クライアントが希望している言語を選択する

説明

string **http_negotiate_language** (array `$supported` [, array `&$result`])

この関数は、クライアントが希望している言語を *Accept-Language* HTTP ヘッダに基づいて選択します。 qualifier も理解し、qualifier の指定されていない言語が最優先されます。 部分的に一致した (つまりプライマリ言語についてのみ一致した) 言語については、qualifier が 10% 割り引かれます。

パラメータ

supported

サポートされる言語を値にもつ配列。

result

選択した結果がこの配列に含まれます。

返り値

選択された言語、あるいは一致するものがなかった場合はデフォルトの言語 (配列の最初のエントリ) を返します。

例

Example#1 http_negotiate_language() の使用法

```
<?php
$langs  =  array(
         'en-US',//  デフォルト
         'fr',
         'fr-FR',
         'de',
         'de-DE',
         'de-AT',
         'de-CH',
);

include  './langs/'. http_negotiate_language($langs, $result) .'.php';

print_r($result);
?>
```

---

ob_deflatehandler

(PECL pecl_http:0.21.0-1.5.5)

ob_deflatehandler — 圧縮出力ハンドラ

説明

string **ob_deflatehandler** (string `$data` , int `$mode`)

ob_start() とともに使用します。

> **注意**: この出力ハンドラは、一度だけ使用できます。

圧縮出力バッファハンドラは、一度だけ使用できます。

これは ob_gzhandler() や zlib.output_compression と衝突します。また、 mbstring 拡張モジュールの mb_output_handler() や session 拡張モジュールの URL-Rewriter (あるいは session.use_trans_sid) の後で使用してはいけません。

参考

- ob_inflatehandler()
- ob_start()

---

ob_etaghandler

(PECL pecl_http:0.10.0-1.5.5)

ob_etaghandler — ETag 出力ハンドラ

説明

string **ob_etaghandler** (string `$data` , int `$mode`)

ob_start() とともに使用します。

この出力バッファハンドラは、 INI 設定 http.etag.mode で指定したハッシュアルゴリズムを使用して、*ETag* を生成します。

この出力ハンドラは、 http_cache_etag() によって使用されます。

参考

- http_cache_etag()
- http_match_etag()

---

ob_inflatehandler

(PECL pecl_http:0.21.0-1.5.5)

ob_inflatehandler — 展開出力ハンドラ

説明

string **ob_inflatehandler** (string `$data` , int `$mode`)

ob_start() とともに使用します。

ob_deflatehandler() と同様の制限が当てはまります。

参考

- ob_deflatehandler()
- ob_start()

---

http_parse_cookie

(PECL pecl_http:0.20.0-1.5.5)

http_parse_cookie — HTTP クッキーをパースする

説明

object **http_parse_cookie** (string `$cookie` [, int `$flags` [, array `$allowed_extras`]])

レスポンスとして送信されるのと同様の形式に HTTP クッキーをパースし、構造体に格納します。

パラメータ

`cookie`

レスポンスヘッダ *Set-Cookie* の値を含む文字列。

`flags`

パースフラグ (**HTTP_COOKIE_PARSE_RAW**)。

`allowed_extras`

特別な キーとして理解される内容を含む配列。 デフォルトでは、未知のキーはすべてクッキー名として扱われます。

返り値

成功した場合は stdClass オブジェクト、 失敗した場合は **FALSE** を返します。

例

Example#1 http_parse_cookie() の使用法

```
<?php
print_r(http_parse_cookie("foo=bar; bar=baz; path=/; domain=example.com; comment=; secure", 0, array("comment")));
?>
```

上の例の出力は以下となります。

```
stdClass Object
(
   [cookies] => Array
   (
      [foo] => bar
      [bar] => baz
   )

   [extras] => Array
   (
      [comment] =>
   )

   [flags] => 16
   [expires] => 0
   [path] => /
   [domain] => example.com
)
```

参考

- http_parse_headers()
- http_parse_message()
- http_build_cookie()

---

http_parse_headers

(PECL pecl_http:0.10.0-1.5.5)

http_parse_headers — HTTP ヘッダをパースする

説明

array **http_parse_headers** (string `$header`)

HTTP ヘッダをパースし、連想配列に格納します。

パラメータ

header

HTTP ヘッダを含む文字列。

返り値

成功した場合に配列、失敗した場合に **FALSE** を返します。

例

Example#1 http_parse_headers() の使用法

```
<?php
$headers = "content-type: text/html; charset=UTF-8¥r¥n".
  "Server: Funky/1.0¥r¥n".
  "Set-Cookie: foo=bar¥r¥n".
  "Set-Cookie: baz=quux¥r¥n".
  "Folded: works¥r¥n¥ttoo¥r¥n";
print_r(http_parse_headers($headers));
?>
```

上の例の出力は以下となります。

```
Array
(
   [Content-Type] => text/html; chatset=UTF-8
   [Server] => Funky/1.0
   [Set-Cookie] => Array
  (
     [0] => foo=bar
     [1] => baz=quux
  )
   [Folded] => works
     too
)
```

参考

- http_parse_message()
- http_parse_cookie()

---

http_parse_message

(PECL pecl_http:0.12.0-1.5.5)

http_parse_message — HTTP メッセージをパースする

説明

object **http_parse_message** (string $message)

HTTP の *message* をパースし、 単純な形式の再帰的な object に格納します。

パラメータ

message

単一の HTTP メッセージ、あるいは複数の連続した HTTP メッセージを含む文字列。

返り値

パースされたメッセージを、階層化されたオブジェクトとして返します。

例

Example#1 http_parse_message() の使用法

```
<?php
define ('URL', 'http://www.example.com/');
print_r(http_parse_message(http_get(URL, array('redirect' => 3))));
?>
```

上の例の出力は、たとえば以下のようになります。

```
stdClass object
```

```
(
   [type] => 2
   [httpVersion] => 1.1
   [responseCode] => 200
   [headers] => Array
   (
      [Content-Length] => 3
      [Server] => Apache
   )
   [body] => Hi!
   [parentMessage] => stdClass object
   (
      [type] => 2
      [httpVersion] => 1.1
      [responseCode] => 302
      [headers] => Array
      (
         [Content-Length] => 0
         [Location] => ...
      )
      [body] =>
      [parentMessage] => ...
   )
)
```

参考

- http_parse_headers()
- HttpMessage クラス

---

http_parse_params

(PECL pecl_http:1.0.0-1.5.5)

http_parse_params — パラメータリストをパースする

説明

object **http_parse_params** (string $param [, int $flags = HTTP_PARAMS_DEFAULT])

パラメータリストをパースします。

引数 *flags* に指定可能な値については パラメータ解析の定数の表 を参照ください。

パラメータ

param

パラメータ。

flags

パースフラグ。

返り値

パラメータリストを stdClass オブジェクトで返します。

例

Example#1 http_parse_params() の例

```
<?php
var_dump(http_parse_params("text/html;  charset=¥"utf8¥""));
?>
```

上の例の出力は以下となります。

```
object(stdClass)#1 (1) {
   ["params"]=>
   array(2) {
      [0]=>
      string(9) "text/html"
      [1]=>
      array(1) {
         ["charset"]=>
         string(4) "utf8"
      }
   }
```

```
}
```

参考

- http_parse_headers()
- http_parse_cookie()
- http_parse_message()

---

http_persistent_handles_count

(PECL pecl_http:1.5.0-1.5.5)

http_persistent_handles_count — 持続ハンドルの状況

説明

object **http_persistent_handles_count** (void)

持続ハンドルの使用状況についての一覧を取得します。

パラメータ

返り値

成功した場合には持続ハンドルの状況を表す stdClass オブジェクト、 失敗した場合に **FALSE** を返します。

例

Example#1 http_persistent_handles_count() の例

```
<?php
print_r(http_persistent_handles_count());
?>
```

上の例の出力は以下となります。

```
stdClass Object
(
    [http_request] => Array
        (
            [GLOBAL] => Array
                (
                    [used] => 0
                    [free] => 1
                )

        )

    [http_request_datashare] => Array
        (
            [GLOBAL] => Array
                (
                    [used] => 1
                    [free] => 0
                )

        )

    [http_request_pool] => Array
        (
        )

)
```

参考

- http_persistent_handles_ident()
- http_persistent_handles_clean()

---

http_persistent_handles_ident

(PECL pecl_http:1.5.0-1.5.5)

http_persistent_handles_ident — 持続ハンドルの ident を取得/設定する

説明

string **http_persistent_handles_ident** (string $ident)

持続ハンドルの ident を取得あるいは定義します。

パラメータ

ident

識別文字列。

返り値

成功した場合に以前の ident、 失敗した場合に **FALSE** を返します。

例

Example#1 http_persistent_handles_ident() の例

```
<?php
echo http_persistent_handles_ident("CUSTOM"), "¥n";
echo http_persistent_handles_ident("MyApp1"), "¥n";
http_get("http://www.example.com/");
print_r(http_persistent_handles_count());
?>
```

上の例の出力は以下となります。

```
GLOBAL
CUSTOM
stdClass Object
(
    [http_request] => Array
        (
            [MyApp1] => Array
                (
                    [used] => 0
                    [free] => 1
                )

        )

    [http_request_datashare] => Array
        (
            [GLOBAL] => Array
                (
                    [used] => 1
                    [free] => 0
                )

        )

    [http_request_pool] => Array
        (
        )

)
```

参考

- http_persistent_handles_count()
- http_persistent_handles_clean()

---

http_persistent_handles_clean

(PECL pecl_http:1.5.0-1.5.5)

http_persistent_handles_clean — 持続ハンドルを消去する

説明

string **http_persistent_handles_clean** ([string $ident])

持続ハンドルを消去します (閉じます)。オプションで、ident による認証を行います。

パラメータ

clean

認証文字列。

返り値

値を返しません。

参考

- http_persistent_handles_ident()
- http_persistent_handles_count()

---

http_get

(PECL pecl_http:0.1.0-1.5.5)

http_get — GET リクエストを実行する

説明

string **http_get** (string $url [, array $options [, array &$info]])

指定した url に対して HTTP GET リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

url

URL。

options

リクエストのオプション

info

リクエスト/レスポンスの情報が格納されます。

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

例

Example#1 http_get() の例

```
<?php
$response = http_get("http://www.example.com/", array("timeout"=>1), $info);
print_r($info);
?>
```

上の例の出力は以下となります。

```
array (
   'effective_url' => 'http://www.example.com/',
   'response_code' => 302,
   'connect_code' => 0,
   'filetime' => -1,
   'total_time' => 0.212348,
   'namelookup_time' => 0.038296,
   'connect_time' => 0.104144,
   'pretransfer_time' => 0.104307,
   'starttransfer_time' => 0.212077,
   'redirect_time' => 0,
   'redirect_count' => 0,
   'size_upload' => 0,
   'size_download' => 218,
   'speed_download' => 1026,
   'speed_upload' => 0,
   'header_size' => 307,
   'request_size' => 103,
   'ssl_verifyresult' => 0,
```

```
   'ssl_engines' =>
  array (
     0 => 'dynamic',
     1 => 'cswift',
     2 => 'chil',
     3 => 'atalla',
     4 => 'nuron',
     5 => 'ubsec',
     6 => 'aep',
     7 => 'sureware',
     8 => '4758cca',
  ),
   'content_length_download' => 218,
   'content_length_upload' => 0,
   'content_type' => 'text/html',
   'httpauth_avail' => 0,
   'proxyauth_avail' => 0,
   'num_connects' => 1,
   'os_errno' => 0,
   'error' => '',
)
```

---

http_head

(PECL pecl_http:0.1.0-1.5.5)

http_head — HEAD リクエストを実行する

説明

string **http_head** ([string $url [, array $options [, array &$info]]])

指定した url に対して HTTP HEAD リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

url

URL。

options

リクエストのオプション

info

リクエスト/レスポンス の情報

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

---

http_post_data

(PECL pecl_http:0.1.0-1.5.5)

http_post_data — エンコードされたデータを使用して POST リクエストを実行する

説明

string **http_post_data** (string $url [, string $data [, array $options [, array &$info]]])

指定した url に対して HTTP POST リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

url

URL。

data

エンコードされた POST データを含む文字列。

options

リクエストのオプション

info

リクエスト/レスポンス の情報

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

---

http_post_fields

(PECL pecl_http:0.10.0-1.5.5)

http_post_fields — エンコードされる前のデータを使用して POST リクエストを実行する

説明

string **http_post_fields** (string `$url` [, array `$data` [, array `$files` [, array `$options` [, array `&$info`]]]])

指定した url に対して HTTP POST リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

url

URL。

data

POST する値の連想配列。

files

POST するファイルの配列。

options

リクエストのオプション

info

リクエスト/レスポンス の情報

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

例

Example#1 http_post_fields() の例

```
<?php
$fields = array(
    'name' => 'mike',
    'pass' => 'se_ret'
);
$files = array(
     array(
         'name' => 'uimg',
         'type' => 'image/jpeg',
         'file' => './profile.jpg',
    )
);

$response = http_post_fields("http://www.example.com/", $fields, $files);
?>
```

---

http_put_data

(PECL pecl_http:0.25.0-1.5.5)

http_put_data — データを使用して PUT リクエストを実行する

説明

string **http_put_data** (string `$url` [, string `$data` [, array `$options` [, array `&$info`]]])

指定した url に対して HTTP PUT リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

`url`

URL。

`data`

PUT リクエストの本文。

`options`

リクエストのオプション

`info`

リクエスト/レスポンス の情報

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

---

http_put_file

(PECL pecl_http:0.10.0-1.5.5)

http_put_file — ファイルを使用して PUT リクエストを実行する

説明

string **http_put_file** (string `$url` [, string `$file` [, array `$options` [, array `&$info`]]])

指定した url に対して HTTP PUT リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

`url`

URL。

`file`

PUT するファイル。

`options`

リクエストのオプション

`info`

リクエスト/レスポンス の情報

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

---

http_put_stream

(PECL pecl_http:0.10.0-1.5.5)

http_put_stream — ストリームを使用して PUT リクエストを実行する

説明

string **http_put_stream** (string `$url` [, resource `$stream` [, array `$options` [, array `&$info`]]])

指定した url に対して HTTP PUT リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

`url`

URL。

`stream`

PUT リクエストの本文の読み込み元となるストリーム。

`options`

リクエストのオプション

`info`

リクエスト/レスポンス の情報

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

---

http_request_method_exists

(PECL pecl_http:0.10.0-1.5.5)

http_request_method_exists — リクエストメソッドが存在するかどうかを調べる

説明

int **http_request_method_exists** (mixed `$method`)

リクエストメソッドが登録されているか (あるいはデフォルトで使用可能か) どうかを調べます。

パラメータ

`method`

リクエストメソッドの名前あるいは ID。

返り値

リクエストメソッドが使用可能な場合に **`TRUE`**、 それ以外の場合に **`FALSE`** を返します。

---

http_request_method_name

(PECL pecl_http:0.10.0-1.5.5)

http_request_method_name — リクエストメソッド名を取得する

説明

string **http_request_method_name** (int $method)

標準的な、あるいは登録されたリクエストメソッドの名前を リテラル文字列として取得します。

パラメータ

method

リクエストメソッドの ID。

返り値

成功した場合にリクエストメソッド名を表す文字列、 失敗した場合に **FALSE** を返します。

---

http_request_method_register

(PECL pecl_http:0.10.0-1.5.5)

http_request_method_register — リクエストメソッドを登録する

説明

int **http_request_method_register** (string $method)

独自のリクエストメソッドを登録します。

パラメータ

method

登録するリクエストメソッドの名前。

返り値

成功した場合にリクエストメソッドの ID、 失敗した場合に FALSE を返します。

---

http_request_method_unregister

(PECL pecl_http:0.10.0-1.5.5)

http_request_method_unregister — リクエストメソッドの登録を解除する

説明

bool **http_request_method_unregister** (mixed $method)

事前に登録された、独自のリクエストメソッドの登録を解除します。

パラメータ

method

リクエストメソッドの名前あるいは ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

http_request

(PECL pecl_http:1.0.0-1.5.5)

http_request — 独自のリクエストを実行する

説明

string **http_request** (int `$method` [, string `$url` [, string `$body` [, array `$options` [, array `&$info`]]]])

指定した url に対して独自の HTTP リクエストを実行します。

リクエストのオプション を参照ください。

パラメータ

`method`

リクエストメソッド。

`url`

URL。

`body`

リクエストの本文。

`options`

リクエストのオプション

`info`

リクエスト/レスポンス の情報

返り値

成功した場合は HTTP レスポンスを文字列で、失敗した場合は FALSE を返します。

---

http_request_body_encode

(PECL pecl_http:1.0.0-1.5.5)

http_request_body_encode — リクエスト本文をエンコードする

説明

string **http_request_body_encode** (array `$fields` , array `$files`)

x-www-form-urlencoded あるいは form-data でエンコードされたリクエスト本文を作成します。

パラメータ

`fields`

POST フィールド。

`files`

POST ファイル。

返り値

成功した場合にエンコードされた文字列、失敗した場合に FALSE を返します。

---

http_redirect

(PECL pecl_http:0.1.0-1.5.5)

http_redirect — HTTP リダイレクトを発行する

説明

void **http_redirect** ([string $url [, array $params [, bool $session = FALSE [, int $status]]]])

指定した url にリダイレクトします。

指定した *url* は http_build_url() で展開され、 *params* の内容は http_build_str() で扱われます。 また、*session* が true の場合はセッション ID が付加されます。 HTTP レスポンスコードは、*status* に応じたものが設定されます。 リダイレクト定数 のいずれかを使用すると便利です。 どのような場面でどのようなレスポンスコードが返されるのかについては、 » RFC 2616 を参照ください。デフォルトでは、いちばんうまく当てはまるステータスを PHP が決定します。

RFC に準拠するため、クライアントがすぐにリダイレクトしない場合でも "Redirecting to <a>URL." と表示されます。 また、リクエストメソッドは HEAD 以外のものとなります。

INI 設定 http.log.redirect が設定されており、リダイレクトの試みが成功した場合には リダイレクトのログにエントリが書き込まれます。

パラメータ

url

リダイレクトする URL。

params

クエリパラメータの連想配列。

session

セッション情報を付加するかどうか。

status

独自のレスポンスステータスコード。

返り値

失敗した場合は **FALSE** を返します。成功した場合は 終了 します。 指定したステータスコードとなります。 "終了" の意味については、INI 設定 http.force_exit を参照ください。

例

Example#1 http_redirect() の例

```
<?php
http_redirect("relpath", array("name" => "value"), true, HTTP_REDIRECT_PERM);
?>
```

上の例の出力は以下となります。

```
HTTP/1.1 301 Moved Permanently
X-Powered-By: PHP/5.2.2
Content-Type: text/html
Location: http://www.example.com/curdir/relpath?name=value&PHPSESSID=abc

<a href="http://www.example.com/curdir/relpath?name=value&PHPSESSID=abc">http://www.example.com/curdir/relpath?name=value
```

にリダイレクトされます。

参考

- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_send_content_disposition

(PECL pecl_http:0.10.0-1.5.5)

http_send_content_disposition — Content-Disposition を送信する

説明

bool **http_send_content_disposition** (string $filename [, bool $inline = FALSE])

Content-Disposition を送信します。 *Content-Disposition* ヘッダは、 実際に送信するデータがファイルなどであり、それをクライアント/ユーザに "保存" させたい (ブラウザのポップアップ "名前を付けて保存..." を出させたい) 場合に便利です。

> **注意**: この関数は、 http_send_data() や http_send_file() そして http_send_stream() と組み合わせて使用することを想定しています。

パラメータ

filename

> "名前を付けて保存..." ダイアログで表示するファイル名。

inline

> これを **TRUE** に設定し、かつユーザエージェントがその content type の処理方法を知っている場合は、ポップアップウィンドウが表示されません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- http_send_data()
- http_send_file()
- http_send_stream()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_send_content_type

(PECL pecl_http:0.10.0-1.5.5)

http_send_content_type — Content-Type を送信する

説明

bool **http_send_content_type** ([string `$content_type = 'application/x-octetstream'`])

エンティティの *Content-Type* を送信します。

> **注意**: この関数は、 http_send_data() や http_send_file() そして http_send_stream() と組み合わせて使用することを想定しています。

パラメータ

content_type

> content type (primary/secondary)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

content_type がプライマリパートおよびセカンダリパートを含んでいない場合に **E_WARNING** を発生させます。

参考

- http_send_data()
- http_send_file()
- http_send_stream()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_send_data

(PECL pecl_http:0.1.0-1.5.5)

http_send_data — 任意のデータを送信する

説明

bool **http_send_data** (string $data)

(複数の) range リクエストをサポートする生のデータを送信します。

パラメータ

data

送信するデータ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- http_send_file()
- http_send_stream()
- http_throttle()
- http_send_content_type()
- http_send_content_disposition()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_send_file

(PECL pecl_http:0.1.0-1.5.5)

http_send_file — ファイルを送信する

説明

bool **http_send_file** (string $file)

(複数の) range リクエストをサポートするファイルを送信します。

この関数の振る舞いおよびその後の動作は、以下の INI 設定 http.send.not_found_404 そして http.log.not_found に依存します。

INI 設定 http.send.not_found_404 が有効で INI 設定 http.log.not_found が書き込み可能なファイルを指している場合は、*file* が見つからないときにそこにログメッセージが書き込まれます。

パラメータ

file

送信するファイル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 http_send_file() の例

```
<?php
http_send_content_disposition("document.pdf", true);
http_send_content_type("application/pdf");
http_throttle(0.1, 2048);
http_send_file("../report.pdf");
?>
```

上の例の出力は以下となります。

```
HTTP/1.1 206 Partial Content
X-Powered-By: PHP/5.2.2
Accept-Ranges: bytes
```

```
Content-Length:  12345
Content-Range:  bytes  0-12344
Content-Type:  application/pdf
Content-Disposition:  inline;  filename="document.pdf"

%PDF...
```

参考

- http_send_data()
- http_send_stream()
- http_throttle()
- http_send_content_type()
- http_send_content_disposition()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_send_last_modified

(PECL pecl_http:0.1.0-1.5.5)

http_send_last_modified — Last-Modified を送信する

説明

bool **http_send_last_modified** ([int $timestamp])

Last-Modified ヘッダに、有効な HTTP 日付を設定して送信します。

> **注意**: この関数は、 http_send_data() や http_send_file() そして http_send_stream() と組み合わせて使用することを想定しています。

パラメータ

timestamp

> Unix タイムスタンプ。これは、有効な HTTP 日付に変換されます。 省略した場合は、現在の時刻が送信されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_send_status

(PECL pecl_http:0.1.0-1.5.5)

http_send_status — HTTP レスポンスステータスを送信する

説明

bool **http_send_status** (int $status)

HTTP ステータスコードを送信します。

パラメータ

status

> HTTP ステータスコード (100-599)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_send_stream

(PECL pecl_http:0.1.0-1.5.5)

http_send_stream — ストリームを送信する

説明

bool **http_send_stream** (resource `$stream`)

(複数の) range リクエストをサポートする、 オープン済みのストリームを送信します。

パラメータ

`stream`

読み込み元となるストリーム (シーク可能でなければなりません)。

返り値

成功した場合に **`TRUE`** を、失敗した場合に **`FALSE`** を返します。

参考

- http_send_data()
- http_send_file()
- http_throttle()
- http_send_content_type()
- http_send_content_disposition()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_throttle

(PECL pecl_http:0.10.0-1.5.5)

http_throttle — HTTP 抑止処理

説明

void **http_throttle** ([float `$sec` [, int `$bytes = 40960`]])

throttle delay および送信バッファサイズを指定します。

注意: この関数は、 http_send_data() や http_send_file() そして http_send_stream() と組み合わせて使用することを想定しています。

注意: 基本的な抑止機構を提供します。 これにより、現在のプロセスがエンティティを完全に送信できるようになるでしょう。

注意: これは、以下の SAPI では期待通りに動作しないかもしれません。 FastCGI.

パラメータ

`sec`

各チャンクが送信された後に待機する秒数。

`bytes`

チャンクの大きさ (バイト単位)。

例

Example#1 http_throttle() の例

約 20 kbyte/秒 でファイルを送信します。

```
<?php
// ~ 20 kbyte/s
#  http_throttle(1,  20000);
#  http_throttle(0.5,  10000);
http_throttle(0.1, 2000);
http_send_file('document.pdf');
?>
?>
```

参考

- http_send_data()
- http_send_file()
- http_send_stream()
- PHP 5.1 以降を使用している場合は HttpResponse クラス

---

http_build_str

(PECL pecl_http:0.23.0-1.5.5)

http_build_str — クエリ文字列を組み立てる

説明

string **http_build_str** (array `$query` [, string `$prefix` [, string `$arg_separator`]])

parse_str() の逆の動作をします。

パラメータ

`query`

クエリ文字列パラメータの連想配列。

`prefix`

トップレベルのプレフィックス。

`arg_separator`

引数の区切りとして使用する文字 (デフォルトでは、INI 設定 arg_separator.output が使用されます。 これも設定されていない場合は "&" が使用されます)。

返り値

成功した場合に組み立てたクエリ文字列、失敗した場合に **`FALSE`** を返します。

参考

- standard http_build_query()
- http_build_url()

---

http_build_url

(PECL pecl_http:0.21.0-1.5.5)

http_build_url — URL を組み立てる

説明

string **http_build_url** ([mixed `$url` [, mixed `$parts` [, int `$flags = HTTP_URL_REPLACE` [, array `&$new_url`]]]])

URL を組み立てます。

引数 flag の設定によっては、二番目の URL の一部が最初の URL に統合されます。

パラメータ

url

文字列形式、あるいは parse_url() の返すような連想配列形式で表した、URL (の一部)。

parts

最初の引数と同じ。

flags

HTTP_URL 定数 の論理和からなるビットマスク。デフォルトは **HTTP_URL_REPLACE** です。

new_url

設定されている場合は、parse_url() が返すような構築された URL の一部が格納されます。

返り値

成功した場合に新しい URL を表す文字列、失敗した場合に **FALSE** を返します。

例

Example#1 http_build_url() の例

```
<?php
echo http_build_url("http://user@www.example.com/pub/index.php?a=b#files",
     array(
         "scheme"  => "ftp",
         "host"  => "ftp.example.com",
         "path"  => "files/current/",
         "query"  => "a=c"
    ),
    HTTP_URL_STRIP_AUTH  | HTTP_URL_JOIN_PATH  | HTTP_URL_JOIN_QUERY  | HTTP_URL_STRIP_FRAGMENT
);
?>
```

上の例の出力は以下となります。

```
ftp://ftp.example.com/pub/files/current/?a=b&a=c
```

参考

- parse_url()
- http_build_str()

目次

- HttpMessage::getHeader — ヘッダを取得する
- HttpMessage::addHeaders — ヘッダを追加する
- HttpMessage::setHeaders — ヘッダを設定する
- HttpMessage::getBody — メッセージの本文を取得する
- HttpMessage::setBody — メッセージの本文を設定する
- HttpMessage::getRequestMethod — リクエストメソッドを取得する
- HttpMessage::setRequestMethod — リクエストメソッドを設定する
- HttpMessage::getRequestUrl — リクエスト URL を取得する
- HttpMessage::setRequestUrl — リクエスト URL を設定する
- HttpMessage::getResponseCode — レスポンスコードを取得する
- HttpMessage::setResponseCode — レスポンスコードを設定する
- HttpMessage::getResponseStatus — レスポンスのステータスを取得する
- HttpMessage::setResponseStatus — レスポンスのステータスを設定する
- HttpQueryString — HTTP クエリ文字列クラス
- HttpQueryString::__construct — HttpQueryString のコンストラクタ
- HttpQueryString::singleton — HttpQueryString のシングルトン
- HttpQueryString::get — クエリ文字列 (の一部) を取得する
- HttpQueryString::mod — クエリ文字列の複製を変更する
- HttpQueryString::set — クエリ文字列パラメータを設定する
- HttpQueryString::toArray — クエリ文字列を配列で取得する
- HttpQueryString::toString — クエリ文字列を取得する
- HttpQueryString::xlate — クエリ文字列の文字セットを変更する
- HttpDeflateStream — HTTP 圧縮ストリームクラス
- HttpDeflateStream::factory — HttpDeflateStream クラスのファクトリ
- HttpDeflateStream::__construct — HttpDeflateStream クラスのコンストラクタ
- HttpDeflateStream::update — 圧縮ストリームを更新する
- HttpDeflateStream::flush — 圧縮ストリームをフラッシュする
- HttpDeflateStream::finish — 圧縮ストリームを終了する
- HttpInflateStream — HTTP 展開ストリーム
- HttpInflateStream::factory — HttpInflateStream クラスのファクトリ
- HttpInflateStream::__construct — HttpInflateStream クラスのコンストラクタ
- HttpInflateStream::update — 展開ストリームを更新する
- HttpInflateStream::flush — 展開ストリームをフラッシュする
- HttpInflateStream::finish — 展開ストリームを終了する
- HttpRequest — HTTP リクエストクラス
- HttpRequest::addCookies — クッキーを追加する
- HttpRequest::addHeaders — ヘッダを追加する
- HttpRequest::addPostFields — POST フィールドを追加する
- HttpRequest::addPostFile — POST ファイルを追加する
- HttpRequest::addPutData — PUT データを追加する
- HttpRequest::addQueryData — クエリデータを追加する
- HttpRequest::addRawPostData — 生の POST データを追加する
- HttpRequest::addSslOptions — SSL オプションを追加する
- HttpRequest::clearHistory — 履歴を消去する
- HttpRequest::__construct — HttpRequest のコンストラクタ
- HttpRequest::enableCookies — クッキーを有効にする
- HttpRequest::getContentType — content type を取得する
- HttpRequest::getCookies — クッキーを取得する
- HttpRequest::getHeaders — ヘッダを取得する
- HttpRequest::getHistory — 履歴を取得する
- HttpRequest::getMethod — メソッドを取得する
- HttpRequest::getOptions — オプションを取得する
- HttpRequest::getPostFields — POST フィールドを取得する
- HttpRequest::getPostFiles — POST ファイルを取得する
- HttpRequest::getPutData — PUT データを取得する
- HttpRequest::getPutFile — PUT ファイルを取得する
- HttpRequest::getQueryData — クエリデータを取得する
- HttpRequest::getRawPostData — 生の POST データを取得する
- HttpRequest::getRawRequestMessage — 名前のリクエストメッセージを取得する
- HttpRequest::getRawResponseMessage — 生のレスポンスメッセージを取得する
- HttpRequest::getRequestMessage — リクエストメッセージを取得する
- HttpRequest::getResponseBody — レスポンスの本文を取得する
- HttpRequest::getResponseCode — レスポンスコードを取得する
- HttpRequest::getResponseCookies — レスポンスのクッキーを取得する

- HttpRequest::getResponseData — レスポンスデータを取得する
- HttpRequest::getResponseHeader — レスポンスヘッダを取得する
- HttpRequest::getResponseInfo — レスポンスの情報を取得する
- HttpRequest::getResponseMessage — レスポンスメッセージを取得する
- HttpRequest::getResponseStatus — レスポンスのステータスを取得する
- HttpRequest::getSslOptions — ssl オプションを取得する
- HttpRequest::getUrl — url を取得する
- HttpRequest::resetCookies — クッキーをリセットする
- HttpRequest::send — リクエストを送信する
- HttpRequest::setContentType — content type を設定する
- HttpRequest::setCookies — クッキーを設定する
- HttpRequest::setHeaders — ヘッダを設定する
- HttpRequest::setMethod — メソッドを設定する
- HttpRequest::setOptions — オプションを設定する
- HttpRequest::setPostFields — POST フィールドを設定する
- HttpRequest::setPostFiles — POST ファイルを設定する
- HttpRequest::setPutData — PUT データを設定する
- HttpRequest::setPutFile — PUT ファイルを設定する
- HttpRequest::setQueryData — クエリデータを設定する
- HttpRequest::setRawPostData — 生の POST データを設定する
- HttpRequest::setSslOptions — SSL オプションを設定する
- HttpRequest::setUrl — URL を設定する
- HttpRequestPool — HTTP リクエストプールクラス
- HttpRequestPool::attach — HttpRequest をアタッチする
- HttpRequestPool::__construct — HttpRequestPool のコンストラクタ
- HttpRequestPool::__destruct — HttpRequestPool のデストラクタ
- HttpRequestPool::detach — HttpRequest をデタッチする
- HttpRequestPool::getAttachedRequests — アタッチされているリクエストを取得する
- HttpRequestPool::getFinishedRequests — 終了したリクエストを取得する
- HttpRequestPool::reset — リクエストプールをリセットする
- HttpRequestPool::send — すべてのリクエストを送信する
- HttpRequestPool::socketPerform — ソケットアクションを実行する
- HttpRequestPool::socketSelect — ソケットの選択を実行する
- HttpResponse — HTTP レスポンスクラス
- HttpResponse::capture — スクリプトの出力を取り込む
- HttpResponse::getBufferSize — バッファサイズを取得する
- HttpResponse::getCacheControl — cache control を取得する
- HttpResponse::getCache — キャッシュを取得する
- HttpResponse::getContentDisposition — content disposition を取得する
- HttpResponse::getContentType — content type を取得する
- HttpResponse::getData — データを取得する
- HttpResponse::getETag — ETag を取得する
- HttpResponse::getFile — ファイルを取得する
- HttpResponse::getGzip — gzip を取得する
- HttpResponse::getHeader — ヘッダを取得する
- HttpResponse::getLastModified — 最終更新日時を取得する
- HttpResponse::getStream — ストリームを取得する
- HttpResponse::getThrottleDelay — throttle delay を取得する
- HttpResponse::getRequestBody — リクエストの本文を取得する
- HttpResponse::getRequestBodyStream — リクエストの本文をストリームとして取得する
- HttpResponse::getRequestHeaders — リクエストのヘッダを取得する
- HttpResponse::guessContentType — content type を推測する
- HttpResponse::redirect — リダイレクトする
- HttpResponse::send — レスポンスを送信する
- HttpResponse::setBufferSize — バッファサイズを設定する
- HttpResponse::setCacheControl — cache control を設定する
- HttpResponse::setCache — キャッシュを設定する
- HttpResponse::setContentDisposition — content disposition を設定する
- HttpResponse::setContentType — content type を設定する
- HttpResponse::setData — データを設定する
- HttpResponse::setETag — ETag を設定する
- HttpResponse::setFile — ファイルを設定する
- HttpResponse::setGzip — gzip を設定する
- HttpResponse::setHeader — ヘッダを設定する

- HttpResponse::setLastModified — 最終更新日時を設定する
- HttpResponse::setStream — ストリームを設定する
- HttpResponse::setThrottleDelay — throttle delay を設定する
- HttpResponse::status — HTTP レスポンスステータスを送信する
- http_cache_etag — ETag でキャッシュする
- http_cache_last_modified — 最終更新日時でキャッシュする
- http_chunked_decode — chunked-encoded データをデコードする
- http_deflate — データを圧縮する
- http_inflate — データを展開する
- http_get_request_body_stream — リクエストの本文をストリームとして取得する
- http_get_request_body — リクエストの本文を文字列として取得する
- http_get_request_headers — リクエストヘッダを配列として取得する
- http_date — HTTP の RFC に準拠した日付を作成する
- http_support — 組み込みの HTTP サポートを調べる
- http_match_etag — ETag を比較する
- http_match_modified — 最終更新日時を比較する
- http_match_request_header — 任意のヘッダを比較する
- http_build_cookie — クッキー文字列を作成する
- http_negotiate_charset — クライアントが希望している文字セットを選択する
- http_negotiate_content_type — クライアントが希望している content type を選択する
- http_negotiate_language — クライアントが希望している言語を選択する
- ob_deflatehandler — 圧縮出力ハンドラ
- ob_etaghandler — ETag 出力ハンドラ
- ob_inflatehandler — 展開出力ハンドラ
- http_parse_cookie — HTTP クッキーをパースする
- http_parse_headers — HTTP ヘッダをパースする
- http_parse_message — HTTP メッセージをパースする
- http_parse_params — パラメータリストをパースする
- http_persistent_handles_count — 持続ハンドルの状況
- http_persistent_handles_ident — 持続ハンドルの ident を取得/設定する
- http_persistent_handles_clean — 持続ハンドルを消去する
- http_get — GET リクエストを実行する
- http_head — HEAD リクエストを実行する
- http_post_data — エンコードされたデータを使用して POST リクエストを実行する
- http_post_fields — エンコードされる前のデータを使用して POST リクエストを実行する
- http_put_data — データを使用して PUT リクエストを実行する
- http_put_file — ファイルを使用して PUT リクエストを実行する
- http_put_stream — ストリームを使用して PUT リクエストを実行する
- http_request_method_exists — リクエストメソッドが存在するかどうかを調べる
- http_request_method_name — リクエストメソッド名を取得する
- http_request_method_register — リクエストメソッドを登録する
- http_request_method_unregister — リクエストメソッドの登録を解除する
- http_request — 独自のリクエストを実行する
- http_request_body_encode — リクエスト本文をエンコードする
- http_redirect — HTTP リダイレクトを発行する
- http_send_content_disposition — Content-Disposition を送信する
- http_send_content_type — Content-Type を送信する
- http_send_data — 任意のデータを送信する
- http_send_file — ファイルを送信する
- http_send_last_modified — Last-Modified を送信する
- http_send_status — HTTP レスポンスステータスを送信する
- http_send_stream — ストリームを送信する
- http_throttle — HTTP 抑止処理
- http_build_str — クエリ文字列を組み立てる
- http_build_url — URL を組み立てる

Hyperwave 関数

導入

Hyperwave は、Graz の » IICM において開発されました。 当初は Hyper-G という名前でしたが、 (1996年に)商品化された際に Hyperwave に変更されました。

Hyperwave はフリーソフトウエアではありません。 最新版は5.5で、» http://www.hyperwave.com/ から得ることができます。30 日間試用可能な版を注文することができます。

Hyperwave API モジュールも参照ください。

Hyperwave は、データベースに似た情報システム (HIS, Hyperwave Information Server)です。この システムは、文書の保存と管理に着目しています。文書には、あらゆる データとすることが可能で、同時にファイルに保存することもできます。 各文書には、オブジェクトレコードが付属しています。 オブジェクトレコードは、その文書のメタデータを有しています。 メタデータは、ユーザーにより拡張可能な属性のリストです。 ある種の属性は、Hyperwave サーバーにより常に設定されますが、 その他は、ユーザーにより修正可能です。 属性は、名前=値 という形式の 名前/値 の組です。完全なオブジェクトレコードは、ユーザーの指定した数のこの 組を有することができます。 属性の名前は、ユニークである必要はありません。 例えば、title はオブジェクトレコードの中に複数回現れる 可能性があります。 これは、複数の言語で title を指定したい場合に意味があります。 このような場合、 各 title の値は、2文字の言語の短縮型の後にコロンが続くものを 前に置くという慣習があります。 例えば、*'en:Title in English'*または *'ge:Titel in deutsch'*。 description または keywords のような他の属性は、潜在的な候補です。 残りの属性の値とコロンで区切ることにより他の文字列で言語の短縮形を 置換することも可能です。

各オブジェクトレコードは、各名前/キーの組で表した文字列であり、改 行で区切られています。Hyperwave拡張には、属性名をキーとした連想配 列としての第二の表現も有しています。多言語属性値は、それ自体で言 語の短縮形をキーとする別の連想配列を構成します。実際、複数の属性 は、属性値をキーとし、左にコロンがある文字列で連想配列を構成しま す。(これは完全に実装されていません。まだ、適正に処理されるのは、 属性 Title, Description, Keyword のみです。)

文書と共に文書中の全てのハイパーリンクは、同じくオブジェクトレコー ドとして保存されます。文書がデータベースに挿入される際には、文書 中のハイパーリンクは文書から取り除かれ独立したオブジェクトとして 保存されます。リンクに関するオブジェクトレコードは、リンクに関す る記述を開始した場所と終了した場所に関する情報を有しています。 オリジナルの文書を得るためには、リンクの無いプレーンな文書とリン クのリストを取り出し、リンクを再び挿入する必要があります。 (関数 hw_pipedocument() および hw_gettext() がこれを行います。) リンクを文書から分離する利点は明白です。リンクをされている文書が 名前を変更した場合でも、そのリンクは簡単に適宜変更可能です。 そのリンクを有する文書は、全く影響を受けません。 文書自体を変更せずに文書にリンクを追加することさえ可能です。

hw_pipedocument() および hw_gettext() が自動的にリンクの挿入を行う というのは、耳で聞く程簡単なことではありません。リンクの挿入は、 ある種のドキュメント階層の操作を含んでいます。Web サーバーではこ の機能はファイルシステムにより提供されますが、Hyperwave は固有の 階層を有しており、名前はその階層のオブジェクトの位置を反映しませ ん。このため、リンク作成時はまず Hyperwave の階層および名前空間か ら各Web名前空間の階層への対応付けを必要とします。HyperwaveとWebの 間の根本的な違いは、Hyperwaveでは名前と階層の間を明確に区別するこ とです。名前は、階層におけるオブジェクトの位置に関する情報を全く 有していません。Webでは、名前もオブジェクトの階層における位置に関 する情報を有しています。このため、対応付けには2種類の手法があります。 つまり、HyperwaveのHyperwaveオブジェクトの名前がURLに反 映されたものまたは名前のみのどちらかです。簡単のため、第2の方法が 使用されます。*my_object*という名前のHyperwaveオ ブジェクトは、Hyperwave階層のどこに位置するかによらず *http://host/my_object*に対応付けます。 Hyperwave階層においては、名前*parent/my_object* を有するオブジェクトは、Hyperwave階層の *my_object*の子となります。 しかし、Web の名前空間では正反対となるため、ユーザーに混乱を生じる 可能性があります。これは、適当なオブジェクト名を選択することによっ てのみ防止することができます。

この選択により第2の問題が生じます。PHPをどうやって起動しますか? URL *http://host/my_object*は、例えば、Webサー バーに対してこのURLを *http://host/php_script/my_object*に書き換えさせ ない限り、いかなる PHPスクリプトもコールしません。 スクリプト*php_script*は *$PATH_INFO*変数を評価し、Hyperwaveサーバーから 名前*my_object*を有するオブジェクトを取得します。 一つだけ欠点がありますが、簡単に修正可能です。URLの書き換えは、そ のWebサーバー上の他のドキュメントにアクセスする時には許可されませ ん。Hyperwaveサーバーで検索を行うPHPスクリプトは実現不可能です。 このため、例えば*http://host/Hyperwave*で始まる ようなURLを除くような少なくとも第2の書き換え規則を必要とします。

上記の機構的なリンクは、他のドキュメントに挿入されます。

PHPがサーバーモジュールでもCGIスクリプトでもなく、例えばCD-ROM上 にHyperwaveサーバーの内容をダンプするといったスタンドアロンのアプリケーションとして実行されている場合はより複雑になります。 このような場合、Hyperwave 階層を保ち、ファイルシステムに対応付けを 行う必要があります。 これは、(例えば、'/' を含む名前を選択することにより)固有の階層に 対応付けている場合には、オブジェクト名と衝突する可能性があります。 このため、'/' は他の文字、例えば '_' に置換されるべきです。

Hyperwaveサーバーと通信するためのネットワークプロトコルは、» HG-CSP(Hyper-G Client/Serverプロト コル)という名前です。このプロトコルは、例えばオブジェクトレコード を得るといったある動作を始めるためのメッセージに基づいています。 Hyperwaveサーバーの初期の版では、二つの専用クライアント (Harmony,Amadeus) がサーバーとの通信用に提供されていました。 これらは、Hyperwaveが商品化された際になくなりました。代わりに、 wavemasterが提供されました。wavemasterは、HTTPか らHG-CSPへのプロトコルコンバーターのようなもので す。これは、データベースの管理とドキュメントの可視化を全てWebイン ターフェースにより行うという考えによるものです。wavemasterは、イ ンターフェースをカスタマイズするための作業用に一連のプレースホル ダーを導入しています。この一連のプレースホルダーは、 PLACE言語と呼ばれています。 PLACEは、通常のプログラミング言語の機能の 多くを欠いており、言語への拡張としては、プレースホルダーの リストを拡大するだけです。 このことは、JavaScript を使用することにより作業が楽になったわけでは ないということと類似しています。

Hyperwave サポートを PHP に付加することにより、インターフェース のカスタマイズ用のプログラミング言語を有していないという穴を 埋めることになります。 この機能は、HG-CSP により定義された 全てのメッセージをサポートするだけでなく、 完全なドキュメントの取得といった更に強力なコマンドも提供します。

Hyperwave は、情報の特定の部分を名付けるために固有の用語法を 用います。 この方法は、広範に用いられ、拡張されています。 ほとんどすべの関数は、次のデータ型のどれかを操作します。

- object ID: Hyperwave サーバーの各オブジェクトについてユニークな 整数値。オブジェクトレコード (ObjectID) の属性の一つでもあります。object id は、オブジェクトを指定するための入力パラメータとして しばしば用いられます。
- object record: attribute=value 形式の属性-値の組となる文字列。 この組は、復改文字で他と区切られています。 オブジェクトレコードは、**hw_object2array()** により オブジェクトレコードに簡単に変換できます。 いくつかの関数は、オブジェクトレコードを返します。 これらの関数の名前は、obj で終わります。
- object array: オブジェクトの全ての属性を有する連想配列。 キーは属性名です。ある属性が、 オブジェクトレコードに複数回現れる場合、別の添字または連想配列 が生成されます。(title,keyword,description のような) 言語に依存する属性は、省略語をキーとした連想配列として 作成されます。 他の複数の属性は、添字配列として作成されます。 PHP 関数は、オブジェクト配列を返しません。
- hw_document: これは、完全に新規のデータ型であり、 HTML,PDF 等といった実際のドキュメントを保持します。 これは、幾分 HTML 用に最適化されていますが、他のフォーマット に使用することが可能です。

オブジェクトレコードの配列を返すいくつかの関数は、そのレコードに 関する統計情報を有する連想配列も返します。この配列は、オブジェク トレコード配列の最後の要素です。統計配列には、次のエントリがあり ます。

Hidden
: 属性 PresentationHints が Hidden であるオブジェクトレコードの数

CollectionHead
: 属性 PresentationHints が CollectionHead であるオブジェクト レコードの数

FullCollectionHead
: 属性 PresentationHints が FullCollectionHead であるオブジェクト レコードの数

CollectionHeadNr
: 属性 PresentationHints が CollectionHead であるオブジェクト レコードの配列のインデックス

FullCollectionHeadNr
: 属性 PresentationHints が FullCollectionHead であるオブジェクト レコードの配列のインデックス

Total
: Total: オブジェクトレコードの数

要件

この拡張モジュールは、Hyperwaveサーバを必要とします。このサーバ は、» http://www.hyperwave.com/でダウン ロード可能です。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

これらの関数を使用するには、オプション --with-hyperwave[=DIR] を使用して Hyperwave サポートつきで PHP をコンパイルする必要があります。

Windows ユーザがこれらの関数を使用するには、*php.ini* の中で *php_hyperwave.dll* を有効にします。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

Apache との組み合わせ

Hyperwave モジュールは、PHP が Apache モジュールとしてコンパイル された場合に、最適化されています。 この場合、Apache が書き換えエンジンを使用する場合、内部の Hyperwave サーバーをほぼ完全にユーザーから隠すことが可能です。 以下の手順によりこのことを説明します。

HyperwaveサポートをApacheに組み込んだPHPは、wavemaster に基づく本来のHyperwaveの手法を置換するものなので、Apacheサーバー は、Hyperwave Webインターフェースとしてのみ処理を行うと仮定します。 これは必然ではありませんが、設定が容易になります。考え方は非常に 簡単です。まず最初に*$_ENV['PATH_INFO']*変数を評 価し、Hyperwaveオブジェクトの名前としてその値を処理するPHPスクリ プトが必要です。このスクリプトを *'Hyperwave'*と 呼びましょう。URL `http://your.hostname/Hyperwave/name_of_object` *'name_of_object'*という名前のHyperwaveオブジェ クトを返します。オブジェクトの型に応じて、スクリプトは対応した処 理を行う必要があります。collectionの場合、恐らく子のリストを返す ことになります。ドキュメントの場合、MIME型と内容を返すことになり ます。Apacheの書換エンジンを使用した場合、若干の改善が見込まれま す。ユーザーの立場で見ると、URL `http://your.hostname/name_of_object` がオブジェクトを返せば、より簡単になります。書き換えの規則は非常 に簡単です。

```
RewriteRule  ^/(.*)  /usr/local/apache/htdocs/HyperWave/$1  [L]
```

この状態で、全てのURLはHyperwaveサーバーのオブジェクトに関係付け られています。これにより問題の解決が容易になります。'Hyperwave' スクリプト以外の検索といった他のスクリプトを実行することはできま せん。これは次のような別の規則により修正することができます。

```
RewriteRule  ^/hw/(.*)  /usr/local/apache/htdocs/hw/$1  [L]
```

これにより、ディレクトリ*/usr/local/apache/htdocs/hw*が追加の スクリプトや他のファイル用に確保されます。この規則は、一つ前の規 則よりも前に評価されることに注意してください。これには若干の欠点が あります。つまり、*'hw/'*で始まる名前を有する Hyperwave オブジェクトは全て隠されてしまいます。このため、このよ うな名前を使用しないようにしてください。例えば画像用により多くのディ レクトリが必要な場合、更に規則を加えるか一つのディレクトリに全て を置くかしてください。最後にRewriteEngineをオンにすることを忘れない でください。

```
RewriteEngine  on
```

経験上、次のようなスクリプトが必要になることと思われます。

- オブジェクト自身を返す
- 検索を許可する
- 自分を定義する
- プロファイルを設定する
- オブジェクト属性を表示したり、ユーザーに関する情報を表示したり、 サーバーのステータスを表示したり等する追加される関数毎に一つ

Rewrite Engineの一つの代わりとして、Apache *ErrorDocument*ディレクティブを使用することもで きます。しかし、*ErrorDocument* でリダイレクトさ れたページは、POSTデータを受け取ることはできないことに注意して下 さい。

実行時設定

php.ini の設定により動作が変化します。

Hyperwave 設定オプション

名前	デフォルト	変更の可否	変更履歴
hyperwave.allow_persistent	"0"	PHP_INI_SYSTEM	PHP 4.3.2 以降で使用可能です。PHP 5.0.0 で削除されました。
hyperwave.default_port	"418"	PHP_INI_ALL	PHP 5.0.0 以降で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

HW_ATTR_LANG (integer)
HW_ATTR_NR (integer)
HW_ATTR_NONE (integer)

Todo

まだやるべきことがいくつかあります。

- hw_InsertDocument は、 hw_insertobject()と **hw_putdocument()**に分割する必要があります。
- いくつかの関数の名前はまだ修正されていません。
- 多くの関数は最初のパラメータとして現在の接続 ID を 必要とします。これにより入力が多くかつ頻繁になってますが、 一つだけの接続をオープンしている場合には必要ないはずです。 デフォルト接続の導入により改善される見込みです。
- 多重属性を処理するには、オブジェクトレコードからオブジェクト配 列への変換機能が必要です。

---

hw_Array2Objrec

(PHP 4)

hw_Array2Objrec — オブジェクト配列からオブジェクトレコードに属性を変換する

説明

string **hw_array2objrec** (array `$object_array`)

`object_array` をオブジェクトレコードに変換します。 'Title' のような異なった言語で多重定義された属性は正しく処理されます。

hw_objrec2array() も参照ください。

---

hw_changeobject

(PHP 4)

hw_changeobject — オブジェクトの属性を変更する(古い関数)

説明

bool **hw_changeobject** (int $link , int $objid , array $attributes)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

hw_Children

(PHP 4)

hw_Children — 子のオブジェクト ID

説明

array **hw_children** (int $connection , int $objectID)

オブジェクト ID の配列を返します。各 ID は、ID *objectID* を有するコレクションの子に 属しています。 配列は、ドキュメントおよびコレクションの全ての子を有しています。

---

hw_ChildrenObj

(PHP 4)

hw_ChildrenObj — 子のオブジェクトレコード

説明

array **hw_childrenobj** (int $connection , int $objectID)

オブジェクトレコードの配列を返します。各オブジェクトレコードは、ID *objectID* を有するコレクションの子に属して います。配列は、ドキュメントおよびコレクションの全ての子を有しています。

---

hw_Close

(PHP 4)

hw_Close — Hyperwave 接続を閉じる

説明

bool **hw_close** (int $connection)

connection が有効な接続インデックスでない場合に **FALSE**、それ以外の 場合に **TRUE** を返します。指定した接続インデックスに関して Hyperwave サーバとの接続を閉じます。

---

hw_Connect

(PHP 4)

hw_Connect — 接続をオープンする

説明

int **hw_connect** (string $host , int $port [, string $username], string $password)

Hyperwave サーバへの接続をオープンし、成功した場合に接続インデックス、 接続できなかった場合に **FALSE** を返します。各引数は、ポート番号を除き引用符付の文字列である必要があります。 *username* および *password* 引数はオプションであり、省略することが できます。この場合、サーバによるユーザ識別は行われません。 これは、匿名ユーザとして識別されることに似ています。この関数は、 他の Hyperwave 関数で必要な接続インデッ

クスを返します。複数の接続を 同時にオープンすることが可能です。パスワードは暗号化されないことを 覚えておいてください。

hw_pConnect() も参照ください。

---

hw_connection_info

(PHP 4)

hw_connection_info — Hyperwave サーバへの接続に関する情報を出力する

説明

void **hw_connection_info** (int `$link`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

hw_cp

(PHP 4)

hw_cp — オブジェクトをコピーする

説明

int **hw_cp** (int `$connection` , array `$object_id_array` , int `$destination_id`)

2番名のパラメータとして指定されたオブジェクトID を有するオブジェ クト ID *`destination id`* を有するコレクション にコピーします。

返される値は、コピーされたオブジェクトの数です。

hw_mv() も参照ください。

---

hw_Deleteobject

(PHP 4)

hw_Deleteobject — オブジェクトを削除する

説明

bool **hw_deleteobject** (int `$connection` , int `$object_to_delete`)

2 番目のパラメータで指定されたオブジェクト ID を有するオブジェクトを 削除します。この関数は、オブジェクトの全てのインスタンスを削除します。

エラーが発生しない場合に **`TRUE`**、それ以外の場合 **`FALSE`** を返します。

hw_mv() も参照ください。

---

hw_DocByAnchor

(PHP 4)

hw_DocByAnchor — アンカーに属するオブジェクトのオブジェクト ID

説明

int **hw_docbyanchor** (int `$connection` , int `$anchorID`)

`anchorID` が指す ドキュメントのオブジェクト ID を返します。

---

hw_DocByAnchorObj

(PHP 4)

hw_DocByAnchorObj — アンカーが指すオブジェクトレコード

説明

string **hw_docbyanchorobj** (int `$connection` , int `$anchorID`)

anchorID が指す ドキュメントのオブジェクトレコードを返します。

hw_Document_Attributes

(PHP 4)

hw_Document_Attributes — hw_document のオブジェクトレコード

説明

string **hw_document_attributes** (int `$hw_document`)

ドキュメントのオブジェクトレコードを返します。

下位互換性のために**hw_DocumentAttributes()** も 使用可能です。しかし、この関数の使用は推奨されません。

hw_Document_BodyTag(), hw_Document_Size() も参照ください。

hw_Document_BodyTag

(PHP 4)

hw_Document_BodyTag — hw_document の BODY タグ

説明

string **hw_document_bodytag** (int `$hw_document` [, string `$prefix`])

ドキュメントの BODY タグを返します。そのドキュメントが HTML ドキュメントの場合、BODY タグがドキュメントの前に 出力されます。

hw_Document_Attributes(), hw_Document_Size() も参照ください。

下位互換性のため **hw_DocumentBodyTag()** も使用可能です。しかし、推奨されません。

hw_Document_Content

(PHP 4)

hw_Document_Content — hw_document の内容を返す

説明

string **hw_document_content** (int `$hw_document`)

ドキュメントの内容を返します。ドキュメントが HTML ドキュメントの場合、 内容は、全て BODY タグの後になります。HEAD および BODY タグからの情報は、 オブジェクトレコードに保存されます。

hw_Document_Attributes(), hw_Document_Size(), hw_Document_SetContent() も参照ください。

hw_Document_SetContent

(PHP 4)

hw_Document_SetContent — hw_document の内容を設定/置換する

説明

bool **hw_document_setcontent** (int `$hw_document` , string `$content`)

ドキュメントの内容を設定または置換します。ドキュメントが HTML ドキュメントの場合、内容はすべて BODY タグの後となります。HEAD および BODY タグからの情報は、オブジェクトレコードに保存されます。ドキュメントの 内容の中にもこの情報を与えた場合、Hyperwave サーバはドキュメントが 挿入された際にオブジェクトレコードを変更します。 これは恐らくあまり良い考えではありません。 この関数は、ドキュメントが古い内容を保持する場合に失敗します。

hw_Document_Attributes(), hw_Document_Size(), hw_Document_Content() も参照ください。

---

hw_Document_Size

(PHP 4)

hw_Document_Size — hw_document のサイ?

説明

int **hw_document_size** (int `$hw_document`)

ドキュメントのサイズをバイト数で返します。

hw_Document_BodyTag(), hw_Document_Attributes() も参照ください。

下位互換性のため、**hw_DocumentSize()** も使用可能です。しかし、この関数の使用は推奨されません。

---

hw_dummy

(PHP 4)

hw_dummy — Hyperwave ダミー関数

説明

string **hw_dummy** (int `$link` , int `$id` , int `$msgid`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

hw_EditText

(PHP 4)

hw_EditText — テキストドキュメントを取得する

説明

bool **hw_edittext** (int `$connection` , int `$hw_document`)

テキストドキュメントをサーバーにアップロードします。 ドキュメントのオブジェクトレコードは、そのドキュメントを編集している間、 修正することができません。 この関数は、純粋なテキストドキュメントについてのみ動作します。 この関数は、特別なデータ接続をオープンしないため、伝送の間、 制御用の接続をブロックします。

hw_pipedocument()、 hw_free_document()、 hw_document_bodytag()、 hw_document_size()、 hw_output_document() および hw_gettext() も参照ください。

---

hw_Error

(PHP 4)

hw_Error — エラー番号

説明

int **hw_error** (int `$connection`)

直近のエラー番号を返します。返り値が 0 の場合、エラーは 発生していません。エラーは直近のコマンドに関係しています。

---

hw_ErrorMsg

(PHP 4)

hw_ErrorMsg — エラーメッセージを返す

説明

string **hw_errormsg** (int `$connection`)

直近のエラーメッセージまたは 'No Error' を含む文字列を返します。 **FALSE** が返された場合、この関数は失敗しています。 このメッセージは、直近のコマンドに関係するものです。

---

hw_Free_Document

(PHP 4)

hw_Free_Document — hw_document を解放する

説明

bool **hw_free_document** (int `$hw_document`)

Hyperwave ドキュメントにより占有されたメモリを解放します。

---

hw_GetAnchors

(PHP 4)

hw_GetAnchors — ドキュメントのアンカーのオブジェクト ID

説明

array **hw_getanchors** (int `$connection` , int `$objectID`)

オブジェクト ID *`objectID`* を有するドキュメントの アンカーのオブジェクト ID の配列を返します。

---

hw_GetAnchorsObj

(PHP 4)

hw_GetAnchorsObj — ドキュメントのアンカーのオブジェクトレコード

説明

array **hw_getanchorsobj** (int `$connection` , int `$objectID`)

オブジェクト ID *`objectID`* を有するドキュメントの アンカーのオブジェクトレコードの配列を返します。

---

hw_GetAndLock

(PHP 4)

hw_GetAndLock — オブジェクトレコードを返しおよびオブジェクトをロックする

説明

string **hw_getandlock** (int `$connection` , int `$objectID`)

ID *`objectID`* を有するオブジェクトの オブジェクトレコードを返します。 また、オブジェクトをロックします。このため、ロックを解放するまで、 他のユーザーはアクセスできません。

hw_unlock() および hw_getobject() も参照ください。

---

hw_GetChildColl

(PHP 4)

hw_GetChildColl — 子のコレクションのオブジェクト ID

説明

array **hw_getchildcoll** (int `$connection` , int `$objectID`)

オブジェクト ID の配列を返します。各オブジェクト ID は、 ID *`objectID`* を有するコレクションの 子コレクションに属しています。 この関数は、子ドキュメントを返しません。

hw_children() および hw_getchilddoccoll() も参照ください。

---

hw_GetChildCollObj

(PHP 4)

hw_GetChildCollObj — 子のコレクションのオブジェクトレコード

説明

array **hw_getchildcollobj** (int `$connection` , int `$objectID`)

オブジェクトレコードの配列を返します。各オブジェクトレコードは、 ID *`objectID`* を有するコレクションの 子コレクションに属しています。 この関数は、子ドキュメントを返しません。

hw_childrenobj() および hw_getchilddoccollobj() も参照ください。

---

hw_GetChildDocColl

(PHP 4)

hw_GetChildDocColl — コレクションの子ドキュメントのオブジェクト ID

説明

array **hw_getchilddoccoll** (int `$connection` , int `$objectID`)

コレクションの子ドキュメントに関するオブジェクト ID の配列を 返します。

hw_children() および hw_getchildcoll() も参照ください。

---

hw_GetChildDocCollObj

(PHP 4)

hw_GetChildDocCollObj — コレクションの子ドキュメントのオブジェクトレコード

説明

array **hw_getchilddoccollobj** (int `$connection` , int `$objectID`)

コレクションの子ドキュメントに関するオブジェクトレコードの配列を 返します。

hw_childrenobj() および hw_getchildcollobj() も参照ください。

---

hw_GetObject

(PHP 4)

hw_GetObject — オブジェクトレコード

説明

mixed **hw_getobject** (int `$connection` , mixed `$objectID` [, string `$query`])

二番目のパラメータが整数(スカラー)の場合、ID *`objectID`* を有するオブジェクトの オブジェクトレコードを返します。 二番目のパラメータが整数の配列の場合、関数はオブジェクトレコードの 配列を返します。このような場合、最後のパラメータであるクエリー文字列も 評価されます。

クエリー文字列は次の構文を有しています。

<expr> ::= "(" <expr> ")" ¦

"!" <expr> ¦ /* NOT */

<expr> "¦¦" <expr> ¦ /* OR */

<expr> "&&" <expr> ¦ /* AND */

<attribute> <operator> <value>

<attribute> ::= /* 属性の名前 (Title, Author, DocumentType ...) */

<operator> ::= "=" ¦ /* 等しい */

"<" ¦ /* より小さい (文字列比較) */

">" ¦ /* より大きい (文字列比較) */

"~" /* 正規表現によるマッチング */

クエリーにより指定したオブジェクトのリストアら特定のオブジェクトを 選択することが可能となります。 他の関数とは異なり、このクエリーはインデックス付きの属性を使用しません。 返されるオブジェクトレコードの数は、クエリーおよび オブジェクトにアクセス可能であるかどうかに依存します。

hw_getandlock() および hw_getobjectbyquery() も参照ください。

---

hw_GetObjectByQuery

(PHP 4)

hw_GetObjectByQuery — オブジェクトを検索する

説明

array **hw_getobjectbyquery** (int `$connection` , string `$query` , int `$max_hits`)

サーバ全体のオブジェクトを検索し、オブジェクト ID の配列を 返します。最大マッチ数は、*`max_hits`* で制限 されます。*`max_hits`* が -1 にセットされた場合、 最大マッチ数は無制限となります。

クエリはインデックス付き属性でのみ動作します。

hw_getobjectbyqueryobj() も参照ください。

hw_GetObjectByQueryColl

(PHP 4)

hw_GetObjectByQueryColl — コレクションのオブジェクトを検索する

説明

array **hw_getobjectbyquerycoll** (int `$connection` , int `$objectID` , string `$query` , int `$max_hits`)

ID *objectID* を有するコレクションのオブジェクトを 検索し、オブジェクト ID の配列を返します。 最大マッチ数は、*max_hits* で制限されます。*max_hits* が -1 にセットされた場合、 最大マッチ数は、無制限となります。

クエリはインデックス付き属性でのみ動作します。

hw_getobjectbyquerycollobj() も参照ください。

---

hw_GetObjectByQueryCollObj

(PHP 4)

hw_GetObjectByQueryCollObj — コレクションのオブジェクトを検索する

説明

array **hw_getobjectbyquerycollobj** (int `$connection` , int `$objectID` , string `$query` , int `$max_hits`)

ID *objectID* を有するコレクションのオブジェクトを 検索し、オブジェクトレコードの配列を返します。 最大マッチ数は、*max_hits* で制限されます。 *max_hits* が -1 にセットされた場合、 最大マッチ数は、無制限となります。

クエリはインデックス付き属性でのみ動作します。

hw_getobjectbyquerycoll() も参照ください。

---

hw_GetObjectByQueryObj

(PHP 4)

hw_GetObjectByQueryObj — オブジェクトを検索する

説明

array **hw_getobjectbyqueryobj** (int `$connection` , string `$query` , int `$max_hits`)

サーバー全体のオブジェクトを検索し、オブジェクトレコードの配列を 返します。最大マッチ数は、*max_hits* で制限 されます。*max_hits* が -1 にセットされた場合、 最大マッチ数は無制限となります。

クエリはインデックス付き属性でのみ動作します。

hw_getobjectbyquery() も参照ください。

---

hw_GetParents

(PHP 4)

hw_GetParents — 親のオブジェクト ID

説明

array **hw_getparents** (int `$connection` , int `$objectID`)

オブジェクト ID の添字配列を返します。各オブジェクト ID は、 ID *objectID* を有するオブジェクトの 親に属しています。

hw_GetParentsObj

(PHP 4)

hw_GetParentsObj — 親のオブジェクトレコード

説明

array **hw_getparentsobj** (int `$connection` , int `$objectID`)

オブジェクトレコードの添字配列に加えてオブジェクトでコードに関する 統計情報を含む連想配列を返します。各オブジェクトレコードは、 ID *`objectID`* を有するオブジェクトの 親に属しています。

hw_getrellink

(PHP 4)

hw_getrellink — rootid に相対的な source から dest へのリンクを得る

説明

string **hw_getrellink** (int `$link` , int `$rootid` , int `$sourceid` , int `$destid`)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

hw_GetRemote

(PHP 4)

hw_GetRemote — リモートドキュメントを得る

説明

int **hw_getremote** (int `$connection` , int `$objectID`)

リモートドキュメントを返します。Hyperwave 表記のリモートドキュメントは、 外部ソースから取得されたドキュメントです。一般的なリモートドキュメントは、 例えば外部 Web ページまたはデータベースのクエリです。 外部ソースをアクセス可能とするために、Hyperwave では CGI に似た HGI(Hyperwave Gateway Interface)を導入しています。現在、 ftp,http サーバといくつかのデータベースのみが HGI によりアクセス可能です。 **hw_GetRemote()** をコールすることにより 外部ソースからのドキュメントが返されます。この関数を使用したい場合、 HGI を熟知している必要があります。外部ソースをアクセスするために Hyperwave のかわりに PHP を使用することも考慮する必要があります。 Hyperwave ゲートウエイにデータベースのサポートを付加するのは、PHP において同じことを行うよりもより困難です。

hw_getremotechildren() も参照ください。

hw_GetRemoteChildren

(PHP 4)

hw_GetRemoteChildren — リモートドキュメントの子を得る

説明

int **hw_getremotechildren** (int `$connection` , string `$object record`)

リモートドキュメントの子を返します。リモートドキュメントの 子は、リモートドキュメント自身です。これは、データベースクエリ の範囲を限定する必要があることを意味し、このことは、Hyperwave Programmers' Guide に説明されています。子の数が1の場合、関数は Hyperwave Gateway Interface (HGI) にフォーマットされたドキュメント 自体を返します。子の数が 1 より大きい場合、 **hw_GetRemoteChildren()** への他のコールに関する 入力値を有するオブジェクトレコードの配列を返します。 これらのオブジェクトレコードは仮想的なもので、Hyperwave サーバには 存在しません。 このため、有効なオブジェクト ID を有していません。 オブジェクトレコードにどれだけ似ているかは HGI 次第です。 この関数を使用したい場

合、HGI に習熟している必要があります。 外部ソースをアクセスするために Hyperwave のかわりに PHP を使用することも 考慮する必要があります。 Hyperwave ゲートウエイにデータベースのサポートを 付加するのは、PHP において同じことを行うよりもより困難です。

hw_getremote()も参照ください。

---

hw_GetSrcByDestObj

(PHP 4)

hw_GetSrcByDestObj — オブジェクトを指すアンカーを返す

説明

array **hw_getsrcbydestobj** (int `$connection` , int `$objectID`)

ID *objectID* を有するオブジェクトを指す 全てのアンカーのオブジェクトレコードを返します。 このオブジェクトは、ドキュメントまたはリンク先を指すアンカー のどちらかとすることができます。

hw_GetAnchors() も参照ください。

---

hw_GetText

(PHP 4)

hw_GetText — テキストドキュメントを取得する

説明

int **hw_gettext** (int `$connection` , int `$objectID` [, mixed `$rootID/prefix`])

オブジェクト ID *objectID* を有するドキュメントを 返します。ドキュメントが、挿入可能なアンカーを有している場合、 既に挿入されています。オプションのパラメータ *rootID/prefix* は、文字列または整数とすることが できます。整数の場合、ドキュメントに挿入されたリンクの数を定義 します。デフォルトは 0 であり、リンク先のオブジェクトの名前から 構成されたリンクが得られます。 この動作は、Web アプリケーションには便利です。 あるリンクが、名前 'internet_movie' を有するオブジェクトを指す 場合、HTML リンクは、<A HREF="/internet_movie"> となり、 ドキュメント階層におけるリンク元およびリンク先のオブジェクトの 実際の位置は無視されます。Web サーバーを設定し、URL を 例えば、'/my_script.php3/internet_movie' に書き変える必要があります。 'my_script.php3' は、$PATH_INFO を評価し、ドキュメントを 取得する必要があります。 全てのリンクは接頭辞 '/my_script.php3/'を有しています。 これが好ましくない場合、使用する接頭辞としてオプションのパラメータ *rootID/prefix* を設定することができます。 このパラメータは、文字列である必要があります。

rootID/prefix が整数であり、 0 に等しくない場合、リンクは ID *rootID/prefix* を有するオブジェクトで 始まる全ての名前から構成されます。 現在のオブジェクトに関したスラッシュにより分割された 全ての名前から構成されます。 例えば、前記のドキュメント 'internet_movie' が 'a-b-c-internet_movie' にあり、'-' が Hyperwave サーバの階層レベルの間のセパレータであり、 元のドキュメントが 'a-b-d-source' にある場合、 結果の HTML リンクは次のようになります。 <A HREF="../c/internet_movie"> この動作は、サーバー全体の内容をディスクにダウンロードし ドキュメント階層をファイルシステムに割り付けたい場合に便利です。

この関数は、純粋なテキストドキュメントについてのみ動作します。 特別なデータ接続をオープンしないため、伝達の間、 制御用の接続はブロックされます。

hw_pipedocument()、 hw_free_document()、 hw_document_bodytag()、 hw_document_size() および hw_output_document() も参照ください。

---

hw_getusername

(PHP 4)

hw_getusername — 現在ログインしているユーザーの名前

説明

string **hw_getusername** (int `$connection`)

接続を行っているユーザ名を返します。

---

hw_Identify

(PHP 4)

hw_Identify — ユーザとして認証する

説明

string **hw_identify** (int `$link` , string `$username` , string `$password`)

ユーザを *username* および *password* で認証します。 認証は、カレントのセッションでのみ有効です。 この関数が頻繁に必要とされるとは考えられません。 多くの場合、接続をオープンする際に認証を行う方がより 簡単でしょう。

hw_connect() も参照ください。

---

hw_InCollections

(PHP 4)

hw_InCollections — コレクションにオブジェクト ID があるかどうかを確認する

説明

array **hw_incollections** (int `$connection` , array `$object_id_array` , array `$collection_id_array` , int `$return_collections`)

object_id_array により指定された オブジェクトの組（ドキュメントまたはコレクション）が、 コレクション *collection_id_array* により定義された コレクションの一部であるかどうかを調べます。 4 番目のパラメータ *return_collections* が 0 である 場合、コレクションの一部(すなわち、それぞれ一つ以上のコレクション ID の コレクションまたはサブコレクションの子ドキュメントまたはコレクション) であるオブジェクト ID のサブセットが配列として返されます。 しかし、4 番目のパラメータが 1 である場合、 一つ以上のこのサブセットのオブジェクトを子として 有するコレクションの組は、配列として返されます。 このオプションにより、例えば、 前のクエリで一致したものを含むコレクション階層の一部を クライアントが、グラフィックで概要を強調表示することが可能となります。

---

hw_Info

(PHP 4)

hw_Info — 接続に関する情報

説明

string **hw_info** (int `$connection`)

現在の接続に関する情報を返します。返される配列は次のような フォーマットとなります。 <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

---

hw_InsColl

(PHP 4)

hw_InsColl — コレクションを挿入する

説明

int **hw_inscoll** (int `$connection` , int `$objectID` , array `$object_array`)

object_array の中の属性を有する新しいコレクションを オブジェクト ID *objectID* のコレクションに 挿入します。

---

hw_InsDoc

(PHP 4)

hw_InsDoc — ドキュメントを挿入する

説明

int **hw_insdoc** (resource `$connection` , int `$parentID` , string `$object_record` [, string `$text`])

object_record の中の属性を有する 新しいドキュメントを オブジェクト ID *parentID* のコレクションに 挿入します。 この関数は、オブジェクトレコードのみまたは *text* が指定された場合に *text* の中の純粋なアスキーテキストと オブジェクトレコードのどちらかを挿入します。 汎用の一般的なドキュメントを挿入したい場合は、代わりに hw_insertdocument() を使用してください。

hw_insertdocument() および hw_inscoll() も参照ください。

---

hw_insertanchors

(PHP 4 >= 4.0.4)

hw_insertanchors — テキストにアンカーのみを挿入する

説明

bool **hw_insertanchors** (int `$hwdoc` , array `$anchorecs` , array `$dest` [, array `$urlprefixes`])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

hw_InsertDocument

(PHP 4)

hw_InsertDocument — ドキュメントをアップロードする

説明

int **hw_insertdocument** (int `$connection` , int `$parent_id` , int `$hw_document`)

ドキュメントを *parent_id* を有するコレクションに アップロードします。 ドキュメントは、事前に **hw_NewDocument()** で 作成されている必要があります。 新規ドキュメントのオブジェクトレコードは、少なくとも次の属性を 有しているよう注意してください。: Type, DocumentType, Title ,Name MimeType を設定したい場合もあるかもしれません。 関数は、新規ドキュメントのオブジェクト ID または **FALSE** を返します。

hw_pipedocument() も参照ください。

---

hw_InsertObject

(PHP 4)

hw_InsertObject — オブジェクトレコードを挿入する

説明

int **hw_insertobject** (int `$connection` , string `$object_rec` , string `$parameter`)

オブジェクトをサーバに挿入します。オブジェクトは、全ての有効な Hyperwavve オブジェクトとすることができます。 パラメータに関する詳細な情報を説明するには、 HG-CSP ドキュメントを参照ください。

注意: アンカーを挿入したい場合、属性 Position は常に start/end の値または 'invisible' のどちらかに常に 設定されています。 不可視(invisible)位置は、注釈が対応する注釈文へのリンクを 有していない場合に必要です。

hw_pipedocument()、 hw_insertdocument()、 hw_insdoc() および hw_inscoll() も参照ください。

---

hw_mapid

(PHP 4)

hw_mapid — グローバル ID を仮想的なローカル ID に割りつける

説明

int **hw_mapid** (int `$connection` , int `$server_id` , int `$object_id`)

hyperwave サーバのグローバルオブジェクト ID を、 仮想的なオブジェクト ID に割りつけます。 これは、hw_connect() で接続を行っていない場合でも 行われます。 この仮想オブジェクト ID は、 hw_getobject() でオブジェクトレコードを得るといった用途に 他のオブジェクト ID と同様に使用することが可能です。 サーバ ID は、オブジェクトのグローバルオブジェクト ID(GOid)の最初の 部分であり、実際にはサーバの IP アドレスを整数で表したものとなります。

注意: この関数を使用するには、F_DISTRIBUTED フラグを設定しておく必要があります。 このフラグは、現在のところコンパイル時に hg_comm.c の中で設定することのみが可能です。 このフラグは、デフォルトでは設定されていません。 hg_comm.c の先頭にあるコメントをお読みください。

---

hw_Modifyobject

(PHP 4)

hw_Modifyobject — オブジェクトレコードを修正する

説明

bool **hw_modifyobject** (int `$connection` , int `$object_to_change` , array `$remove` , array `$add` [, int `$mode`])

このコマンドによりオブジェクトレコードの個々の属性を削除、追加、 修正することが可能となります。オブジェクトはオブジェクト ID *object_to_change* により指定します。最初の 配列 *remove* は、削除する属性のリストです。 二番目の配列 *add* は、加えたい属性のリストです。 ある属性を修正するためには、古い属性を削除し、新規に追加する 必要があります。**hw_modifyobject()** は、 削除する属性の値が文字列または配列でない限り、つねに属性を追加する前に 属性の削除を行います。

最後のパラメータは、修正を再帰的に行うかどうかを指定します。 1 は、再帰的な修正を意味します。オブジェクトのいくつかを修正できない 場合、通知されることなくスキップされます。 hw_error() は、いくつかのオブジェクトが 修正されなかったことからエラーを発生しない可能性があります。

二つの配列のキーは属性の名前です。各配列要素の値は、配列または文字列 またはその他のものとすることができます。配列の場合、各属性の 値は、各要素のキーにコロンと各要素の値を加えたものから構成されます。 文字列の場合、属性の値として指定します。空文字列を指定すると その属性は完全に削除されます。値が文字列でも配列でもなく、整数の ような他のものである場合、その属性に関して処理は全く行われません。 このような動作は、既存の属性に新規の値を追加するのではなく、 完全に新規の属性を追加したい場合に必要です。remove 配列がその属性に 関する空の文字列を含む場合、値が存在しないため、その属性の削除は 失敗します。続いてその属性に新規の値を追加する処理も同じく失敗します。 属性の値を例えば 0 に設定すると削除することさえできませんが、 加算は実行可能です。

値 'books' を有する属性 'Name' を 'articles' に変更したい場合、 二つの配列を作成し、**hw_modifyobject()** をコールする 必要があります。

Example#1 属性を修正する

```
<?php
       // $connect は、Hyperwave サーバへの既存の接続です。
        // $objid は、修正するオブジェクトの ID です。
       $remarr = array("Name" => "books");
       $addarr = array("Name" => "articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

名前=値の組を削除/追加するためには、オブジェクトレコードをremove/add 配列にて渡し、最後/3番目のパラメータは空の配列にします。 新規に属性を追加する場合には、 remove 配列におけるその属性の値を整数に設定してください。

Example#2 完全に新規の属性を追加する

```
<?php
       // $connect は、Hyperwave サーバへの既存の接続です。
        // $objid は、修正するオブジェクトの ID です。
       $remarr = array("Name" => 0);
       $addarr = array("Name" => "articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

> **注意**: 'Title' のような多言語属性は、2つの方法で修正可能です。 この場合、 'language':'title' という基本形式で属性の値を与えるか、 下記のように各言語毎に配列の要素を与えることにより指定します。 まずは、以下の例を見てみましょう。

Example#3 Title 属性を修正する

```
<?php
        $remarr = array("Title" => "en:Books");
       $addarr = array("Title" => "en:Articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

または

Example#4 Title 属性を修正する

```
<?php
        $remarr = array("Title" => array("en" => "Books"));
       $addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

この例では、英語のタイトル 'Books' を削除し、 英語のタイトル 'Articles' およびドイツ語のタイトル 'Artikel' を追加しています。

Example#5 属性を削除する

```
<?php
        $remarr = array("Title" => "");
       $addarr = array("Title" => "en:Articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

> **注意**: この例では、 'Title' という名前の属性を全て削除し、 属性 'Title' を新規に追加しています。 全ての属性を再帰的に削除したい場合には、これは簡便な方法です。
>
> **注意**: ある名前を有する全ての属性を削除する必要がある場合には、 属性の値として空の文字列を渡してください。
>
> **注意**: 属性 'Title', 'Description' , 'Keyword' のみが 言語指定用の接頭辞を正しく処理します。 これらの属性において言語指定用接頭辞が指定されていない場合、 接頭辞 'xx' が割りつけられます。
>
> **注意**: 'Name' 属性もやや特殊です。いくつかの場合には、完全に削除することは できません。 'Change of base attribute' というエラーメッセージが発生する可能性が あります(いつこのエラーが発生するかは明らかではありません) 。 このため、新しい Name をまず追加してから、古いものを削除する 必要があります。
>
> **注意**: この関数のコールの前後に hw_getandlock() および hw_unlock() をコールする必要はありません。 **hw_modifyobject()** は内部的にこの処理を 行います。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

hw_mv

(PHP 4)

hw_mv — オブジェクトを移動する

説明

int **hw_mv** (int `$connection` , array `$object_id_array` , int `$source_id` , int `$destination_id`)

2 番目のパラメータで指定されたオブジェクト ID のオブジェクトを ID *source id* のコレクションから ID *destination id* のコレクションに 移動します。 ディスティネーション ID が 0 の場合、オブジェクトはソースコレクション からリンクされていません。 これがオブジェクトの最後のインスタンスの場合、このオブジェクトは 削除されます。 全てのインスタンスを一度に削除したい場合、 hw_deleteobject()を使用してください。

移動されたオブジェクトの数が値として返されます。

hw_cp(), hw_deleteobject() も参照ください。

---

hw_New_Document

(PHP 4)

hw_New_Document — 新しいドキュメントを作成する

説明

int **hw_new_document** (string `$object_record` , string `$document_data` , int `$document_size`)

ドキュメントデータの組を *document_data* に、 オブジェクトレコードの組を *object_record* に 有する新しい Hyperwave ドキュメントを返します。 *document_data* の長さを *document_size* で渡す必要があります。 この関数は、ドキュメントを Hyperwave サーバに挿入しません。

hw_free_document()、 hw_document_size()、 hw_document_bodytag()、 hw_output_document() および hw_insertdocument() も参照ください。

---

hw_objrec2array

(PHP 4)

hw_objrec2array — 属性をオブジェクトレコードからオブジェクト配列に変換する

説明

array **hw_objrec2array** (string `$object_record` [, array `$ format`])

`object_record` をオブジェクト配列に変換します。 結果の配列のキーは属性の名前です。'Title' のように 異なった言語で多重定義を行う属性は、それ自体で配列となります。 この配列のキーは、属性の値のコロンの左の部分にあります。 現在、'Title', 'Description', 'Keyword' のみが適正に処理されます。 他の多値属性は、添字配列となります。この左側の部分は、2 文字分の長さと する必要があります。接頭辞のない他の多値属性は添字配列となり、 オプションのパラメータに属性 'Title'が指定されない場合、' Description'と'Keyword'は、言語属性として処理され、属性 'Group'、'Parent'、'HtmlAttr'は接頭辞のない多値属性として処理され ます。各属性の型を有する配列を指定することにより、この動作を変更 することが可能です。配列は、属性名をキーとする連想配列であり、 その値は、*HW_ATTR_LANG* または *HW_ATTR_NONE*のどちらかとなります。

hw_array2objrec() も参照ください。

---

hw_Output_Document

(PHP 4)

hw_Output_Document — hw_document を出力する

説明

bool **hw_output_document** (int `$hw_document`)

BODY タグを除きドキュメントを出力します。

下位互換性のため、**hw_OutputDocument()** も使用可能です。しかし、この関数の使用は推奨されません。

---

hw_pConnect

(PHP 4)

hw_pConnect — 持続的データベース接続を作成する

説明

int **hw_pconnect** (string `$host` , int `$port` [, string `$username`], string `$password`)

成功時に接続インデックスを、接続できなかった場合に **FALSE** を返します。 Hyperwave サーバへの持続的接続をオープンします。 各引数は、ポート番号を除き引用符で括った文字列である必要があります。 *`username`* と*`password`* 引数は オプションであり、省略することができます。この場合、サーバとの認証は 行われません。これは、匿名ユーザとして認証されることに似ています。 この関数は、他の Hyperwave 関数で必要な接続インデックスを返します。 同時に複数の持続的接続をオープンすることができます。

hw_connect() も参照ください。

---

hw_PipeDocument

(PHP 4)

hw_PipeDocument — ドキュメントを取得する

説明

int **hw_pipedocument** (int `$connection` , int `$objectID` [, array `$url_prefixes`])

オブジェクト ID *`objectID`* を有する Hyperwave ドキュメントを返します。 ドキュメントが、挿入可能なアンカーを有している場合、 既に挿入されているかもしれません。 ドキュメントは、制御用接続をブロックしない特別なデータ接続により 伝達されます。

リンク挿入に関する詳細は、hw_gettext()、 hw_free_document()、 hw_document_size()、 hw_document_bodytag() および hw_output_document() も参照ください。

---

hw_Root

(PHP 4)

hw_Root — ルートオブジェクト ID

説明

int **hw_root** ($)

hyperroot コレクションのオブジェクト ID を返します。 現在、これは常に 0 となります。hyperroot の子コレクションは、 接続するサーバのルートコレクションです。

---

hw_setlinkroot

(PHP 4)

hw_setlinkroot — 計算されたリンクの ID を設定する

説明

int **hw_setlinkroot** (int $link , int $rootid)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

hw_stat

(PHP 4)

hw_stat — ステータス文字列を返す

説明

string **hw_stat** (int $link)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

hw_Unlock

(PHP 4)

hw_Unlock — オブジェクトをアンロックする

説明

bool **hw_unlock** (int $connection , int $objectID)

ドキュメントをアンロックし、他のユーザーが再びアクセスできる ようにします。

hw_getandlock() も参照ください。

---

hw_Who

(PHP 4)

hw_Who — 現在ログイン中のユーザのリスト

説明

array **hw_who** (int $connection)

Hyperwave サーバに現在ログイン中のユーザの配列を返します。 この配列の各エントリは、要素 ID、名前、システム、onSinceDate, onSinceTime, TotalTime , self を有する配列です。 このエントリがリクエストを初期化したユーザに属している場合、 'self' は 1 となります。

目次

Hyperwave API 関数

導入

Hyperwave は、グラーツにある » IICM で開発されています。 当初は Hyper-G という名前でしたが、 商用化の際に (1996 年に) Hyperwave という名前に変わりました。

Hyperwave はフリーソフトウェアではありません。現在のバージョンである 5.5 は » http://www.hyperwave.com/ で入手可能です。 期間限定 (30 日) のフリー版も注文できます。

Hyperwave モジュールも参照ください。

Hyperwave は、データベースに似た情報システム (HIS, Hyperwave Information Server) で、 文書の保存や管理に重点をおいています。 文書は、いくつかのデータに分割されてファイルに保存されます。 個々の文書にはオブジェクトレコードが付属し、 オブジェクトレコードには文書のメタデータが含まれます。 メタデータは属性のリストで、これはユーザが拡張することができます。 属性の中のいくつかは Hyperwave サーバが常に設定し、 それ以外についてはユーザによって変更されます。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。
> PHP 5.2.0.

要件

2001 年より、Hyperwave SDK が使用可能です。これは Java、 JavaScript そして C++ をサポートしています。この PHP 拡張モジュールは C++ 版のインターフェイスをもとにしています。 PHP で hwapi のサポートを有効にするには、まずはじめに Hyperwave SDK をインストールしなければなりません。

インストール手順

Hyperwave SDK をインストールしてから、 --with-hwapi[=DIR] を指定して PHP の configure を行います。

Apache との統合

Apache やその他のサーバと統合するための情報は、すでに Hyperwave モジュールで説明されています。 これは、Hyperwave サーバに接続するための最初の拡張モジュールでした。

実行時設定

php.ini の設定により動作が変化します。

Hyperwave API 設定オプション

名前	デフォルト	変更の可否	変更履歴
hwapi.allow_persistent	"0"	PHP_INI_SYSTEM	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

クラス

HW_API 拡張モジュールが提供する API は、完全にオブジェクト指向です。 Hyperwave SDK の C++ インターフェイスに非常によく似ています。この API は、以下のクラスからなります。

- HW_API
- HW_API_Object
- HW_API_Attribute
- HW_API_Error
- HW_API_Content
- HW_API_Reason

Hyperwave SDK に含まれているクラスのうち、 HW_API_String、 HW_API_String_Array のような基本的なクラスは実装されていません。 PHP には、それらに代わる強力な機能があるからです。

各クラスにはメソッドが含まれており、その名前は Hyperwave SDK の対応するメソッドと同じです。この関数に引数を渡す方法は PHP の他の拡張モジュールとは少し異なっており、HW SDK の C++ API に似ています。複数のパラメータを別々に渡すのではなく、 それらをひとつの連想配列にまとめて単一のパラメータとして渡します。 連想配列のキーの名前は、HW SDK に記述されている内容と同じです。 一般的なパラメータについては以下で説明します。 それ以外のパラメータが必要になる場合は、必要に応じて説明されます。

- objectIdentifier オブジェクトの名前あるいは ID。 例 "rootcollection"、"0x873A8768 0x00000002"。
- parentIdentifier 親オブジェクトの名前あるいは ID。
- object HW_API_Object クラスのインスタンス。
- parameters HW_API_Object クラスのインスタンス。
- version オブジェクトのバージョン。
- mode 実行される操作を定義する整数値。

- attributeSelector 文字列の配列で、個々の要素は属性の名前となります。 オブジェクトレコードを取得する際に特定の属性を含めたい場合に使用します。
- objectQuery オブジェクトのリストの中から特定のオブジェクトを選択するためのクエリ。 **hw_api->children()** や **hw_api->find()** のような関数から返されるオブジェクトの数を減らすために使用します。

注意: boolean 型のメソッドは、 **TRUE**、**FALSE** あるいは HW_API_Error オブジェクトを返します。

---

hw_api_attribute->key

(No version information available, might be only in CVS)

hw_api_attribute->key — 属性のキーを返す

説明

hw_api_attribute
string **key** (void)

属性の名前を返します。

返り値

属性の名前を表す文字列を返します。

参考

- hw_api_attribute->value

---

hw_api_attribute->langdepvalue

(No version information available, might be only in CVS)

hw_api_attribute->langdepvalue — 指定した言語の値を返す

説明

hw_api_attribute
string **langdepvalue** (string $language)

属性の、指定した言語における値を返します。

パラメータ

language

返り値

属性の値を表す文字列を返します。

参考

- hw_api_attribute->value

---

hw_api_attribute->value

(No version information available, might be only in CVS)

hw_api_attribute->value — 属性の値を返す

説明

hw_api_attribute
string **value** (void)

属性の値を取得します。

返り値

属性の値を表す文字列を返します。

参考

- hw_api_attribute->key
- hw_api_attribute->values

---

hw_api_attribute->values

(No version information available, might be only in CVS)

hw_api_attribute->values — 属性のすべての値を返す

説明

hw_api_attribute
array **values** (void)

属性のすべての値を取得します。

返り値

属性の値を配列として返します。

参考

- hw_api_attribute->value

---

hw_api_attribute

(No version information available, might be only in CVS)

hw_api_attribute — hw_api_attribute クラスのインスタンスを作成する

説明

HW_API_Attribute **hw_api_attribute** ([string $name [, string $value]])

指定した名前および値を使用して、hw_api_attribute の新しいインスタンスを作成します。

パラメータ

name

属性の名前。

value

属性の値。

返り値

hw_api_attribute のインスタンスを返します。

---

hw_api->checkin

(No version information available, might be only in CVS)

hw_api->checkin — オブジェクトをチェックインする

説明

hw_api
bool **checkin** (array $parameter)

この関数は、オブジェクトあるいはオブジェクト階層の全体をチェックインします。 パラメータ配列の中には、必須要素 'objectIdentifier' およびオプションの要素 'version'、'comment'、'mode'、'objectQuery' が含まれます。'version' ではオブジェクトのバージョンを設定します。これは、メジャーバージョンとマイナーバージョンをピリオドで区切ったものです。 version が設定されていない場合、マイナーバージョンがひとつ加算されます。 'mode' は以下の値のいずれかです。

HW_API_CHECKIN_NORMAL
オブジェクトをチェックインし、コミットします。 オブジェクトは文書でなければなりません。
HW_API_CHECKIN_RECURSIVE
チェックインするオブジェクトがコレクションであった場合、 もし子要素が文書であればそれらもすべて再帰的にチェックインされます。 コレクションをチェックインしようとすると、エラーが発生することがあります。
HW_API_CHECKIN_FORCE_VERSION_CONTROL
バージョン管理されていないオブジェクトも強制的にチェックインします。
HW_API_CHECKIN_REVERT_IF_NOT_CHANGED
新しいバージョンが以前のバージョンと変わっているかどうかを調べます。 変わっている場合にのみオブジェクトがチェックインされます。
HW_API_CHECKIN_KEEP_TIME_MODIFIED
直近のオブジェクトの更新時刻を保持します。
HW_API_CHECKIN_NO_AUTO_COMMIT
オブジェクトのチェックイン時に、自動的なコミットは行われません。

パラメータ

parameter

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- hw_api->checkout

---

hw_api->checkout

(No version information available, might be only in CVS)

hw_api->checkout — オブジェクトをチェックアウトする

説明

hw_api
bool **checkout** (array $parameter)

この関数は、オブジェクトあるいはオブジェクト階層の全体をチェックアウトします。

パラメータ

parameter

パラメータ配列の中には、必須要素 'objectIdentifier' およびオプションの要素 'version'、'mode'、'objectQuery' が含まれます。
'mode' は以下の値のいずれかです。

HW_API_CHECKIN_NORMAL
オブジェクトをチェックアウトします。 オブジェクトは文書でなければなりません。
HW_API_CHECKIN_RECURSIVE
チェックアウトするオブジェクトがコレクションであった場合、 もし子要素が文書であればそれらもすべて再帰的にチェックアウトされます。
コレクションをチェックアウトしようとすると、エラーが発生することがあります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- hw_api->checkin

---

hw_api->children

(No version information available, might be only in CVS)

hw_api->children — オブジェクトの子を返す

説明

hw_api
array **children** (array $parameter)

コレクションの子、あるいは文書の属性を取得します。 オブジェクトクエリによって、取得する内容をさらに絞り込むことが可能です。

パラメータ

parameter

パラメータ配列の中には、必須要素 'objectIdentifier' およびオプションの要素 'attributeSelector'、'objectQuery' が含まれます。

返り値

HW_API_Object あるいは HW_API_Error 型のオブジェクトの配列が返されます。

参考

- hw_api->parents

---

hw_api_content->mimetype

(No version information available, might be only in CVS)

hw_api_content->mimetype — mimetype を返す

説明

hw_api_content
string **mimetype** (void)

コンテンツの mimetype を返します。

返り値

mimetype を表す文字列を返します。

---

hw_api_content->read

(No version information available, might be only in CVS)

hw_api_content->read — コンテンツを読み込む

説明

hw_api_content
string **read** (string $buffer , int $len)

コンテンツを *len* バイト分読み込み、 指定したバッファに保存します。

パラメータ

buffer

len

読み込むバイト数。

返り値

---

hw_api->content

(No version information available, might be only in CVS)

hw_api->content — オブジェクトのコンテンツを返す

説明

hw_api
HW_API_Content **content** (array $parameter)

この関数は、文書のコンテンツを hw_api_content 型のオブジェクトとして返します。

パラメータ

parameter

パラメータの配列には、必須要素 'objectidentifier' およびオプションの要素 'mode' が含まれます。 mode は **HW_API_CONTENT_ALLLINKS**、**HW_API_CONTENT_REACHABLELINKS** あるいは **HW_API_CONTENT_PLAIN** のうちのいずれかです。

HW_API_CONTENT_ALLLINKS は、 たとえそれが指している先が存在しない場合でもすべての対象アンカーを挿入します。

HW_API_CONTENT_REACHABLELINKS は、 到達可能なリンクのみを挿入するよう **hw_api_content()** に指示し、 **HW_API_CONTENT_PLAIN** は文書にリンクを一切含めません。

返り値

hw_api_content のインスタンスを返します。

---

hw_api->copy

(No version information available, might be only in CVS)

hw_api->copy — 物理的にコピーする

説明

hw_api
hw_api_content **copy** (array $parameter)

この関数は物理的なコピーを作成します。 もしコンテンツが含まれる場合は、それも含めてコピーされ、 新しいオブジェクトあるいはエラーオブジェクトを返します。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier' および 'destinationParentIdentifier' が含まれます。オプションのパラメータは 'attributeSelector' です。

返り値

コピーしたオブジェクトを返します。

参考

- hw_api->move
- hw_api->link

---

hw_api->dbstat

(No version information available, might be only in CVS)

hw_api->dbstat — データベースサーバの統計情報を返す

説明

hw_api
hw_api_object **dbstat** (array $parameter)

データベースサーバの統計情報を返します。

パラメータ

parameter

返り値

参考

- hw_api->dcstat
- hw_api->hwstat
- hw_api->ftstat

---

hw_api->dcstat

(No version information available, might be only in CVS)

hw_api->dcstat — 文書キャッシュサーバの統計情報を返す

説明

hw_api
hw_api_object **dcstat** (array $parameter)

文書キャッシュサーバの統計情報を返します。

パラメータ

parameter

返り値

参考

- hw_api->dbstat
- hw_api->hwstat
- hw_api->ftstat

---

hw_api->dstanchors

(No version information available, might be only in CVS)

hw_api->dstanchors — すべての対象アンカーの一覧を返す

説明

hw_api
array **dstanchors** (array $parameter)

オブジェクトのすべての対象アンカーを取得します。

パラメータ

parameter

パラメータ配列には、必須要素 'objectIdentifier'、オプションの要素 'attributeSelector' および 'objectQuery' が含まれます。

返り値

参考

- hw_api->srcanchors

---

hw_api->dstofsrcanchor

(No version information available, might be only in CVS)

hw_api->dstofsrcanchor — ソースアンカーの対象を返す

説明

hw_api
hw_api_object **dstofsrcanchor** (array $parameter)

指定したソースアンカーが指す対象オブジェクトを取得します。 対象オブジェクトは、対象アンカーあるいは文書全体となります。

パラメータ

parameter

パラメータ配列には、必須要素 'objectIdentifier' およびオプションの要素 'attributeSelector' が含まれます。

返り値

参考

- hw_api->srcanchors
- hw_api->dstanchors
- hw_api->objectbyanchor

---

hw_api_error->count

(No version information available, might be only in CVS)

hw_api_error->count — 原因の数を返す

説明

hw_api_error
int **count** (void)

エラーの原因の数を返します。

返り値

エラーの原因の数を返します。

参考

- hw_api_error->reason

---

hw_api_error->reason

(No version information available, might be only in CVS)

hw_api_error->reason — エラーの原因を返す

説明

hw_api_error
HW_API_Reason **reason** (void)

最初のエラーの原因を返します。

返り値

参考

- hw_api_error->count

---

hw_api->find

(No version information available, might be only in CVS)

hw_api->find — オブジェクトを検索する

説明

hw_api
array **find** (array $parameter)

この関数は、キーやフルテキストクエリを使用してオブジェクトを検索します。 見つかったオブジェクトは、オプションのオブジェクトクエリによって さらに絞り込むことができます。また、その重要性の順に並べ替えられます。 2 回目の検索は相対的に遅くなり、その結果はある程度絞り込まれます。 これによりインクリメンタルな検索ができ、前の検索の結果をもとに 検索を進めていくことが可能となります。

パラメータ

parameter

パラメータの配列には、 検索したい方法に応じて 'keyquery' や 'fulltextquery' が含まれます。オプションのパラメータは 'objectquery'、'scope'、'languages' および 'attributeselector' です。インクリメンタルな検索の場合、オプションのパラメータ 'startIndex'、'numberOfObjectsToGet' および 'exactMatchUnit' が渡されます。

返り値

---

hw_api->ftstat

(No version information available, might be only in CVS)

hw_api->ftstat — フルテキストサーバの統計情報を返す

説明

hw_api
hw_api_object **ftstat** (array $parameter)

フルテキストサーバの統計情報を返します。

パラメータ

parameter

返り値

参考

- hw_api->dcstat
- hw_api->dbstat
- hw_api->hwstat

---

hwapi_hgcsp

(No version information available, might be only in CVS)

hwapi_hgcsp — hw_api クラスのオブジェクトを返す

説明

HW_API **hwapi_hgcsp** (string $hostname [, int $port])

ホスト *hostname* 上の Hyperwave サーバへの接続をオープンします。 プロトコルは HGCSP を使用します。

パラメータ

hostname

ホスト名。

port

ポート番号を渡さなかった場合は、418 が用いられます。

返り値

HW_API のインスタンスを返します。

---

hw_api->hwstat

(No version information available, might be only in CVS)

hw_api->hwstat — Hyperwave サーバについての統計情報を返す

説明

hw_api
hw_api_object **hwstat** (array $parameter)

Hyperwave サーバについての統計情報を返します。

パラメータ

parameter

返り値

参考

- hw_api->dcstat
- hw_api->dbstat
- hw_api->ftstat

---

hw_api->identify

(No version information available, might be only in CVS)

hw_api->identify — Hyperwave サーバにログインする

説明

hw_api
bool **identify** (array $parameter)

Hyperwave サーバにログインします。

パラメータ

parameter

パラメータの配列には、 要素 'username' および 'password' が含まれている必要があります。

返り値

認証に失敗した場合は HW_API_Error 型のオブジェクト、 成功した場合は **TRUE** を返します。

---

hw_api->info

(No version information available, might be only in CVS)

hw_api->info — サーバ設定についての情報を返す

説明

hw_api
array **info** (array $parameter)

サーバ設定についての情報を返します。

パラメータ

parameter

返り値

参考

- hw_api->dcstat
- hw_api->dbstat
- hw_api->ftstat
- hw_api->hwstat

---

hw_api->insert

(No version information available, might be only in CVS)

hw_api->insert — 新しいオブジェクトを挿入する

説明

hw_api
hw_api_object **insert** (array $parameter)

新しいオブジェクトを挿入します。オブジェクトは ユーザ(user)・グループ(group)・文書(document)・アンカー(anchor) のいずれかの型となります。 オブジェクトの型に応じて、設定すべき他の属性が決まります。

パラメータ

parameter

パラメータの配列には、必須要素 'object' および 'content' (オブジェクトが文書である場合)、そしてオプションの要素として 'parameters'、'mode' および 'attributeSelector' が含まれます。 'object' は、オブジェクトの全ての属性を含む必要があります。 'parameters' は、例えば destination のような (属性のキーは 'Parent' です) 追加の属性を保持するものです 'content' は文書の内容です。'mode' は以下のフラグの組み合わせです。

HW_API_INSERT_NORMAL
　　オブジェクトはサーバに挿入されます。
HW_API_INSERT_FORCE_VERSION_CONTROL
HW_API_INSERT_AUTOMATIC_CHECKOUT
HW_API_INSERT_PLAIN
HW_API_INSERT_KEEP_TIME_MODIFIED
HW_API_INSERT_DELAY_INDEXING

返り値

参考

- hw_api->replace

---

hw_api->insertanchor

(No version information available, might be only in CVS)

hw_api->insertanchor — アンカー型の新しいオブジェクトを挿入する

説明

hw_api
hw_api_object **insertanchor** (array $parameter)

この関数は hwapi_insert() の短縮版です。 アンカー型のオブジェクトを挿入し、必要な属性を設定します。

パラメータ

parameter

パラメータの配列には、必須要素 'object' および 'documentIdentifier'、そしてオプションの要素 'destinationIdentifier'、'parameter'、'hint' および 'attributeSelector' が含まれます。 'documentIdentifier' は、アンカーが挿入される文書を指定します。アンカーの対象が既に存在する場合は、それを 'destinationIdentifier' に設定します。対象が存在しない場合は、 後で挿入する予定のオブジェクトの名前を 'hint' に設定しなければなりません。 オブジェクトが挿入された後に、アンカーの指し示す先は自動的に解決されます。

返り値

参考

- hw_api->insert
- hw_api->insertdocument
- hw_api->insertcollection

---

hw_api->insertcollection

(No version information available, might be only in CVS)

hw_api->insertcollection — コレクション型の新しいオブジェクトを挿入する

説明

hw_api
hw_api_object **insertcollection** (array $parameter)

この関数は hwapi_insert() の短縮版です。 コレクション型のオブジェクトを挿入し、必要な属性を設定します。

パラメータ

parameter

パラメータの配列には、必須要素 'object' および 'parentIdentifier'、そしてオプションの要素 'parameter' および 'attributeSelector' が含まれます。 個々の要素の意味については hwapi_insert() を参照ください。

返り値

参考

- hw_api->insert
- hw_api->insertdocument
- hw_api->insertanchor

---

hw_api->insertdocument

(No version information available, might be only in CVS)

hw_api->insertdocument — 文書型の新しいオブジェクトを挿入する

説明

hw_api
hw_api_object **insertdocument** (array $parameter)

この関数は hwapi_insert() の短縮版です。 内容を含むのオブジェクトを挿入し、文書に必要な属性を設定します。

パラメータ

parameter

パラメータの配列には、必須要素 'object'、'parentIdentifier' および 'content'、そしてオプションの要素 'mode'、'parameter' および 'attributeSelector' が含まれます。

個々の要素の意味については hwapi_insert() を参照ください。

返り値

参考

- hw_api->insert
- hw_api->insertcollection
- hw_api->insertanchor

---

hw_api->link

(No version information available, might be only in CVS)

hw_api->link — オブジェクトへのリンクを作成する

説明

hw_api
bool **link** (array $parameter)

オブジェクトへのリンクを作成します。このリンクへのアクセスは、 リンクが指している先のオブジェクトへのアクセスと同じになります。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier' および 'destinationParentIdentifier' が含まれます。
'destinationParentIdentifier' は対象のコレクションです。

返り値

この関数は、成功した場合に **TRUE**、それ以外の場合にエラーオブジェクトを返します。

参考

- hw_api->copy

---

hw_api->lock

(No version information available, might be only in CVS)

hw_api->lock — オブジェクトをロックする

説明

hw_api
bool **lock** (array $parameter)

この関数をコールすると、オブジェクトを排他的に編集するためにロックします。 ロックを解除できるのは、ロックしたユーザあるいはシステムユーザのみです。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier' とオプションのパラメータ 'mode' および 'objectquery' が含まれます。

'mode' は、オブジェクトをどのようにロックするのかを指定します。 **HW_API_LOCK_NORMAL** は、 ロックが解除されるまでオブジェクトをロックします。 **HW_API_LOCK_RECURSIVE** はコレクションの場合にのみ有効で、 コレクション内のすべてのオブジェクト・子コレクションをロックします。 **HW_API_LOCK_SESSION** は、 セッションが有効な間のみオブジェクトをロックします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- hw_api->unlock

---

hw_api->move

(No version information available, might be only in CVS)

hw_api->move — コレクション間でオブジェクトを移動する

説明

hw_api
bool **move** (array $parameter)

コレクション間でオブジェクトを移動します。

パラメータ

parameter

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- hw_objrec2array()

---

hw_api_content

(No version information available, might be only in CVS)

hw_api_content — hw_api_content クラスの新しいインスタンスを作成する

説明

HW_API_Content **hw_api_content** (string $content , string $mimetype)

文字列 *content* から新しいコンテンツオブジェクトを作成します。

パラメータ

content

mimetype

コンテンツの mime タイプ。

返り値

---

hw_api_object->assign

(No version information available, might be only in CVS)

hw_api_object->assign — オブジェクトを複製する

説明

hw_api_object
bool **assign** (array $parameter)

オブジェクトの属性を複製します。

パラメータ

parameter

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

hw_api_object->attreditable

(No version information available, might be only in CVS)

hw_api_object->attreditable — 属性が編集可能かどうかを調べる

説明

hw_api_object
bool **attreditable** (array $parameter)

属性が編集可能かどうかを調べます。

パラメータ

parameter

返り値

属性が編集可能な場合に **TRUE**、それ以外の場合に **FALSE** を返します。

---

hw_api_object->count

(No version information available, might be only in CVS)

hw_api_object->count — 属性の数を返す

説明

hw_api_object
int **count** (array $parameter)

属性の数を返します。

パラメータ

parameter

返り値

属性の数を返します。

---

hw_api_object->insert

(No version information available, might be only in CVS)

hw_api_object->insert — 新しい属性を挿入する

説明

hw_api_object
bool **insert** (HW_API_Attribute $attribute)

オブジェクトに新しい属性を追加します。

パラメータ

attribute

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- hw_api_object->remove

---

hw_api_object

(No version information available, might be only in CVS)

hw_api_object — hw_api_object クラスの新しいインスタンスを作成する

説明

hw_api_object **hw_api_object** (array $parameter)

hw_api_object クラスの新しいインスタンスを作成します。

パラメータ

parameter

返り値

参考

- hw_api->lock

---

hw_api_object->remove

(No version information available, might be only in CVS)

hw_api_object->remove — 属性を削除する

説明

hw_api_object
bool **remove** (string $name)

指定した名前の属性を削除します。

パラメータ

name

属性の名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- hw_api_object->insert

---

hw_api_object->title

(No version information available, might be only in CVS)

hw_api_object->title — title 属性を返す

説明

hw_api_object
string **title** (array $parameter)

title 属性を返します。

パラメータ

parameter

返り値

タイトルを文字列で返します。

---

hw_api_object->value

(No version information available, might be only in CVS)

hw_api_object->value — 属性の値を返す

説明

hw_api_object
string **value** (string $name)

指定した名前の属性の値を返します。

パラメータ

name

属性の名前。

返り値

指定した名前の属性の値を返します。 エラーが発生した場合には **FALSE** を返します。

---

hw_api->object

(No version information available, might be only in CVS)

hw_api->object — 属性の情報を取得する

説明

hw_api
hw_api_object **object** (array $parameter)

この関数は、いずれかのバージョンのオブジェクトの属性情報を取得します。 文書の内容は返しません。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier'、そしてオプションの要素 'attributeSelector' および 'version' が含まれます。

返り値

返されるオブジェクトは、 成功した場合に HW_API_Object、 エラーが発生した場合に HW_API_Error となります。

例

この単純な例では、オブジェクトを取得してエラーが発生したかどうかを調べます。

Example#1 オブジェクトの取得

```
<?php
function handle_error($error)
{
  $reason = $error->reason(0);
  echo "Type: <b>";
  switch ($reason->type()) {
    case 0:
      echo "Error";
      break;
    case 1:
      echo "Warning";
      break;
    case 2:
      echo "Message";
      break;
  }
  echo "</b><br />¥n";
  echo "Description: " . $reason->description("en") . "<br />¥n";
}

function list_attr($obj)
{
  echo "<table>¥n";
  $count = $obj->count();
  for ($i=0; $i<$count; $i++) {
    $attr = $obj->attribute($i);
    printf("<tr><td align=¥"right¥" bgcolor=¥"#c0c0c0¥"><b>%s</b></td><td bgcolor=¥"#F0F0F0¥">%s</td></tr>¥n",
           $attr->key(), $attr->value());
  }
  echo "</table>¥n";
}

$hwapi = hwapi_hgcsp($g_config[HOSTNAME]);
$parms = array("objectIdentifier"=>"rootcollection", "attributeSelector"=>array("Title", "Name", "DocumentType"));
$root = $hwapi->object($parms);
if (get_class($root) == "HW_API_Error") {
  handle_error($root);
  exit;
}
list_attr($root);
?>
```

参考

- hw_api->content

---

hw_api->objectbyanchor

(No version information available, might be only in CVS)

hw_api->objectbyanchor — アンカーが所属しているオブジェクトを返す

説明

hw_api
hw_api_object **objectbyanchor** (array $parameter)

この関数は、指定したアンカーが属しているオブジェクトを取得します。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier' およびオプションの要素 'attributeSelector' が含まれます。

返り値

参考

- hw_api->dstofsrcanchor
- hw_api->srcanchors
- hw_api->dstanchors

---

hw_api->parents

(No version information available, might be only in CVS)

hw_api->parents — オブジェクトの親を返す

説明

hw_api
array **parents** (array $parameter)

オブジェクトの親を取得します。オブジェクトクエリを指定することにより、 親をさらに絞り込むことが可能です。

パラメータ

parameter

パラメータの配列には、必須要素 'objectidentifier'、 そしてオプションの要素 'attributeselector' および 'objectquery' が含まれます。

返り値

返される値は、 HW_API_Object オブジェクトの配列あるいは HW_API_Error となります。

参考

- hw_api->children

---

hw_api_reason->description

(No version information available, might be only in CVS)

hw_api_reason->description — 原因の説明を返す

説明

hw_api_reason
string **description** (void)

原因の説明を返します。

返り値

原因の説明を返します。

---

hw_api_reason->type

(No version information available, might be only in CVS)

hw_api_reason->type — 原因の型を返す

説明

hw_api_reason
HW_API_Reason **type** (void)

原因の型を返します。

返り値

HW_API_Reason のインスタンスを返します。

---

hw_api->remove

(No version information available, might be only in CVS)

hw_api->remove — オブジェクトを削除する

説明

hw_api
bool **remove** (array $parameter)

指定した親から、オブジェクトを削除します。コレクションは、 再帰的に削除されます。

パラメータ

parameter

オプションのオブジェクトクエリを渡し、 条件を満たすオブジェクトのみを削除することも可能です。 それが最後のインスタンスである場合、オブジェクトは物理的に削除されます。

パラメータの配列には、必須要素 'objectidentifier' および 'parentidentifier' が含まれます。 ユーザやグループを削除したい場合には 'parentidentifier' は省略することができます。

オプションのパラメータ 'mode' は、削除をどのように行うかを指定します。 normal モードは、すべてのインスタンスが削除されるまで物理的な削除を行いません。 physical モードは、オブジェクトの全インスタンスが即時に削除されます。 removelinks モードは、そのオブジェクトが参照している先、 およびそのオブジェクトを参照している元も含めて削除されます。 nonrecursive の場合は、再帰的な削除は行われません。 空でないコレクションを削除しようとすると、エラーが発生します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- hw_api->move

---

hw_api->replace

(No version information available, might be only in CVS)

hw_api->replace — オブジェクトを置き換える

説明

hw_api
hw_api_object **replace** (array $parameter)

オブジェクトの属性および内容を置き換えます。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier' および 'object'、 そしてオプションの要素 'content'、'parameters'、'mode' および 'attributeSelector' が含まれます。 'objectIdentifier' には置換元のオブジェクトを指定します。 'object' には新しいオブジェクトを指定します。 'content' には新しいオブジェクトの内容を指定します。 'parameters' には HTML 文書の追加情報を指定します。
HTML_Language は、タイトルの言語の短縮形です。 HTML_Base は、HTML 文書の base 属性です。

'mode' は以下のフラグの組み合わせを指定します。

HW_API_REPLACE_NORMAL
　　サーバ上のオブジェクトを渡したオブジェクトに置き換えます。
HW_API_REPLACE_FORCE_VERSION_CONTROL
HW_API_REPLACE_AUTOMATIC_CHECKOUT
HW_API_REPLACE_AUTOMATIC_CHECKIN
HW_API_REPLACE_PLAIN
HW_API_REPLACE_REVERT_IF_NOT_CHANGED
HW_API_REPLACE_KEEP_TIME_MODIFIED

返り値

参考

- hw_api->insert

---

hw_api->setcommittedversion

(No version information available, might be only in CVS)

hw_api->setcommittedversion — 最後のバージョン以外のバージョンをコミットする

説明

hw_api

hw_api_object **setcommittedversion** (array $parameter)

文書をコミットします。コミットされたバージョンが、 読み込みアクセス権を持つユーザから見えるものとなります。 デフォルトでは、最後のバージョンがコミットされたバージョンとなります。

パラメータ

parameter

返り値

参考

- hw_api->checkin
- hw_api->checkout

---

hw_api->srcanchors

(No version information available, might be only in CVS)

hw_api->srcanchors — ソースアンカーの一覧を返す

説明

hw_api
array **srcanchors** (array $parameter)

オブジェクトの全てのソースアンカーを取得します。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier'、 そしてオプションの要素 'attributeSelector' および 'objectQuery' が含まれます。

返り値

参考

- hw_api->dstanchors

---

hw_api->srcsofdst

(No version information available, might be only in CVS)

hw_api->srcsofdst — 対象オブジェクトのソースを返す

説明

hw_api
array **srcsofdst** (array $parameter)

指定した対象を指しているすべてのソースアンカーを取得します。 対象となるオブジェクトとしては、 対象アンカーあるいはドキュメント全体を指定できます。

パラメータ

parameter

パラメータの配列には、必須要素 'objectIdentifier'、 そしてオプションの要素 'attributeSelector' および 'objectQuery' が含まれます。 この関数は、オブジェクトの配列あるいはエラーを返します。

返り値

参考

- hw_api->dstofsrcanchor

---

hw_api->unlock

(No version information available, might be only in CVS)

hw_api->unlock — オブジェクトのロックを解除する

説明

hw_api
bool **unlock** (array $parameter)

ロックされたオブジェクトのロックを解除します。 ロックを解除できるのは、そのオブジェクトをロックしたユーザか システムユーザのみです。

パラメータ

parameter

パラメータの配列には、 必須属性 'objectIdentifier'、そしてオプションのパラメータ 'mode' および 'objectquery' が含まれます。
'mode' の意味は hwapi_lock() 関数と同じです。

返り値

成功した場合に **TRUE**、それ以外の場合に HW_API_Error クラスのオブジェクトを返します。

参考

- hw_api->lock

---

hw_api->user

(No version information available, might be only in CVS)

hw_api->user — 自分自身のユーザオブジェクトを返す

説明

hw_api
hw_api_object **user** (array $parameter)

自分自身のユーザオブジェクトを返します。

パラメータ

parameter

返り値

参考

- hw_api->userlist

---

hw_api->userlist

(No version information available, might be only in CVS)

hw_api->userlist — ログイン中の全ユーザの一覧を返す

説明

hw_api
array **userlist** (array $parameter)

ログイン中の全ユーザの一覧を返します。

パラメータ

parameter

返り値

参考

- hw_api->user

目次

- hw_api->content — オブジェクトのコンテンツを返す
- hw_api->copy — 物理的にコピーする
- hw_api->dbstat — データベースサーバの統計情報を返す
- hw_api->dcstat — 文書キャッシュサーバの統計情報を返す
- hw_api->dstanchors — すべての対象アンカーの一覧を返す
- hw_api->dstofsrcanchor — ソースアンカーの対象を返す
- hw_api_error->count — 原因の数を返す
- hw_api_error->reason — エラーの原因を返す
- hw_api->find — オブジェクトを検索する
- hw_api->ftstat — フルテキストサーバの統計情報を返す
- hwapi_hgcsp — hw_api クラスのオブジェクトを返す
- hw_api->hwstat — Hyperwave サーバについての統計情報を返す
- hw_api->identify — Hyperwave サーバにログインする
- hw_api->info — サーバ設定についての情報を返す
- hw_api->insert — 新しいオブジェクトを挿入する
- hw_api->insertanchor — アンカー型の新しいオブジェクトを挿入する
- hw_api->insertcollection — コレクション型の新しいオブジェクトを挿入する
- hw_api->insertdocument — 文書型の新しいオブジェクトを挿入する
- hw_api->link — オブジェクトへのリンクを作成する
- hw_api->lock — オブジェクトをロックする
- hw_api->move — コレクション間でオブジェクトを移動する
- hw_api_content — hw_api_content クラスの新しいインスタンスを作成する
- hw_api_object->assign — オブジェクトを複製する
- hw_api_object->attreditable — 属性が編集可能かどうかを調べる
- hw_api_object->count — 属性の数を返す
- hw_api_object->insert — 新しい属性を挿入する
- hw_api_object — hw_api_object クラスの新しいインスタンスを作成する
- hw_api_object->remove — 属性を削除する
- hw_api_object->title — title 属性を返す
- hw_api_object->value — 属性の値を返す
- hw_api->object — 属性の情報を取得する
- hw_api->objectbyanchor — アンカーが所属しているオブジェクトを返す
- hw_api->parents — オブジェクトの親を返す
- hw_api_reason->description — 原因の説明を返す
- hw_api_reason->type — 原因の型を返す
- hw_api->remove — オブジェクトを削除する
- hw_api->replace — オブジェクトを置き換える
- hw_api->setcommittedversion — 最後のバージョン以外のバージョンをコミットする
- hw_api->srcanchors — ソースアンカーの一覧を返す
- hw_api->srcsofdst — 対象オブジェクトのソースを返す
- hw_api->unlock — オブジェクトのロックを解除する
- hw_api->user — 自分自身のユーザオブジェクトを返す
- hw_api->userlist — ログイン中の全ユーザの一覧を返す

---

i18n（国際化）関数

導入

i18n (internationalization:国際化) とは、 あるアプリケーションを多言語・多文化環境で使用できるようにすることを指します。 i18n サポートが対象とする範囲は、 日付や時刻の書式、通貨、単位、数値の書式、 さまざまな文字エンコーディングなどです。 PHP6 では、i18n や l10n (localization:地域化) をネイティブにサポートするようになります。

PHP6 では POSIX のロケールは廃止され、 代わりに ICU ライブラリに含まれるロケールを使用するようになります。

警告

この拡張モジュールは現在開発中のものであり、まだ一般には公開されていません。

要件

Unicode 拡張モジュール が必要です。

---

locale_get_default

(No version information available, might be only in CVS)

locale_get_default — デフォルトのロケールを取得する

説明

string **locale_get_default** (void)

この関数は、デフォルトのロケールを返します。これは、 PHP が何らかの機能をローカライズする際に使用します。 このロケールは、setlocale() やシステム設定の影響を受けないことに注意しましょう。

返り値

現在のロケールを表す文字列を返します。

例

Example#1 locale_get_default() の例

```
<?php

// デフォルトのロケールを取得します
echo locale_get_default();

// 新しいロケールを設定し……
locale_set_default('pt_PT');

// ……それを表示します。
echo locale_get_default();

?>
```

上の例の出力は以下となります。

```
en_US_POSIX
pt_PT
```

参考

- locale_set_default()

---

locale_set_default

(No version information available, might be only in CVS)

locale_set_default — デフォルトのロケールを設定する

説明

bool **locale_set_default** (string $name)

PHP プログラムのデフォルトのロケールを設定します。 この関数は、setlocale() やシステムロケールには何の影響も及ぼさないことに注意しましょう。

パラメータ

name

新しいロケール名。サポートされるロケールの一覧は » http://demo.icu-project.org/icu-bin/locexp にあります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 locale_set_default() の例

ここでは、 **locale_set_default()** を使用して sort() 関数をローカライズする例を示します。

```
<?php

// ソートする文字列のリスト
$array = array(
    'caramelo',
    'cacto',
    'caçada'
);

// ロケールを設定します（この場合はポルトガル語）
locale_set_default('pt_PT');

// 設定したロケールを使用して並べ替えます
sort($array, SORT_LOCALE_STRING);

print_r($array);
?>
```

上の例の出力は以下となります。

```
Array
```

```
(
    [0] => caçada
    [1] => cacto
    [2] => caramelo
)
```

ロケールを使用しない場合は、PHP は ASCII 文字コードを使用してソートを行います。そのため、 以下のような結果となります（これは間違った結果です）

```
Array
(
    [0] => cacto
    [1] => caramelo
    [2] => caçada
)
```

参考

- locale_get_default()

目次

IBM 関数 (PDO_IBM)

導入

PDO_IBM は PHP Data Objects (PDO) インターフェイスを実装したドライバで、 PHP から IBM のデータベースにアクセスできるようになります。

インストール手順

PDO_IBM 拡張モジュールをビルドするには、DB2 クライアントの v9.1 以降がシステムにインストールされている必要があります。 DB2 クライアントは、IBM の » Application Development Site からダウンロードできます。

> **注意**: **注意** DB2 クライアント v9.1 以降では、DB2 for Linux、 UNIX および Windows の v8、v9.1 サーバへの直接接続がサポートされています。
> DB2 クライアント v9.1 は、DB2 UDB for i5 および DB2 UDB for z/OS servers への接続もサポートしています。この接続には、別途 » DB2 Connect product を購入します。

PDO_IBM は » PECL 拡張モジュールです。 PECL 拡張モジュールのインストール の手順に従って PDO_IBM 拡張モジュールをインストールします。 **configure** コマンドを発行し、 DB2 クライアントのヘッダファイルやライブラリの位置をこのように指定します。

```
bash$  ./configure  --with-pdo-ibm=/path/to/sqllib[,shared]
```

configure コマンドは、デフォルトでは環境変数 *DB2DIR* の値を使用します。

PDO_IBM DSN

(No version information available, might be only in CVS)

PDO_IBM DSN — IBM データベースへの接続

説明

PDO_IBM のデータソース名 (DSN) は、IBM CLI DSN にもとづいています。 PDO_IBM DSN は、主に以下の部分から構成されます。

DSN プレフィックス

DSN プレフィックスは **ibm:** です。

DSN

DSN は、次のうちのいずれかです。

- a) *db2cli.ini* あるいは *odbc.ini* を使用したデータソースの設定
- b) カタログされたデータベース名、つまり、 DB2 クライアントにおけるデータベースエイリアス
- c) 完全な接続文字列。次のような形式になります。 DRIVER={IBM DB2 ODBC DRIVER};DATABASE=*database* ;HOSTNAME=*hostname* ;PORT=*port* ;PROTOCOL=TCPIP;UID=*username* ;PWD=*password* ; 各パラメータは次の内容を表します。

  database

データベースの名前。

hostname

データベースサーバのホスト名あるいは IP アドレス。

port

データベースが要求を待ち受けている TCP/IP ポート。

username

データベースに接続する際のユーザ名。

password

データベースに接続する際のパスワード。

例

Example#1 *db2cli.ini* を使用した PDO_IBM DSN の例

この例は、PDO_IBM DSN で DB2 データベースに接続する方法を示すものです。 このデータベースは、DB2_9 という名前で *db2cli.ini* にカタログされています。

```
$db = new PDO("ibm:DSN=DB2_9", "", "");

[DB2_9]
Database=testdb
Protocol=tcpip
Hostname=11.22.33.444
Servicename=56789
```

Example#2 接続文字列を使用した PDO_IBM DSN の例

この例は、PDO_IBM DSN で DB2 データベースに接続する方法を示すものです。 このデータベースの名前は **testdb** で、DB2 CLI の接続文字列構文を使用します。

```
$db = new PDO("ibm:DRIVER={IBM DB2 ODBC DRIVER};DATABASE=testdb;" .
    "HOSTNAME=11.22.33.444;PORT=56789;PROTOCOL=TCPIP;", "testuser", "tespass");
```

IBM DB2、Cloudscape および Apache Derby 関数

導入

これらの関数により、DB2 Call Level Interface (DB2 CLI) を使用した IBM DB2 Universal Database、IBM Cloudscape および Apache Derby データベースへのアクセスが可能となります。

要件

IBM DB2 Universal Database for Linux・UNIX・Windows、 IBM Cloudscape、Apache Derby に接続するには、 PHP を稼動させるコンピュータ上に IBM DB2 Universal Database client がインストールされていなければなりません。 この拡張モジュールは、DB2 バージョン 8.2 を対象にして開発およびテストが行われています。

IBM DB2 Universal Database for z/OS・iSeries に接続するには、IBM DB2 Connect あるいはそれと同等の DRDA ゲートウェイソフトウェアも必要となります。

Linux あるいは Unix についての要件

これらの関数を使用する前には、PHP 実行ファイルあるいは SAPI を実行するユーザで DB2 インスタンスを指定する必要があります。 *php.ini* で *ibm_db2.instance_name* を使用して DB2 インスタンス名を指定するか、 PHP 実行ファイルの起動前に DB2 インスタンスプロファイルを 読み込むことができます。

例えば、*db2inst1* という名前の DB2 インスタンスを */home/db2inst1/* に作成した場合には *php.ini* に以下の行を追加します。

```
ibm_db2.instance_name=db2inst1
```

php.ini にこれを指定しない場合は、 DB2 へのアクセス用の環境変数を設定するために 以下のコマンドを実行しなければなりません。

```
bash$ source /home/db2inst1/sqllib/db2profile
```

PHP が使用できる Web サーバからこれらの関数を使用するには、 *php.ini* で *ibm_db2.instance_name* を設定するか、 Web サーバの起動スクリプト (*/etc/init.d/httpd* あるいは */etc/init.d/apache* であることが多いでしょう) で DB2 インスタンス環境変数を読み込む必要があります。

インストール手順

ibm_db2 拡張モジュールをビルドするには、 DB2 アプリケーション開発用のヘッダファイルおよびライブラリが システムにインストールされていなければなりません。 これらのファイルはデフォルトではインストールされないので、 DB2 のインストーラをもう一度起動して このオプションを追加する必要があります。 IBM DB2 Universal Database の » サポートサイト にてフリーで公開されている DB2 Application Development Client に、これらのヘッダファイルが含まれています。

既に DB2 がインストールされている Linux や Unix 上に DB2 アプリケーション開発用のヘッダおよびライブラリを追加した場合は、 DB2 インスタンス内のヘッダファイルやライブラリに対する シンボリックリンクを更新するため、 コマンド **db2iupdt -e** を実行しなければなりません。

ibm_db2 は » PECL 拡張モジュールなので、 PECL 拡張モジュールのインストール の手順にしたがって PHP にインストールすることができます。
まず、DB2 ヘッダファイルおよびライブラリの場所を指定するために 次のように **configure** コマンドを実行します。

```
bash$  ./configure  --with-IBM_DB2=/path/to/DB2
```

configure コマンドのデフォルト値は */opt/IBM/db2/V8.1* となっています。

> **注意**: **IIS ユーザ向けの注意** ibm_db2 ドライバを Microsoft Internet Information Server (IIS) で使用する場合は、以下ようにする必要があります。
>
> - DB2 を、拡張オペレーティングセキュリティ (extended operating system security) つきでインストールします。
> - PHP のバイナリへのパス (デフォルトは C:\php\) を、システムの環境変数 PATH に追加します。
> - システムの環境変数をもうひとつ作成し、PHP.INI のある場所を指定します (例: PHPRC = C:\php\)。
> - IUSR_COMPUTERNAME を DB2USERS グループに追加します。

実行時設定

php.ini の設定により動作が変化します。

ibm_db2 設定オプション

名前	デフォルト	変更可能	変更履歴
ibm_db2.binmode	"1"	PHP_INI_ALL	
ibm_db2.i5_allow_commit	"0"	PHP_INI_SYSTEM	ibm_db2 1.4.9 以降で利用可能。
ibm_db2.i5_dbcs_alloc	"0"	PHP_INI_SYSTEM	ibm_db2 1.5.0 以降で利用可能。
ibm_db2.instance_name	NULL	PHP_INI_SYSTEM	ibm_db2 1.0.2 以降で利用可能。

以下に設定ディレクティブに関する簡単な説明を示します。

ibm_db2.binmode integer

このオプションは、バイナリデータを PHP アプリケーションで 使用する際のモードを制御します。

- 1 (DB2_BINARY)
- 2 (DB2_CONVERT)
- 3 (DB2_PASSTHRU)

ibm_db2.i5_allow_commit integer

このオプションは、i5 スキーマコレクションで使用するコミットモードを PHP アプリケーションで使用する際のモードを制御します。

- 0 はコミットしません (オーバーライドについては i5_commit を参照ください)。
- 1 はコミットを許可します (オーバーライドについては i5_commit を参照ください)。

ibm_db2.i5_dbcs_alloc integer

このオプションは、ラージ DBCS カラムバッファ用の内部 ibm_db2 アロケーションスキームを制御します。

- 0 は拡張アロケーションを使用しません (オーバーライドについては i5_dbcs_alloc を参照ください)。
- 1 は拡張アロケーションを使用します (オーバーライドについては i5_dbcs_alloc を参照ください)。

ibm_db2.instance_name string

Linux および UNIX では、カタログデータベースへの接続に使用する インスタンス名をこのオプションで指定します。 このオプションは、環境変数 DB2INSTANCE の設定を上書きします。

Windows では、このオプションは無視されます。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

DB2_BINARY (integer)
バイナリデータをそのままの形式で返します。デフォルトのモードです。
DB2_CONVERT (integer)
バイナリデータを十六進表現にエンコードし、ASCII 文字列として返します。
DB2_PASSTHRU (integer)
バイナリデータを **NULL** 値に変換します。
DB2_SCROLLABLE (integer)
ステートメントのリソースとして、スクロール可能なカーソルを指定します。 このモードでは結果セット内の行へのランダムアクセスが可能となりますが、 現在は IBM DB2 Universal Database でしかサポートされていません。
DB2_FORWARD_ONLY (integer)
ステートメントのリソースとして、前進のみ可能なカーソルを指定します。 これはデフォルトのカーソル型であり、 すべてのデータベースサーバでサポートされています。
DB2_PARAM_IN (integer)
PHP 変数を、ストアドプロシージャの IN パラメータとしてバインドします。
DB2_PARAM_OUT (integer)
PHP 変数を、ストアドプロシージャの OUT パラメータとしてバインドします。
DB2_PARAM_INOUT (integer)
PHP 変数を、ストアドプロシージャの INOUT パラメータとしてバインドします。

DB2_PARAM_FILE (integer)
カラムを直接ファイルにバインドし、そのデータを読み込みます。
DB2_AUTOCOMMIT_ON (integer)
自動コミットを有効にします。
DB2_AUTOCOMMIT_OFF (integer)
自動コミットを無効にします。
DB2_DOUBLE (integer)
変数を、データ型 DOUBLE、FLOAT あるいは REAL にバインドします。
DB2_LONG (integer)
変数を、データ型 SMALLINT、INTEGER あるいは BIGINT にバインドします。
DB2_CHAR (integer)
変数を、データ型 CHAR あるいは VARCHAR にバインドします。
DB2_CASE_NATURAL (integer)
カラム名の大文字小文字を変換せずに返します。
DB2_CASE_LOWER (integer)
カラム名を小文字に変換して返します。
DB2_CASE_UPPER (integer)
カラム名を大文字に変換して返します。
DB2_DEFERRED_PREPARE_ON (integer)
指定したステートメントリソースについて、遅延プリペアを有効にします。
DB2_DEFERRED_PREPARE_OFF (integer)
指定したステートメントリソースについて、遅延プリペアを無効にします。

リソース型

ibm_db2 拡張モジュールは、接続リソース・ステートメントリソース および結果セットリソースを返します。

---

db2_autocommit

(PECL ibm_db2:1.0-1.6.2)

db2_autocommit — データベース接続の AUTOCOMMIT の状態を取得または設定する

説明

mixed **db2_autocommit** (resource $connection [, bool $value])

指定した接続リソースについての AUTOCOMMIT の状態を設定あるいは取得します。

パラメータ

connection

db2_connect() あるいは db2_pconnect() が返した有効なデータベース接続リソース。

value

以下の定数のいずれか。

DB2_AUTOCOMMIT_OFF

AUTOCOMMIT を無効にします。

DB2_AUTOCOMMIT_ON

AUTOCOMMIT を有効にします。

返り値

db2_autocommit() に *connection* パラメータのみを渡した場合、 指定した接続の AUTOCOMMIT の状態を整数値で返します。 0 の場合は無効、1 の場合は有効です。

db2_autocommit() に *connection* および *autocommit* の両方のパラメータを渡した場合、 指定した接続の AUTOCOMMIT を対応する状態に設定します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 接続の AUTOCOMMIT 値の取得

以下の例では、AUTOCOMMIT を無効にした接続について **db2_autocommit()** 関数で調べます。

```
<?php
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF);
$conn = db2_connect($database, $user, $password, $options);
$ac = db2_autocommit($conn);
if ($ac == 0) {
    print "$ac -- AUTOCOMMIT は無効です。";
} else {
    print "$ac -- AUTOCOMMIT が有効です。";
}
?>
```

上の例の出力は以下となります。

```
0 -- AUTOCOMMIT は無効です。
```

Example#2 接続の AUTOCOMMIT 値の設定

以下の例では、AUTOCOMMIT を無効にして作成した接続の AUTOCOMMIT を変更し、有効にします。

```
<?php
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF);
$conn = db2_connect($database, $user, $password, $options);

// AUTOCOMMIT を有効にします
$rc = db2_autocommit($conn, DB2_AUTOCOMMIT_ON);
if ($rc) {
    print "AUTOCOMMIT を有効にしました。\n";
}

// AUTOCOMMIT の状態を調べます
$ac = db2_autocommit($conn);
if ($ac == 0) {
    print "$ac -- AUTOCOMMIT は無効です。";
} else {
    print "$ac -- AUTOCOMMIT が有効です。";
}
?>
```

上の例の出力は以下となります。

```
AUTOCOMMIT を有効にしました。
1 -- AUTOCOMMIT が有効です。
```

参考

- db2_connect()
- db2_pconnect()

---

db2_bind_param

(PECL ibm_db2:1.0-1.6.2)

db2_bind_param — PHP 変数を SQL 文のパラメータにバインドする

説明

bool **db2_bind_param** (resource $stmt , int $parameter-number , string $variable-name [, int $parameter-type [, int $data-type [, int $precision [, int $scale]]]])

db2_prepare() が返したステートメントリソース内の SQL 文のパラメータに PHP 変数をバインドします。 db2_execute() へのオプション配列の一部として 変数を渡すのに比べ、この関数を使用すると パラメータの型、データの型、精度、位取りなどの詳細を制御できるようになります。

パラメータ

stmt

db2_prepare() が返すプリペアドステートメント。

parameter-number

プリペアドステートメントのパラメータの位置。 1 から始まります。

variable-name

parameter-number で指定したパラメータに バインドする PHP 変数の名前を表す文字列。

parameter-type

A constant specifying whether the PHP variable should be bound to the PHP 変数を SQL パラメータにバインドする際に 入力パラメータとするのか (*DB2_PARAM_IN*) 出力パラメータとするのか (*DB2_PARAM_OUT*) あるいは入出力両方を許可するのか (*DB2_PARAM_INOUT*) を指定する定数。 メモリのオーバーヘッドを避けるため、*DB2_PARAM_FILE* を指定して PHP 変数をファイルにバインドし、 ファイルからラージオブジェクト (BLOB、CLOB あるいは DBCLOB) データを読み込むようにするすることも可能です。

data-type

PHP 変数をどの SQL データ型にバインドするのかを指定する定数。 *DB2_BINARY*、 *DB2_CHAR*、*DB2_DOUBLE* あるいは *DB2_LONG* のうちのいずれか。

precision

変数をデータベースにバインドする際の精度を指定します。

scale

変数をデータベースにバインドする際の位取りを指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 PHP 変数をプリペアドステートメントにバインドする

以下の例の SQL 文では、WHERE 句の中で 2 つの入力パラメータを使用しています。 この 2 つのパラメータに PHP 変数をバインドするために **db2_bind_param()** をコールします。 **db2_bind_param()** をコールする前に PHP 変数を宣言したり代入したりする必要がないことに注意しましょう。 この例では、*$lower_limit* については **db2_bind_param()** がコールされる前に 値が代入されていますが、*$upper_limit* については

db2_bind_param() をコールした後で代入されています。 db2_execute() をコールする前には、 必ず変数をバインドしなければなりません。また、入力パラメータについては 何らかの値を代入しておく必要があります。

```
<?php

$sql = 'SELECT name, breed, weight FROM animals
     WHERE weight > ? AND weight < ?';
$conn = db2_connect($database, $user, $password);
$stmt = db2_prepare($conn, $sql);

// db2_bind_param() のコール前に変数を宣言できます
$lower_limit = 1;

db2_bind_param($stmt, 1, "lower_limit", DB2_PARAM_IN);
db2_bind_param($stmt, 2, "upper_limit", DB2_PARAM_IN);

// また、db2_bind_param() をコールした後で変数を宣言することも可能です
$upper_limit = 15.0;

if (db2_execute($stmt)) {
     while ($row = db2_fetch_array($stmt)) {
         print "{$row[0]}, {$row[1]}, {$row[2]}¥n";
     }
}
?>
```

上の例の出力は以下となります。

```
Pook, cat, 3.2
Rickety Ride, goat, 9.7
Peaches, dog, 12.3
```

Example#2 IN および OUT パラメータを使用したストアドプロシージャのコール

以下の例のストアドプロシージャ match_animal は、 3 つのさまざまなパラメータを受け取ります。

1. 最初の動物の名前を入力として受け取る 入力 (IN) パラメータ。
2. 2 番目の動物の名前を入力として受け取り、その名前に一致する動物が データベースに存在する場合に文字列 *TRUE* を返す、入出力 (INOUT) パラメータ。
3. 指定した 2 匹の動物の合計体重を返す、出力 (OUT) パラメータ。

さらにこのストアドプロシージャは結果セットを返します。その内容は、 最初のパラメータで指定した動物から 2 番目のパラメータで指定した動物までの 動物の一覧をアルファベット順に並べたものとなります。

```
<?php

$sql = 'CALL match_animal(?, ?, ?)';
$conn = db2_connect($database, $user, $password);
$stmt = db2_prepare($conn, $sql);

$name = "Peaches";
$second_name = "Rickety Ride";
$weight = 0;

db2_bind_param($stmt, 1, "name", DB2_PARAM_IN);
db2_bind_param($stmt, 2, "second_name", DB2_PARAM_INOUT);
db2_bind_param($stmt, 3, "weight", DB2_PARAM_OUT);

print "コール前のバインド変数の値:\n";
print "   1: {$name} 2: {$second_name} 3: {$weight}¥n¥n";

if (db2_execute($stmt)) {
    print "コール後のバインド変数の値:\n";
     print "   1: {$name} 2: {$second_name} 3: {$weight}¥n¥n";

     print "結果:\n";
     while ($row = db2_fetch_array($stmt)) {
         print "   {$row[0]}, {$row[1]}, {$row[2]}¥n";
     }
}
?>
```

上の例の出力は以下となります。

```
コール前のバインド変数の値:
   1: Peaches 2: Rickety Ride 3: 0

コール後のバインド変数の値:
   1: Peaches 2: TRUE 3: 22

結果:
   Peaches, dog, 12.3
   Pook, cat, 3.2
   Rickety Ride, goat, 9.7
```

Example#3 バイナリラージオブジェクト (BLOB) をファイルから直接挿入する

ラージオブジェクトのデータは、通常は XML ドキュメントあるいは 音声ファイルのようなファイルに保存されています。 いったんファイルの内容を PHP 変数に読み込んだうえで SQL 文にバインドする代わりに、ファイルを直接 sql 文の入力パラメータに バインドすることでメモリのオーバーヘッドを避けることができます。 以下の例で、BLOB カラムにファイルを直接バインドする方法を説明します。

```
<?php
```

```
$stmt = db2_prepare($conn, "INSERT INTO animal_pictures(picture) VALUES (?)");

$picture = "/opt/albums/spook/grooming.jpg";
$rc = db2_bind_param($stmt, 1, "picture", DB2_PARAM_FILE);
$rc = db2_execute($stmt);
?>
```

参考

- db2_execute()
- db2_prepare()

---

db2_client_info

(PECL ibm_db2:1.1.1-1.6.2)

db2_client_info — DB2 データベースクライアントの情報をプロパティに保持するオブジェクトを返す

説明

object **db2_client_info** (resource $connection)

この関数は、DB2 データベースクライアントについての情報を 読み取り専用のプロパティに保持するオブジェクトを返します。 以下の表は、DB2 クライアントプロパティの一覧です。

DB2 クライアントプロパティ

プロパティ名	返り値の型	説明
APPL_CODEPAGE	int	アプリケーションのコードページ。
CONN_CODEPAGE	int	現在の接続のコードページ。
DATA_SOURCE_NAME	string	現在のデータベース接続に使用されているデータソース名 (DSN)。
DRIVER_NAME	string	DB2 コールレベルインターフェイス (CLI) の仕様を実装した ライブラリの名前。
DRIVER_ODBC_VER	string	DB2 クライアントがサポートしている ODBC のバージョン。 "MM.mm" という形式の文字列で、*MM* がメジャーバージョン、*mm* がマイナーバージョンを表します。DB2 クライアントは、 常に "03.51" を返します。
DRIVER_VER	string	クライアントのバージョン。"MM.mm.uuuu" という形式の文字列で、 *MM* がメジャーバージョン、 *mm* がマイナーバージョン、 そして *uuuu* がアップデートを表します。 例えば "08.02.0001" は メジャーバージョン 8、マイナーバージョン 2、アップデート 1 を表します。
ODBC_SQL_CONFORMANCE	string	クライアントがサポートする ODBC SQL 構文レベル。 MINIMUM 最小限の ODBC SQL 構文をサポートします。 CORE コア ODBC SQL をサポートします。 EXTENDED 拡張 ODBC SQL 構文をサポートします。
ODBC_VER	string	ODBC ドライバマネージャがサポートする ODBC のバージョン。 "MM.mm.rrrr" という形式の文字列で、 *MM* がメジャーバージョン、 *mm* がマイナーバージョン、 そして *rrrr* がリリースを表します。 DB2 クライアントは、常に "03.01.0000" を返します。

パラメータ

connection

アクティブな DB2 クライアント接続を指定します。

返り値

成功した場合にオブジェクト、失敗した場合に **FALSE** を返します。

例

Example#1 db2_client_info() の例

クライアントの情報を取得するには、有効なデータベース接続リソースを **db2_client_info()** に渡す必要があります。

```
<?php
$conn = db2_connect( 'SAMPLE', 'db2inst1', 'ibmdb2' );
$client = db2_client_info( $conn );
```

```
if ($client) {
     echo "DRIVER_NAME:  ";            var_dump( $client->DRIVER_NAME  );
     echo "DRIVER_VER:  ";             var_dump( $client->DRIVER_VER  );
     echo "DATA_SOURCE_NAME:  ";       var_dump( $client->DATA_SOURCE_NAME  );
     echo "DRIVER_ODBC_VER:  ";        var_dump( $client->DRIVER_ODBC_VER  );
     echo "ODBC_VER:  ";               var_dump( $client->ODBC_VER  );
     echo "ODBC_SQL_CONFORMANCE:  ";   var_dump( $client->ODBC_SQL_CONFORMANCE  );
     echo "APPL_CODEPAGE:  ";          var_dump( $client->APPL_CODEPAGE  );
     echo "CONN_CODEPAGE:  ";          var_dump( $client->CONN_CODEPAGE  );
}
else {
    echo "クライアント情報の取得エラー。
      おそらくデータベース接続が無効です。";
}
db2_close($conn);

?>
```

上の例の出力は以下となります。

```
DRIVER_NAME:  string(8)  "libdb2.a"
DRIVER_VER:  string(10)  "08.02.0001"
DATA_SOURCE_NAME:  string(6)  "SAMPLE"
DRIVER_ODBC_VER:  string(5)  "03.51"
ODBC_VER:  string(10)  "03.01.0000"
ODBC_SQL_CONFORMANCE:  string(8)  "EXTENDED"
APPL_CODEPAGE:  int(819)
CONN_CODEPAGE:  int(819)
```

参考

- db2_server_info()

---

db2_close

(PECL ibm_db2:1.0-1.6.2)

db2_close — データベース接続を閉じる

説明

bool **db2_close** (resource $connection)

この関数は、db2_connect() で作成した DB2 クライアント接続を閉じ、データベースサーバへのリソースを返します。

db2_pconnect() で作成した持続的 DB2 クライアント接続を閉じようとするとその要求は無視され、 次にコールされるときまで持続的 DB2 クライアント接続はそのまま残ります。

パラメータ

connection

アクティブな DB2 クライアント接続を指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 接続を閉じる

以下では IBM DB2、Cloudscape あるいは Apache Derby データベースとの接続が正常に閉じられた場合の例を示します。

```
<?php
$conn = db2_connect('SAMPLE', 'db2inst1', 'ibmdb2');
$rc = db2_close($conn);
if ($rc) {
    echo "接続が正しく閉じられました。";
}
?>
```

上の例の出力は以下となります。

```
接続が正しく閉じられました。
```

参考

- db2_connect()
- db2_pconnect()

---

db2_column_privileges

(PECL ibm_db2:1.0-1.6.2)

db2_column_privileges — テーブルのカラムおよび関連する権限情報を含む結果セットを返す

説明

resource **db2_column_privileges** (resource $connection [, string $qualifier [, string $schema [, string $table-name [, string $column-name]]]])

テーブルのカラムおよび関連する権限情報を含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

テーブルを含むスキーマ。すべてのスキーマに一致させるには、 **NULL** あるいは空の文字列を渡します。

table-name

テーブルあるいはビューの名前。データベース内のすべてのテーブルに一致させるには、 **NULL** あるいは空の文字列を渡します。

column-name

カラムの名前。テーブル内のすべてのカラムに一致させるには、 **NULL** あるいは空の文字列を渡します。

返り値

指定したパラメータに一致するカラムの権限情報を含むステートメントリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	説明
TABLE_CAT	カタログの名前。テーブルがカタログを保持していない場合は NULL。
TABLE_SCHEM	スキーマの名前。
TABLE_NAME	テーブルあるいはビューの名前。
COLUMN_NAME	カラムの名前。
GRANTOR	その権限を与えたユーザの認証 ID。
GRANTEE	その権限を与えられたユーザの認証 ID。
PRIVILEGE	カラムの権限。
IS_GRANTABLE	GRANTEE が、この権限を他のユーザに与えることができるかどうか。

参考

- db2_columns()
- db2_foreign_keys()
- db2_primary_keys()
- db2_procedure_columns()
- db2_procedures()
- db2_special_columns()
- db2_statistics()
- db2_table_privileges()
- db2_tables()

---

db2_columns

(PECL ibm_db2:1.0-1.6.2)

db2_columns — テーブルのカラムおよび関連するメタデータを含む結果セットを返す

説明

resource **db2_columns** (resource $connection [, string $qualifier [, string $schema [, string $table-name [, string $column-name]]]])

テーブルのカラムおよび関連するメタデータを含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

テーブルを含むスキーマ。すべてのスキーマに一致させるには、 *'%'* を渡します。

table-name

テーブルあるいはビューの名前。データベース内のすべてのテーブルに一致させるには、 **NULL** あるいは空の文字列を渡します。

column-name

カラムの名前。テーブル内のすべてのカラムに一致させるには、 **NULL** あるいは空の文字列を渡します。

返り値

指定したパラメータに一致するカラムの情報を含む結果セットの ステートメントリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	説明
TABLE_CAT	カタログの名前。テーブルがカタログを保持していない場合は NULL。
TABLE_SCHEM	スキーマの名前。
TABLE_NAME	テーブルあるいはビューの名前。
COLUMN_NAME	カラムの名前。
DATA_TYPE	カラムの SQL データ型を整数値で表したもの。
TYPE_NAME	カラムのデータ型を文字列で表したもの。
COLUMN_SIZE	カラムのサイズを表す整数値。
BUFFER_LENGTH	このカラムのデータを保存するために必要な最大のバイト数。
DECIMAL_DIGITS	カラムの位取り。位取りが適用できない場合は **NULL**。
NUM_PREC_RADIX	*10* (正確な数値データ型を表す)、 *2* (概数データ型を表す)、 あるいは **NULL** (基数が適用できないデータ型を表す) のいずれか。
NULLABLE	カラムが null 値をとることができるかどうかを表す整数値。
REMARKS	カラムの説明。
COLUMN_DEF	カラムのデフォルト値。
SQL_DATA_TYPE	カラムのサイズを表す整数値。
SQL_DATETIME_SUB	datetime 型のコードを表す整数値、 あるいはこれが適用できない SQL データ型である場合に **NULL**。
CHAR_OCTET_LENGTH	文字型のカラムにおける最大のオクテット数。 シングルバイト文字セットのデータの場合、これは COLUMN_SIZE に一致します。文字型でないカラムの場合は **NULL** となります。
ORDINAL_POSITION	テーブル内でのカラムの位置を表す、1 から始まるインデックス。
IS_NULLABLE	カラムが null 値をとることができるかどうかを表す文字列。 'YES' の場合は null 値をとることができ、 'NO' の場合はできません。

参考

- db2_column_privileges()
- db2_foreign_keys()
- db2_primary_keys()
- db2_procedure_columns()
- db2_procedures()
- db2_special_columns()
- db2_statistics()
- db2_table_privileges()
- db2_tables()

---

db2_commit

(PECL ibm_db2:1.0-1.6.2)

db2_commit — トランザクションをコミットする

説明

bool **db2_commit** (resource $connection)

指定した接続リソース上で実行中のトランザクションをコミットし、 新しいトランザクションを開始します。PHP アプリケーションのデフォルトは AUTOCOMMIT モードなので、接続リソースに対して AUTOCOMMIT を無効にしていない限り **db2_commit()** は必要ありません。

注意: 指定した接続が持続的接続であった場合、持続的接続を使用している すべてのアプリケーションで実行中のトランザクションがコミットされます。 そのため、トランザクションが必要なアプリケーションでは 持続的接続の使用は推奨されません。

パラメータ

connection

db2_connect() あるいは db2_pconnect() が返した有効なデータベース接続リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- db2_autocommit()
- db2_rollback()

---

db2_conn_error

(PECL ibm_db2:1.0-1.6.2)

db2_conn_error — 直近の接続から返された SQLSTATE を含む文字列を返す

説明

string **db2_conn_error** ([resource $connection])

db2_conn_error() は、データベースへの直近の接続が 失敗した原因を表す SQLSTATE を返します。接続が失敗した場合には、 db2_connect() は **FALSE** を返します。 この場合、SQLSTATE の値を取得するために **db2_conn_error()** に何もパラメータを渡す必要はありません。

しかし、接続は成功したもののその後に無効になった場合は、 パラメータ *connection* を指定することで 特定の接続についての SQLSTATE を取得することができます。

SQLSTATE の値の意味を調べるには、DB2 コマンドラインプロセッサのプロンプトで 次のコマンドを発行します。 **db2 '? *sqlstate-value* '** また、db2_conn_errormsg() をコールして、 明示的なエラーメッセージと、それに関連する SQLCODE の値を取得することも可能です。

パラメータ

connection

当初は成功したものの、その後に無効になった接続に 関連付けられた接続リソース。

返り値

失敗した接続試行の結果を表す SQLSTATE を返します。 直近の接続試行に関連するエラーがない場合には空の文字列を返します。

例

Example#1 失敗した接続試行からの SQLSTATE の取得

以下の例では、db2_connect() に わざと無効なパラメータを渡し、SQLSTATE を取得しています。

```
<?php
$conn = db2_connect('badname', 'baduser', 'badpassword');
if (!$conn) {
    print "SQLSTATE の値: " . db2_conn_error();
}
?>
```

上の例の出力は以下となります。

```
SQLSTATE の値: 08001
```

参考

- db2_conn_errormsg()
- db2_connect()
- db2_stmt_error()
- db2_stmt_errormsg()

---

db2_conn_errormsg

(PECL ibm_db2:1.0-1.6.2)

db2_conn_errormsg — 直近の接続エラーメッセージおよび SQLCODE を返す

説明

string **db2_conn_errormsg** ([resource $connection])

db2_conn_errormsg() は、データベースへの直近の接続が 失敗した原因を表す SQLCODE およびエラーメッセージを返します。 接続が失敗した場合には、db2_connect() は **FALSE** を返します。 この場合、エラーメッセージおよび SQLSTATE の値を取得するために **db2_conn_errormsg()** に何もパラメータを渡す必要はありません。

しかし、接続は成功したもののその後に無効になった場合は、 パラメータ *connection* を指定することで 特定の接続についてのエラーメッセージおよび SQLSTATE を取得することができます。

パラメータ

connection

当初は成功したものの、その後に無効になった接続に 関連付けられた接続リソース。

返り値

失敗した接続試行の結果を表すエラーメッセージおよび SQLSTATE を返します。 直近の接続試行に関連するエラーがない場合には、**db2_conn_errormsg()** はn空の文字列を返します。

例

Example#1 失敗した接続試行からのエラーメッセージの取得

以下の例では、db2_connect() に わざと無効なパラメータを渡し、エラーメッセージおよび SQLSTATE を取得しています。

```
<?php
$conn = db2_connect('badname', 'baduser', 'badpassword');
if (!$conn) {
    print db2_conn_errormsg();
}
?>
```

上の例の出力は以下となります。

```
[IBM][CLI Driver] SQL1013N  The database alias name
or database name "BADNAME" could not be found.  SQLSTATE=42705
  SQLCODE=-1013
```

参考

- db2_conn_error()
- db2_connect()
- db2_stmt_error()
- db2_stmt_errormsg()

---

db2_connect

(PECL ibm_db2:1.0-1.6.2)

db2_connect — データベースへの接続を返す

説明

resource **db2_connect** (string $database , string $username , string $password [, array $options])

IBM DB2 Universal Database、IBM Cloudscape あるいは Apache Derby データベースへの新しい接続を作成します。

パラメータ

database

データベースへのカタログ接続の場合には、*database* は DB2 クライアントカタログ内でのデータベースエイリアスを表します。

データベースへの非カタログ接続の場合には、 *database* は以下のような形式の接続文字列を表します。

```
DATABASE=database
;HOSTNAME=hostname
;PORT=port
;PROTOCOL=TCPIP;UID=username
;PWD=password
;
```

それぞれのパラメータは以下の内容を表します。
database

データベースの名前。

hostname

データベースサーバのホスト名あるいは IP アドレス。

port

データベースが要求を待ち受ける TCP/IP ポート。

username

データベースに接続するユーザ名。

password

データベースに接続するパスワード。

username

データベースに接続するユーザ名。

カタログでない接続の場合は、**NULL** あるいは空の文字列を渡す必要があります。

password

データベースに接続するパスワード。

カタログでない接続の場合は、**NULL** あるいは空の文字列を渡す必要があります。

options

接続の振る舞いを指定する接続オプションの連想配列。 使用可能なキーは以下のとおりです。

autocommit

DB2_AUTOCOMMIT_ON を渡すと、 この接続ハンドルで自動コミットを有効にします。

DB2_AUTOCOMMIT_OFF を渡すと、 この接続ハンドルで自動コミットを無効にします。

DB2_ATTR_CASE

DB2_CASE_NATURAL を渡すと、 カラム名の大文字小文字を変換せずに返します。

DB2_CASE_LOWER を渡すと、 カラム名を小文字に変換して返します。

DB2_CASE_UPPER を渡すと、 カラム名を大文字に変換して返します。

CURSOR

DB2_FORWARD_ONLY を渡すと、 ステートメントリソースで前進のみのカーソルを使用します。 これはデフォルトのカーソル型であり、 すべてのデータベースサーバでサポートされています。

DB2_SCROLLABLE を渡すと、 ステートメントリソースでスクロール可能なカーソルを使用します。 このモードでは結果セット内の行へのランダムアクセスが可能となりますが、 現在は IBM DB2 Universal Database でしかサポートされていません。

以下の新しい i5/OS オプションは、ibm_db2 のバージョン 1.5.1 以降で使用可能です。 注意: それ以前のバージョンの ibm_db2 は、これらの新しい i5 オプションをサポートしていません。

i5_lib

未解決のファイル参照を解決する際に使用するデフォルトのライブラリを指定します。 システムのネーミングモードを使用している接続の場合は、これは無効です。

i5_naming

DB2_I5_NAMING_ON は、DB2 UDB CLI iSeries のシステムネーミングモードを有効にします。ファイルの識別の際、 区切り文字としてスラッシュ (/) を使用します。 識別されないファイルの解決には、ジョブのライブラリー覧を使用します。

DB2_I5_NAMING_OFF は、DB2 UDB CLI のデフォルトのネーミングモード (SQL ネーミング) を無効にします。 ファイルの識別の際、区切り文字としてピリオド (.) を使用します。 識別されないファイルの解決には、 デフォルトのライブラリあるいは現在のユーザ ID を使用します。

i5_commit

i5_commit 属性は、 **db2_connect()** の前に設定しなければなりません。 接続が確立された後にこの値が変更され、その接続がリモートデータソースに対するものだった場合は、 その接続ハンドルに対してもう一度 **db2_connect()** がコールされるまで変更は反映されません。

注意: php.ini の設定は *ibm_db2.i5_allow_commit ==0* あるいは *DB2_I5_TXN_NO_COMMIT* がデフォルトです。 しかし、その設定よりも *i5_commit* オプションの内容のほうが優先されます。

DB2_I5_TXN_NO_COMMIT - コミットの管理を使用しません。

DB2_I5_TXN_READ_UNCOMMITTED - ダーティリード、 反復不能読み取り、ファントムリードが発生する可能性があります。

DB2_I5_TXN_READ_COMMITTED - ダーティリードは発生しません。 反復不能読み取り、ファントムリードが発生する可能性があります。

DB2_I5_TXN_REPEATABLE_READ - ダーティリード、 反復不能読み取りは発生しません。 ファントムリードが発生する可能性があります。

DB2_I5_TXN_SERIALIZABLE - トランザクションの一貫性を保持します。 ダーティリード、反復不能読み取り、ファントムリードは発生しません。

i5_query_optimize

DB2_FIRST_IO すべてのクエリは、結果の一行目をできるだけ早く返すように最適化されます。 これが有効なのは、出力の制御をユーザが行う場合です。出力の制御とは、 たとえば、出力の最初のページを見てそこで処理を停止させるなどのことです。 OPTIMIZE FOR nnn ROWS 句を指定したクエリは、その指定が優先されます。

DB2_ALL_IO すべてのクエリは、結果全体をできるだけ短時間で取得できるように最適化されます。 これは、結果をファイルや帳票に書き出す場合に有用なオプションです。 あるいは出力データを順次処理するインターフェイスなどにも有用です。 OPTIMIZE FOR nnn ROWS 句を指定したクエリは、その指定が優先されます。 これはデフォルトの設定です。

i5_dbcs_alloc

DB2_I5_DBCS_ALLOC_ON は、 DBCS トランザクションカラムサイズ用の DB2 6X アロケーションを有効にします。

DB2_I5_DBCS_ALLOC_OFF は、 DBCS トランザクションカラムサイズ用の DB2 6X アロケーションを無効にします。

注意: php.ini では *ibm_db2.i5_dbcs_alloc ==0* あるいは *DB2_I5_DBCS_ALLOC_OFF* がデフォルト設定となっています。 しかし、*i5_dbcs_alloc* の値のほうが優先されます。

i5_date_fmt

SQL_FMT_ISO - 国際標準化機構 (ISO) の日付書式 yyyy-mm-dd を使用します。デフォルトです。

DB2_I5_FMT_USA - 合衆国の日付書式 mm/dd/yyyy を使用します。

DB2_I5_FMT_EUR - 欧州の日付書式 format dd.mm.yyyy を使用します。

DB2_I5_FMT_JIS - 日本工業規格 (JIS) の日付書式 yyyy-mm-dd を使用します。

DB2_I5_FMT_MDY - 日付書式 mm/dd/yyyy を使用します。

DB2_I5_FMT_DMY - 日付書式 dd/mm/yyyy を使用します。

DB2_I5_FMT_YMD - 日付書式 yy/mm/dd を使用します。

DB2_I5_FMT_JUL - ユリウス日 yy/ddd を使用します。

DB2_I5_FMT_JOB - そのジョブのデフォルトを使用します。

i5_date_sep

DB2_I5_SEP_SLASH - スラッシュ (/) を日付の区切り文字として使用します。 デフォルトです。

DB2_I5_SEP_DASH - ダッシュ (-) を日付の区切り文字として使用します。

DB2_I5_SEP_PERIOD - ピリオド (.) を日付の区切り文字として使用します。

DB2_I5_SEP_COMMA - カンマ (,) を日付の区切り文字として使用します。

DB2_I5_SEP_BLANK - ブランクを日付の区切り文字として使用します。

DB2_I5_SEP_JOB - そのジョブのデフォルトを使用します。

i5_time_fmt

DB2_I5_FMT_ISO - 国際標準化機構 (ISO) の時刻書式 hh.mm.ss を使用します。デフォルトです。

DB2_I5_FMT_USA - 合衆国の時刻書式 hh:mmxx を使用します。xx には AM あるいは PM が入ります。

DB2_I5_FMT_EUR - 欧州の時刻書式 hh.mm.ss を使用します。

DB2_I5_FMT_JIS - 日本工業規格 (JIS) の時刻書式 hh:mm:ss を使用します。

DB2_I5_FMT_HMS - hh:mm:ss を使用します。

i5_time_sep

DB2_I5_SEP_COLON - コロン (:) を時刻の区切り文字として使用します。 デフォルトです。

DB2_I5_SEP_PERIOD - ピリオド (.) を時刻の区切り文字として使用します。

DB2_I5_SEP_COMMA - カンマ (,) を時刻の区切り文字として使用します。

DB2_I5_SEP_BLANK - ブランクを時刻の区切り文字として使用します。

DB2_I5_SEP_JOB - そのジョブのデフォルトを使用します。

i5_decimal_sep

DB2_I5_SEP_PERIOD - ピリオド (.) を小数点として使用します。 デフォルトです。

DB2_I5_SEP_COMMA - カンマ (,) を小数点として使用します。

DB2_I5_SEP_JOB - そのジョブのデフォルトを使用します。

返り値

接続に成功した場合は接続ハンドルリソースを返します。 接続に失敗した場合は、**db2_connect()** は **FALSE** を返します。

例

Example#1 カタログ接続の作成

カタログ接続を行うには、DB2 コマンドラインプロセッサ (CLP) あるいは DB2 Configuration Assistant を使用して 事前に対象データベースをカタログしておく必要があります。

```
<?php
$database = 'SAMPLE';
$user = 'db2inst1';
$password = 'ibmdb2';

$conn = db2_connect($database, $user, $password);

if ($conn) {
    echo "接続に成功しました。";
    db2_close($conn);
}
else {
    echo "接続に失敗しました。";
}
?>
```

上の例の出力は以下となります。

接続に成功しました。

Example#2 非カタログ接続の作成

非カタログ接続の場合は、データベースに動的に接続することができます。

```
<?php
$database = 'SAMPLE';
$user = 'db2inst1';
$password = 'ibmdb2';
$hostname = 'localhost';
$port = 50000;

$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;" .
  "HOSTNAME=$hostname;PORT=$port;PROTOCOL=TCPIP;UID=$user;PWD=$password;";
$conn = db2_connect($conn_string, '', '');

if ($conn) {
    echo "接続に成功しました。";
    db2_close($conn);
}
else {
    echo "接続に失敗しました。";
}
?>
```

上の例の出力は以下となります。

```
接続に成功しました。
```

Example#3 自動コミットをデフォルトで無効にした接続の作成

オプションの配列を **db2_connect()** に渡すことで、接続ハンドルのデフォルトの振る舞いを変更できます。

```
<?php
$database = 'SAMPLE';
$user = 'db2inst1';
$password = 'ibmdb2';
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF);

$conn = db2_connect($database, $user, $password, $options);

if ($conn) {
    echo "接続に成功しました。\n";
    if (db2_autocommit($conn)) {
        echo "自動コミットが有効です。\n";
    }
    else {
        echo "自動コミットは無効です。\n";
    }
    db2_close($conn);
}
else {
    echo "接続に失敗しました。";
}
?>
```

上の例の出力は以下となります。

```
接続に成功しました。
自動コミットは無効です。
```

Example#4 i5/OS best performance

i5/OS で最高のパフォーマンスを引き出すためには、ibm_db2 1.5.1 を使用した PHP アプリケーションで **db2_connect()** にデフォルトのホスト、ユーザ ID およびパスワードを使用します。

```
<?php
   $library = "ADC";
  $i5 = db2_connect("", "", "", array("i5_lib"=>"qsys2"));
  $result = db2_exec($i5,
       "select * from systables where table_schema = '$library'");
  while ($row = db2_fetch_both($result)) {
     echo $row['TABLE_NAME']."</br>";
  }
  db2_close($i5);
?>
```

上の例の出力は以下となります。

```
ANIMALS
NAMES
PICTURES
```

参考

- db2_close()
- db2_pconnect()

---

db2_cursor_type

(PECL ibm_db2:1.0-1.6.2)

db2_cursor_type — ステートメントリソースが使用しているカーソルの型を返す

説明

int **db2_cursor_type** (resource $stmt)

ステートメントリソースが使用しているカーソルの型を返します。 使用中のカーソルが前進のみなのかスクロール可能なのかを調べるために、 これを使用します。

パラメータ

stmt

有効なステートメントリソース。

返り値

ステートメントリソースが前進のみのカーソルを使用している場合に *DB2_FORWARD_ONLY*、スクロール可能なカーソルを 使用している場合に *DB2_SCROLLABLE* を返します。

参考

- db2_prepare()

---

db2_escape_string

(PECL ibm_db2:1.6.0-1.6.2)

db2_escape_string — 特定の文字をエスケープする

説明

string **db2_escape_string** (string $string_literal)

引数で指定した文字列中の特殊文字の前にバックスラッシュを追加します。

パラメータ

string_literal

変更すべき特殊文字を含む文字列。 *¥x00*、 *¥n*、*¥r*、*¥*、 *'*、*"* および *¥x1a* について、その前にバックスラッシュを追加します。

返り値

string_literal の中の上で挙げた文字について、 その前にバックスラッシュを追加した文字列を返します。

例

Example#1 db2_escape_string() の例

db2_escape_string() 関数を使用した結果

```
<?php

$conn = db2_connect($database, $user, $password);

if ($conn) {
    $str[0] = "All characters: ¥x00 , ¥n , ¥r , ¥ , ' , ¥" , ¥x1a .";
    $str[1] = "Backslash (¥). Single quote ('). Double quote (¥")";
    $str[2] = "The NULL character ¥0 must be quoted as well";
    $str[3] = "Intersting characters: ¥x1a , ¥x00 .";
    $str[4] = "Nothing to quote";
    $str[5] = 200676;
    $str[6] = "";

    foreach( $str as $string ) {
        echo "db2_escape_string: " . db2_escape_string($string). "¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
db2_escape_string: All characters: ¥0 , ¥n , ¥r , ¥¥ , ¥' , ¥" , ¥Z .
db2_escape_string: Backslash (¥¥). Single quote (¥'). Double quote (¥")
db2_escape_string: The NULL character ¥0 must be quoted as well
db2_escape_string: Intersting characters: ¥Z , ¥0 .
db2_escape_string: Nothing to quote
db2_escape_string: 200676
db2_escape_string:
```

参考

- db2_prepare()

---

db2_exec

(PECL ibm_db2:1.0-1.6.2)

db2_exec — SQL 文を直接実行する

説明

```
resource db2_exec ( resource $connection , string $statement [, array $options ] )
```

SQL 文を直接実行します。

PHP 変数の内容を SQL 文に組み込みたい場合は、この関数を使用すると 典型的なセキュリティ上の問題を引き起こしかねないことを知っておきましょう。 db2_prepare() をコールして入力パラメータつきの SQL 文を準備することを検討してください。その後で db2_execute() をコールして入力値を渡すことで、 SQL インジェクション攻撃を避けることができます。

同じ SQL 文にさまざまなパラメータを指定して何度も発行する場合は、 db2_prepare() および db2_execute() の使用を検討してください。 これにより、データベースサーバが実行計画を再利用することができて データベースアクセスの効率が向上します。

パラメータ

connection

db2_connect() あるいは db2_pconnect() が返した有効なデータベース接続リソース。

statement

SQL 文。パラメータマーカを含めることはできません。

options

文のオプションを含む連想配列。 データベースサーバがその機能をサポートしている場合に、 このパラメータを使用してスクロール可能なカーソルの使用を 要求することができます。

cursor

DB2_FORWARD_ONLY を渡すと、 この SQL 文で前進のみのカーソルを使用することを要求します。 これはデフォルトのカーソル型であり、すべてのデータベースサーバで サポートされています。また、スクロール可能なカーソルに比べて 非常に高速になります。

DB2_SCROLLABLE を渡すと、 この SQL 文でスクロール可能なカーソルを使用することを要求します。 このカーソル型を使用すると、データベースサーバから 行の並び順を気にせずにデータを取得できるようになります。 しかし、この型は DB2 サーバでしかサポートされておらず、前進のみのカーソルに比べて非常に低速です。

返り値

SQL 文の実行に成功した場合にステートメントリソース、 SQL 文の実行に失敗した場合に **FALSE** を返します。

例

Example#1 db2_exec() でのテーブルの作成

以下の例では、**db2_exec()** を使用して テーブルを作成する DDL 文を発行します。

```
<?php
$conn = db2_connect($database, $user, $password);

// テストテーブルを作成します
$create = 'CREATE TABLE animals (id INTEGER, breed VARCHAR(32),
     name CHAR(16), weight DECIMAL(7,2))';
$result = db2_exec($conn, $create);
if ($result) {
    print "テーブルの作成に成功しました。\n";
}

// テストテーブルに値を投入します
$animals = array(
     array(0, 'cat', 'Pook', 3.2),
     array(1, 'dog', 'Peaches', 12.3),
     array(2, 'horse', 'Smarty', 350.0),
     array(3, 'gold fish', 'Bubbles', 0.1),
     array(4, 'budgerigar', 'Gizmo', 0.2),
     array(5, 'goat', 'Rickety Ride', 9.7),
     array(6, 'llama', 'Sweater', 150)
);

foreach ($animals as $animal) {
    $rc = db2_exec($conn, "INSERT INTO animals (id, breed, name, weight)
         VALUES ({$animal[0]}, '{$animal[1]}', '{$animal[2]}', {$animal[3]})");
     if ($rc) {
         print "Insert... ";
     }
}
?>
```

上の例の出力は以下となります。

```
テーブルの作成に成功しました。
Insert...  Insert...  Insert...  Insert...  Insert...  Insert...  Insert...
```

Example#2 スクロール可能なカーソルでの SELECT 文の実行

以下の例では、**db2_exec()** で発行された SQL 文にスクロール可能なカーソルを要求する方法を説明します。

```
<?php
$conn = db2_connect($database, $user, $password);
$sql = "SELECT name FROM animals
     WHERE weight < 10.0
     ORDER BY name";
if ($conn) {
     require_once('prepare.inc');
    $stmt = db2_exec($conn, $sql, array('cursor' => DB2_SCROLLABLE));
    while ($row = db2_fetch_array($stmt)) {
        print "$row[0]¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
Bubbles
Gizmo
Pook
Rickety Ride
```

Example#3 XML データを SQL の結果セットとして返す

次の例は、XML カラムに格納されたデータの扱い方を、SAMPLE データベースを用いて説明するものです。ごく単純な SQL/XML を使用して、この例では XML ドキュメントのいくつかのノードを みなさんおなじみの SQL 結果セット形式で返します。

```
<?php

$conn = db2_connect("SAMPLE", "db2inst1", "ibmdb2");

$query = 'SELECT * FROM XMLTABLE(
       XMLNAMESPACES (DEFAULT ¥'http://posample.org¥'),
        ¥'db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo¥'
     COLUMNS
      "CID" VARCHAR (50) PATH ¥'@Cid¥',
      "NAME" VARCHAR (50) PATH ¥'name¥',
      "PHONE" VARCHAR (50) PATH ¥'phone [ @type = "work"]¥'
     ) AS T
     WHERE NAME = ¥'Kathy Smith¥'
    ';
$stmt = db2_exec($conn, $query);

while($row = db2_fetch_object($stmt)){
    printf("$row->CID       $row->NAME       $row->PHONE¥n");
}
db2_close($conn);

?>
```

上の例の出力は以下となります。

```
1000       Kathy Smith       416-555-1358
1001       Kathy Smith       905-555-7258
```

Example#4 XML データの "JOIN"

次の例は、SAMPLE データベースの 2 つの異なる XML カラムに格納されたドキュメントを使用します。 まず、2 つの異なるカラムの XML ドキュメントをもとにして 2 つのテンポラリテーブルを作成します。 そして特定の顧客の配送情報を SQL 結果セットで返します。

```
<?php

$conn = db2_connect("SAMPLE", "db2inst1", "ibmdb2");

$query = '
      SELECT A.CID, A.NAME, A.PHONE, C.PONUM, C.STATUS
     FROM
      XMLTABLE(
       XMLNAMESPACES (DEFAULT ¥'http://posample.org¥'),
        ¥'db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo¥'
      COLUMNS
       "CID" BIGINT PATH ¥'@Cid¥',
       "NAME" VARCHAR (50) PATH ¥'name¥',
       "PHONE" VARCHAR (50) PATH ¥'phone [ @type = "work"]¥'
      ) as A,
      PURCHASEORDER AS B,
      XMLTABLE (
       XMLNAMESPACES (DEFAULT ¥'http://posample.org¥'),
        ¥'db2-fn:xmlcolumn("PURCHASEORDER.PORDER")/PurchaseOrder¥'
      COLUMNS
       "PONUM"   BIGINT PATH ¥'@PoNum¥',
       "STATUS" VARCHAR (50) PATH ¥'@Status¥'
      ) as C
      WHERE A.CID = B.CUSTID AND
      B.POID = C.PONUM AND
```

```
     A.NAME = ¥'Kathy Smith¥'
';

$stmt = db2_exec($conn, $query);

while($row = db2_fetch_object($stmt)){
    printf("$row->CID       $row->NAME       $row->PHONE       $row->PONUM       $row->STATUS¥n");
}

db2_close($conn);

?>
```

上の例の出力は以下となります。

```
1001       Kathy Smith       905-555-7258       5002       Shipped
```

Example#5 大きな XML ドキュメントの一部を SQL データとして返す

次の例は、SAMPLE データベースのドキュメント PRODUCT.DESCRIPTION の一部を使用します。これは、商品の説明 (XML データ) および価格情報 (SQL データ) を含む XML ドキュメントを返します。

```
<?php

$conn = db2_connect("SAMPLE", "db2inst1", "ibmdb2");

$query = '
SELECT
XMLSERIALIZE(
XMLQUERY(¥'
      declare boundary-space strip;
      declare default element namespace "http://posample.org";
     <promoList> {
     for $prod in $doc/product
     where $prod/description/price < 10.00
     order by $prod/description/price ascending
     return(
         <promoitem> {
         $prod,
           <startdate> {$start} </startdate>,
           <enddate> {$end} </enddate>,
           <promoprice> {$promo} </promoprice>
         } </promoitem>
    )
     } </promoList>
¥' passing by ref DESCRIPTION AS "doc",
PROMOSTART as "start",
PROMOEND as "end",
PROMOPRICE as "promo"
RETURNING SEQUENCE)
AS CLOB (32000))
AS NEW_PRODUCT_INFO
FROM PRODUCT
WHERE PID = ¥'100-100-01¥'
';

$stmt = db2_exec($conn, $query);

while($row = db2_fetch_array($stmt)){
    printf("$row[0]¥n");
}
db2_close($conn);

?>
?>
```

上の例の出力は以下となります。

```
<promoList xmlns="http://posample.org">
     <promoitem>
      <product pid="100-100-01">
          <description>
               <name>Snow Shovel, Basic 22 inch</name>
                <details>Basic Snow Shovel, 22 inches wide, straight handle with D-Grip</details>
              <price>9.99</price>
              <weight>1 kg</weight>
          </description>
     </product>
      <startdate>2004-11-19</startdate>
      <enddate>2004-12-19</enddate>
      <promoprice>7.25</promoprice>
     </promoitem>
</promoList>
```

参考

- db2_execute()
- db2_prepare()

---

db2_execute

(PECL ibm_db2:1.0-1.6.2)

db2_execute — プリペアドステートメントを実行する

説明

bool **db2_execute** (resource $stmt [, array $parameters])

db2_execute() は、 db2_prepare() で準備された SQL 文を実行します。

SQL 文が結果セットを返す場合、例えば SELECT 文であったり 結果セットを返すストアドプロシージャの CALL であったりした場合には、 *stmt* から結果の行を配列として取得することができます。 取得には db2_fetch_assoc()、 db2_fetch_both() あるいは db2_fetch_array() を使用します。 あるいは、db2_fetch_row() を使用して 結果セットポインタを次の行に移動させ、 db2_result() で行の内容をその都度取得することも可能です。

db2_exec() の代わりに db2_prepare() および **db2_execute()** を使用することの利点については、 db2_prepare() での簡単な説明を参照ください。

パラメータ

stmt

db2_prepare() が返すプリペアドステートメント。

parameters

プリペアドステートメント内に含まれるパラメータマーカに対応する、 入力パラメータの配列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 パラメータマーカを使用した SQL 文の準備と実行

以下の例では、4 つのパラメータマーカを含む INSERT 文を準備し、 入力値の配列を含む配列を順に処理しながら **db2_execute()** に値を渡します。

```
<?php
$pet = array(0, 'cat', 'Pook', 3.2);

$insert = 'INSERT INTO animals (id, breed, name, weight)
     VALUES (?, ?, ?, ?)';

$stmt = db2_prepare($conn, $insert);
if ($stmt) {
    $result = db2_execute($stmt, $pet);
    if ($result) {
        print "新しいペットの追加に成功しました。";
    }
}
?>
```

上の例の出力は以下となります。

新しいペットの追加に成功しました。

Example#2 OUT パラメータを使用したストアドプロシージャのコール

以下の例では、まずひとつの OUT パラメータをパラメータマーカで受け取る CALL 文を準備し、db2_bind_param() を使用して PHP 変数 *$my_pets* をそのパラメータにバインドし、 **db2_execute()** で CALL 文を実行します。 ストアドプロシージャの CALL 終了後は、*$num_pets* にはストアドプロシージャの OUT パラメータに返された値が反映されています。

```
<?php
$num_pets = 0;
$res = db2_prepare($conn, "CALL count_my_pets(?)");
$rc = db2_bind_param($res, 1, "num_pets", DB2_PARAM_OUT);
$rc = db2_execute($res);
print "私の飼っているペットの数は $num_pets です!";
?>
```

上の例の出力は以下となります。

私の飼っているペットの数は 7 です!

Example#3 XML データを SQL の結果セットとして返す

次の例は、XML カラムに格納されたデータの扱い方を、SAMPLE データベースを用いて説明するものです。ごく単純な SQL/XML を使用して、この例では XML ドキュメントのいくつかのノードを みなさんおなじみの SQL 結果セット形式で返します。

```
<?php

$conn = db2_connect("SAMPLE", "db2inst1", "ibmdb2");

$query = 'SELECT * FROM XMLTABLE(
      XMLNAMESPACES (DEFAULT ¥'http://posample.org¥'),
       ¥'db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo¥'
```

```
     COLUMNS
      "CID" VARCHAR (50) PATH ¥'@Cid¥',
      "NAME" VARCHAR (50) PATH ¥'name¥',
      "PHONE" VARCHAR (50) PATH ¥'phone [ @type = "work"]¥'
     ) AS T
     WHERE NAME = ?
     ';

$stmt = db2_prepare($conn, $query);

$name = 'Kathy  Smith';

if ($stmt) {
    db2_bind_param($stmt, 1, "name", DB2_PARAM_IN);
    db2_execute($stmt);

     while($row = db2_fetch_object($stmt)){
    printf("$row->CID       $row->NAME       $row->PHONE¥n");
    }
}
db2_close($conn);

?>
```

上の例の出力は以下となります。

```
1000       Kathy Smith       416-555-1358
1001       Kathy Smith       905-555-7258
```

Example#4 XML データの "JOIN"

次の例は、SAMPLE データベースの 2 つの異なる XML カラムに格納されたドキュメントを使用します。 まず、2 つの異なるカラムの XML ドキュメントをもとにして 2 つのテンポラリテーブルを作成します。 そして特定の顧客の配送情報を SQL 結果セットで返します。

```
<?php

$conn = db2_connect("SAMPLE", "db2inst1", "ibmdb2");

$query = '
SELECT A.CID, A.NAME, A.PHONE, C.PONUM, C.STATUS
FROM
XMLTABLE(
XMLNAMESPACES (DEFAULT ¥'http://posample.org¥'),
¥'db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo¥'
COLUMNS
"CID" BIGINT PATH ¥'@Cid¥',
"NAME" VARCHAR (50) PATH ¥'name¥',
"PHONE" VARCHAR (50) PATH ¥'phone [ @type = "work"]¥'
) as A,
PURCHASEORDER AS B,
XMLTABLE (
XMLNAMESPACES (DEFAULT ¥'http://posample.org¥'),
¥'db2-fn:xmlcolumn("PURCHASEORDER.PORDER")/PurchaseOrder¥'
COLUMNS
"PONUM"   BIGINT PATH ¥'@PoNum¥',
"STATUS" VARCHAR (50) PATH ¥'@Status¥'
) as C
WHERE A.CID = B.CUSTID AND
     B.POID = C.PONUM AND
     A.NAME = ?
';

$stmt = db2_prepare($conn, $query);

$name = 'Kathy  Smith';

if ($stmt) {
    db2_bind_param($stmt, 1, "name", DB2_PARAM_IN);
    db2_execute($stmt);

     while($row = db2_fetch_object($stmt)){
    printf("$row->CID       $row->NAME       $row->PHONE       $row->PONUM       $row->STATUS¥n");
    }
}

db2_close($conn);

?>
```

上の例の出力は以下となります。

```
1001       Kathy Smith       905-555-7258       5002       Shipped
```

Example#5 大きな XML ドキュメントの一部を SQL データとして返す

次の例は、SAMPLE データベースのドキュメント PRODUCT.DESCRIPTION の一部を使用します。これは、商品の説明 (XML データ) および価格情報 (SQL データ) を含む XML ドキュメントを返します。

```
<?php

$conn = db2_connect("SAMPLE", "db2inst1", "ibmdb2");

$query = '
SELECT
```

```
XMLSERIALIZE(
XMLQUERY(¥'
      declare boundary-space strip;
      declare default element namespace "http://posample.org";
     <promoList> {
     for $prod in $doc/product
     where $prod/description/price < 10.00
     order by $prod/description/price ascending
     return(
          <promoitem> {
         $prod,
           <startdate> {$start} </startdate>,
           <enddate> {$end} </enddate>,
           <promoprice> {$promo} </promoprice>
              } </promoitem>
    )
     } </promoList>
¥' passing by ref DESCRIPTION AS "doc",
PROMOSTART as "start",
PROMOEND as "end",
PROMOPRICE as "promo"
RETURNING SEQUENCE)
AS CLOB (32000))
AS NEW_PRODUCT_INFO
FROM PRODUCT
WHERE PID = ?
';

$stmt = db2_prepare($conn, $query);

$pid = "100-100-01";

if ($stmt) {
    db2_bind_param($stmt, 1, "pid", DB2_PARAM_IN);
    db2_execute($stmt);

     while($row = db2_fetch_array($stmt)){
    printf("$row[0]¥n");
    }
}

db2_close($conn);

?>
```

上の例の出力は以下となります。

```
<promoList xmlns="http://posample.org">
     <promoitem>
      <product pid="100-100-01">
          <description>
               <name>Snow Shovel, Basic 22 inch</name>
                 <details>Basic Snow Shovel, 22 inches wide, straight handle with D-Grip</details>
              <price>9.99</price>
              <weight>1 kg</weight>
          </description>
     </product>
       <startdate>2004-11-19</startdate>
       <enddate>2004-12-19</enddate>
       <promoprice>7.25</promoprice>
     </promoitem>
</promoList>
```

参考

- db2_exec()
- db2_fetch_array()
- db2_fetch_assoc()
- db2_fetch_both()
- db2_fetch_row()
- db2_prepare()
- db2_result()

---

db2_fetch_array

(PECL ibm_db2:1.0.1-1.6.2)

db2_fetch_array — 結果セット内の行を表す、カラム位置をインデックスとする配列を返す

説明

array **db2_fetch_array** (resource $stmt [, int $row_number])

結果セット内の行を表す、カラム位置をインデックスとする配列を返します。 インデックスは 0 から始まります。

パラメータ

stmt

結果セットを含む有効な *stmt* リソース。

row_number

結果セット内の 1 から始まる行番号を指定します。 結果セットで前進のみのカーソルを使用している場合にこのパラメータを渡すと、 PHP の警告が発生します。

返り値

結果セットの次の行あるいは要求した行のデータを表す配列を返します。 配列の 0 から始まるインデックスが、カラムの位置を表します。 結果セットに行がもうない場合、あるいは *row_number* で指定された行が結果セットに存在しない場合に **FALSE** を返します。

例

Example#1 前進のみのカーソルを使用して順に処理する

行番号を指定せずに **db2_fetch_array()** をコールすると、 自動的に結果セットの次の行を取得します。

```
<?php

$sql = "SELECT id, name, breed, weight FROM animals ORDER BY breed";
$stmt = db2_prepare($conn, $sql);
$result = db2_execute($stmt);

while ($row = db2_fetch_array($stmt)) {
    printf ("%-5d %-16s %-32s %10s¥n",
        $row[0], $row[1], $row[2], $row[3]);
}
?>
```

上の例の出力は以下となります。

```
0     Pook              cat                                3.20
5     Rickety Ride      goat                               9.70
2     Smarty            horse                            350.00
```

Example#2 スクロール可能なカーソルから、指定した行を db2_fetch_array() で取得する

スクロール可能なカーソルを使用している場合は、行番号を指定して **db2_fetch_array()** をコールすることができます。 次の例は、結果セットの 2 行目から始めて 1 行おきにデータを取得します。

```
<?php

$sql = "SELECT id, name, breed, weight FROM animals ORDER BY breed";
$result = db2_exec($stmt, $sql, array('cursor' => DB2_SCROLLABLE));

$i=2;
while ($row = db2_fetch_array($result, $i)) {
    printf ("%-5d %-16s %-32s %10s¥n",
        $row[0], $row[1], $row[2], $row[3]);
    $i = $i + 2;
}
?>
```

上の例の出力は以下となります。

```
0     Pook              cat                                3.20
5     Rickety Ride      goat                               9.70
2     Smarty            horse                            350.00
```

参考

- db2_fetch_assoc()
- db2_fetch_both()
- db2_fetch_object()
- db2_fetch_row()
- db2_result()

---

db2_fetch_assoc

(PECL ibm_db2:1.0-1.6.2)

db2_fetch_assoc — 結果セット内の行を表す、カラム名をインデックスとする配列を返す

説明

```
array db2_fetch_assoc ( resource $stmt [, int $row_number ] )
```

結果セット内の行を表す、カラム名をインデックスとする配列を返します。

パラメータ

stmt

結果セットを含む有効な *stmt* リソース。

row_number

結果セット内の 1 から始まる行番号を指定します。 結果セットで前進のみのカーソルを使用している場合にこのパラメータを渡すと、 PHP の警告が発生します。

返り値

結果セットの次の行あるいは要求した行のデータを表す、 カラム名をインデックスとした連想配列を返します。 結果セットに行がもうない場合、あるいは *row_number* で指定された行が結果セットに存在しない場合に **FALSE** を返します。

例

Example#1 前進のみのカーソルを使用して順に処理する

行番号を指定せずに **db2_fetch_assoc()** をコールすると、 自動的に結果セットの次の行を取得します。

```
<?php

$sql = "SELECT id, name, breed, weight FROM animals ORDER BY breed";
$stmt = db2_prepare($conn, $sql);
$result = db2_execute($stmt);

while ($row = db2_fetch_assoc($stmt)) {
    printf ("%-5d %-16s %-32s %10s¥n",
        $row['ID'], $row['NAME'], $row['BREED'], $row['WEIGHT']);
}
?>
```

上の例の出力は以下となります。

```
0     Pook              cat                                3.20
5     Rickety Ride      goat                               9.70
2     Smarty            horse                            350.00
```

Example#2 スクロール可能なカーソルから、指定した行を db2_fetch_assoc() で取得する

スクロール可能なカーソルを使用している場合は、行番号を指定して **db2_fetch_assoc()** をコールすることができます。 次の例は、結果セットの 2 行目から始めて 1 行おきにデータを取得します。

```
<?php

$sql = "SELECT id, name, breed, weight FROM animals ORDER BY breed";
$result = db2_exec($stmt, $sql, array('cursor' => DB2_SCROLLABLE));

$i=2;
while ($row = db2_fetch_assoc($result, $i)) {
    printf ("%-5d %-16s %-32s %10s¥n",
        $row['ID'], $row['NAME'], $row['BREED'], $row['WEIGHT']);
    $i = $i + 2;
}
?>
```

上の例の出力は以下となります。

```
0     Pook              cat                                3.20
5     Rickety Ride      goat                               9.70
2     Smarty            horse                            350.00
```

参考

- db2_fetch_array()
- db2_fetch_both()
- db2_fetch_object()
- db2_fetch_row()
- db2_result()

---

db2_fetch_both

(PECL ibm_db2:1.0-1.6.2)

db2_fetch_both — 結果セット内の行を表す、 カラム位置およびカラム名の両方をインデックスとする配列を返す

説明

```
array db2_fetch_both ( resource $stmt [, int $row_number ] )
```

結果セット内の行を表す、 カラム位置およびカラム名の両方をインデックスとする配列を返します。 **db2_fetch_both()** が返す行は、 一種類のインデックスしか返さない db2_fetch_assoc() あるいは db2_fetch_array() よりもメモリを消費することに注意しましょう。

パラメータ

stmt

結果セットを含む有効な *stmt* リソース。

row_number

結果セット内の 1 から始まる行番号を指定します。 結果セットで前進のみのカーソルを使用している場合にこのパラメータを渡すと、 PHP の警告が発生します。

返り値

結果セットの次の行あるいは要求した行のデータを表す、 カラム名および（0 から始まる）カラム番号の両方をインデックスとした連想配列を返します。 結果セットに行がもうない場合、あるいは *row_number* で指定された行が結果セットに存在しない場合に **FALSE** を返します。

例

Example#1 前進のみのカーソルを使用して順に処理する

行番号を指定せずに **db2_fetch_both()** をコールすると、 自動的に結果セットの次の行を取得します。次の例では、 返された配列に対してカラム名および数値インデックスの両方でアクセスします。

```
<?php

$sql = "SELECT id, name, breed, weight FROM animals ORDER BY breed";
$stmt = db2_prepare($conn, $sql);
$result = db2_execute($stmt);

while ($row = db2_fetch_both($stmt)) {
    printf ("%-5d %-16s %-32s %10s¥n",
        $row['ID'], $row[0], $row['BREED'], $row[3]);
}
?>
```

上の例の出力は以下となります。

```
0     Pook              cat                              3.20
5     Rickety Ride      goat                             9.70
2     Smarty            horse                          350.00
```

Example#2 スクロール可能なカーソルから、指定した行を db2_fetch_both() で取得する

スクロール可能なカーソルを使用している場合は、行番号を指定して **db2_fetch_both()** をコールすることができます。 次の例は、結果セットの 2 行目から始めて 1 行おきにデータを取得します。

```
<?php

$sql = "SELECT id, name, breed, weight FROM animals ORDER BY breed";
$result = db2_exec($stmt, $sql, array('cursor' => DB2_SCROLLABLE));

$i=2;
while ($row = db2_fetch_both($result, $i)) {
    printf ("%-5d %-16s %-32s %10s¥n",
        $row[0], $row['NAME'], $row[2], $row['WEIGHT']);
    $i = $i + 2;
}
?>
```

上の例の出力は以下となります。

```
0     Pook              cat                              3.20
5     Rickety Ride      goat                             9.70
2     Smarty            horse                          350.00
```

参考

- db2_fetch_array()
- db2_fetch_assoc()
- db2_fetch_object()
- db2_fetch_row()
- db2_result()

---

db2_fetch_object

(PECL ibm_db2:1.0-1.6.2)

db2_fetch_object — 結果セット内の行を表す、カラムをプロパティとするオブジェクトを返す

説明

object **db2_fetch_object** (resource $stmt [, int $row_number])

結果セット内の行を表す、カラムをプロパティとするオブジェクトを返します。

パラメータ

stmt

結果セットを含む有効な *stmt* リソース。

row_number

結果セット内の 1 から始まる行番号を指定します。 結果セットで前進のみのカーソルを使用している場合にこのパラメータを渡すと、 PHP の警告が発生します。

返り値

結果セット内の行を表すオブジェクトを返します。 オブジェクトのプロパティが、結果セット内のカラム名に対応します。

IBM DB2、Cloudscape および Apache Derby データベースサーバは、 通常はカラム名を大文字として扱います。そのため、 オブジェクトのプロパティも同じようになります。

SELECT 文の中でスカラ関数をコールすることでカラムの値を変更している場合、 データベースサーバは、そのカラムの名前としてカラム番号を返します。 もし何らかの意味のある名前をオブジェクトのプロパティとして使用したいのなら、 AS 句を使用して結果セット内のカラムに名前を割り当てる必要があります。

行が取得されなかった場合は **FALSE** を返します。

例

Example#1 db2_fetch_object() の例

次の例では、スカラ関数 RTRIM を使用した SELECT 文を発行します。 この関数は、カラムの最後の空白を取り除きます。プロパティ "BREED" および "2" を持つオブジェクトを作成するのではなく、 SELECT 文で AS 句を使用し、変更したカラムに "name" という名前をつけます。 データベースサーバはカラム名を大文字に変換するので、 返されるオブジェクトのプロパティは "BREED" および "NAME" となります。

```
<?php
$conn = db2_connect($database, $user, $password);

$sql = "SELECT breed, RTRIM(name) AS name
     FROM animals
     WHERE id = ?";

if ($conn) {
    $stmt = db2_prepare($conn, $sql);
    db2_execute($stmt, array(0));

    while ($pet = db2_fetch_object($stmt)) {
        echo "Come here, {$pet->NAME}, my little {$pet->BREED}!";
    }
    db2_close($conn);
}
?>
```

上の例の出力は以下となります。

```
Come here, Pook, my little cat!
```

参考

- db2_fetch_array()
- db2_fetch_assoc()
- db2_fetch_both()
- db2_fetch_row()
- db2_result()

---

db2_fetch_row

(PECL ibm_db2:1.0-1.6.2)

db2_fetch_row — 結果セットポインタを次の行あるいは要求された行に設定する

説明

bool **db2_fetch_row** (resource $stmt [, int $row_number])

db2_fetch_row() を使用して結果セットを順に処理します。 あるいは、スクロール可能なカーソルを使用している場合は 指定した行を指すようにします。

結果セットから個々のフィールドを取得するには db2_result() 関数をコールします。

db2_fetch_row() および db2_result() をコールするのではなく、 ほとんどのアプリケーションでは db2_fetch_assoc()、db2_fetch_both()、 および db2_fetch_array() のいずれかをコールするでしょう。 これらは、結果セットのポインタを前に進めたうえで行の内容を配列として取得します。

パラメータ

stmt

有効な *stmt* リソース。

row_number

スクロール可能なカーソルの場合に、結果セットの行番号を指定します。 行番号は 1 から始まります。

返り値

結果セットに指定した行が存在する場合に **TRUE**、存在しない場合に **FALSE** を返します。

例

Example#1 結果セットを順に処理する

次の例では、**db2_fetch_row()** を使用して結果セットを順に処理し、db2_result() で結果セットからカラムを取得する方法を示します。

```
<?php
$sql = 'SELECT name, breed FROM animals WHERE weight < ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array(10));
while (db2_fetch_row($stmt)) {
    $name = db2_result($stmt, 0);
    $breed = db2_result($stmt, 1);
    print "$name $breed";
}
?>
```

上の例の出力は以下となります。

```
cat Pook
gold fish Bubbles
budgerigar Gizmo
goat Rickety Ride
```

Example#2 i5/OS recommended alternatives to db2_fetch_row/db2_result

i5/OS では、**db2_fetch_row()**/db2_result() ではなく db2_fetch_both()、db2_fetch_array() あるいは db2_fetch_object() を使用することを推奨します。一般に **db2_fetch_row()**/db2_result() は、さまざまなカラム型で *EBCDIC* から *ASCII* への変換の際に問題が発生します。 *DBCS* アプリケーションではデータが切り捨てられてしまう可能性もあります。 また、パフォーマンス面でも db2_fetch_both()、db2_fetch_array() および db2_fetch_object() は **db2_fetch_row()**/db2_result() よりすぐれています。

```
<?php
  $conn = db2_connect("","","");
  $sql = 'SELECT SPECIFIC_SCHEMA, SPECIFIC_NAME, ROUTINE_SCHEMA, ROUTINE_NAME, ROUTINE_TYPE, ROUTINE_CREATED, ROUTINE_BODY
  $stmt = db2_exec($conn, $sql, array('cursor' => DB2_SCROLLABLE));
  while ($row = db2_fetch_both($stmt)){
    echo "<br>db2_fetch_both {$row['SPECIFIC_NAME']} {$row['ROUTINE_CREATED']} {$row[5]}";
  }
  $stmt = db2_exec($conn, $sql, array('cursor' => DB2_SCROLLABLE));
  while ($row = db2_fetch_array($stmt)){
    echo "<br>db2_fetch_array {$row[1]} {$row[5]}";
  }
  $stmt = db2_exec($conn, $sql, array('cursor' => DB2_SCROLLABLE));
  while ($row = db2_fetch_object($stmt)){
    echo "<br>db2_fetch_object {$row->SPECIFIC_NAME} {$row->ROUTINE_CREATED}";
  }
  db2_close($conn);
?>
```

上の例の出力は以下となります。

```
db2_fetch_both  MATCH_ANIMAL  2006-08-25-17.10.23.775000  2006-08-25-17.10.23.775000
db2_fetch_both  MULTIRESULTS  2006-10-17-10.11.05.308000  2006-10-17-10.11.05.308000
db2_fetch_array  MATCH_ANIMAL  2006-08-25-17.10.23.775000
db2_fetch_array  MULTIRESULTS  2006-10-17-10.11.05.308000
db2_fetch_object  MATCH_ANIMAL  2006-08-25-17.10.23.775000
db2_fetch_object  MULTIRESULTS  2006-10-17-10.11.05.308000
```

参考

- db2_fetch_array()
- db2_fetch_assoc()
- db2_fetch_both()
- db2_fetch_object()
- db2_result()

---

db2_field_display_size

(PECL ibm_db2:1.0-1.6.2)

db2_field_display_size — カラムを表示するために必要な最大のバイト数を返す

説明

int **db2_field_display_size** (resource $stmt , mixed $column)

結果セット内のカラムを表示するために必要な最大のバイト数を返す

パラメータ

stmt

結果セットを含むステートメントリソースを指定します。

column

結果セット内のカラムを指定します。0 から始まるインデックス、 あるいはカラム名を表す文字列のいずれかが使用可能です。

返り値

指定したカラムを表示するために必要な最大バイト数を表す整数値を返します。 結果セット内にそのカラムが存在しない場合には、
db2_field_display_size() は **FALSE** を返します。

参考

- db2_field_name()
- db2_field_num()
- db2_field_precision()
- db2_field_scale()
- db2_field_type()
- db2_field_width()

---

db2_field_name

(PECL ibm_db2:1.0-1.6.2)

db2_field_name — 結果セット内のカラムの名前を返す

説明

string **db2_field_name** (resource $stmt , mixed $column)

結果セット内の指定したカラムの名前を返します。

パラメータ

stmt

結果セットを含むステートメントリソースを指定します。

column

結果セット内のカラムを指定します。0 から始まるインデックス、 あるいはカラム名を表す文字列のいずれかが使用可能です。

返り値

指定したカラムの名前を含む文字列を返します。 結果セット内にそのカラムが存在しない場合には、 **db2_field_name()** は **FALSE** を返します。

参考

- db2_field_display_size()
- db2_field_num()
- db2_field_precision()
- db2_field_scale()
- db2_field_type()
- db2_field_width()

---

db2_field_num

(PECL ibm_db2:1.0-1.6.2)

db2_field_num — 結果セット内の指定したカラムの位置を返す

説明

int **db2_field_num** (resource $stmt , mixed $column)

結果セット内の指定したカラムの位置を返します。

パラメータ

stmt

結果セットを含むステートメントリソースを指定します。

column

結果セット内のカラムを指定します。0 から始まるインデックス、 あるいはカラム名を表す文字列のいずれかが使用可能です。

返り値

結果セット内でのカラムの位置を、0 から始まる整数値で返します。 結果セット内にそのカラムが存在しない場合には、 **db2_field_num()** は **FALSE** を返します。

参考

- db2_field_display_size()
- db2_field_name()
- db2_field_precision()
- db2_field_scale()
- db2_field_type()
- db2_field_width()

---

db2_field_precision

(PECL ibm_db2:1.0-1.6.2)

db2_field_precision — 結果セット内の指定したカラムの精度を返す

説明

int **db2_field_precision** (resource $stmt , mixed $column)

結果セット内の指定したカラムの精度を返します。

パラメータ

stmt

結果セットを含むステートメントリソースを指定します。

column

結果セット内のカラムを指定します。0 から始まるインデックス、 あるいはカラム名を表す文字列のいずれかが使用可能です。

返り値

指定したカラムの精度を表す整数値を返します。 結果セット内にそのカラムが存在しない場合には、 **db2_field_precision()** は **FALSE** を返します。

参考

- db2_field_display_size()
- db2_field_name()
- db2_field_num()
- db2_field_scale()
- db2_field_type()
- db2_field_width()

---

db2_field_scale

(PECL ibm_db2:1.0-1.6.2)

db2_field_scale — 結果セット内の指定したカラムの位取りを返す

説明

int **db2_field_scale** (resource $stmt , mixed $column)

結果セット内の指定したカラムの位取りを返します。

パラメータ

stmt

結果セットを含むステートメントリソースを指定します。

column

結果セット内のカラムを指定します。0 から始まるインデックス、 あるいはカラム名を表す文字列のいずれかが使用可能です。

返り値

指定したカラムの位取りを整数値で返します。 結果セット内にそのカラムが存在しない場合には、 **db2_field_scale()** は **FALSE** を返します。

参考

- db2_field_display_size()
- db2_field_name()
- db2_field_num()
- db2_field_precision()
- db2_field_type()
- db2_field_width()

---

db2_field_type

(PECL ibm_db2:1.0-1.6.2)

db2_field_type — 結果セット内の指定したカラムのデータ型を返す

説明

string **db2_field_type** (resource $stmt , mixed $column)

結果セット内の指定したカラムのデータ型を返します。

パラメータ

stmt

結果セットを含むステートメントリソースを指定します。

column

結果セット内のカラムを指定します。0 から始まるインデックス、 あるいはカラム名を表す文字列のいずれかが使用可能です。

返り値

指定したカラムに定義されているデータ型を表す文字列を返します。 結果セット内にそのカラムが存在しない場合には、 **db2_field_type()** は **FALSE** を返します。

参考

- db2_field_display_size()
- db2_field_name()
- db2_field_num()
- db2_field_precision()
- db2_field_scale()
- db2_field_width()

---

db2_field_width

(PECL ibm_db2:1.0-1.6.2)

db2_field_width — 結果セット内の指定したカラムの現在値の幅を返す

説明

int **db2_field_width** (resource $stmt , mixed $column)

結果セット内の指定したカラムの現在値の幅を返します。 固定長のデータ型では、この値はカラムの最大幅になります。 可変長のデータ型では、実際のカラムの幅となります。

パラメータ

stmt

結果セットを含むステートメントリソースを指定します。

column

結果セット内のカラムを指定します。0 から始まるインデックス、 あるいはカラム名を表す文字列のいずれかが使用可能です。

返り値

結果セット内の文字型あるいはバイナリデータ型のカラムについて、 その幅を整数値で返します。 結果セット内にそのカラムが存在しない場合には、 **db2_field_width()** は **FALSE** を返します。

参考

- db2_field_display_size()
- db2_field_name()
- db2_field_num()
- db2_field_precision()

- db2_field_scale()
- db2_field_type()

---

db2_foreign_keys

(PECL ibm_db2:1.0-1.6.2)

db2_foreign_keys — テーブルの外部キーを含む結果セットを返す

説明

resource **db2_foreign_keys** (resource $connection , string $qualifier , string $schema , string $table-name)

テーブルの外部キーを含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

テーブルを含むスキーマ。*schema* が **NULL** の場合は、**db2_foreign_keys()** 現在の接続のスキーマに一致します。

table-name

テーブルの名前。

返り値

指定したテーブルの外部キーを含む結果セットのステートメントリソースを返します。 結果セットは、以下のカラムで構成されています。

カラム名	説明
PKTABLE_CAT	主キーを含むテーブルのカタログの名前。 テーブルがカタログを保持していない場合は NULL。
PKTABLE_SCHEM	主キーを含むテーブルのスキーマの名前。
PKTABLE_NAME	主キーを含むテーブルの名前。
PKCOLUMN_NAME	主キーを含むカラムの名前。
FKTABLE_CAT	外部キーを含むテーブルのカタログの名前。 テーブルがカタログを保持していない場合は NULL。
FKTABLE_SCHEM	外部キーを含むテーブルのスキーマの名前。
FKTABLE_NAME	外部キーを含むテーブルの名前。
FKCOLUMN_NAME	外部キーを含むカラムの名前。
KEY_SEQ	1 から始まる数字で表した、キー内のカラムの位置。
UPDATE_RULE	SQL で UPDATE 操作を行った際に外部キーに適用される動作を表す整数値。
DELETE_RULE	SQL で DELETE 操作を行った際に外部キーに適用される動作を表す整数値。
FK_NAME	外部キーの名前。
PK_NAME	主キーの名前。
DEFERRABILITY	外部キーの遅延度を表す整数値。 SQL_INITIALLY_DEFERRED、SQL_INITIALLY_IMMEDIATE あるいは SQL_NOT_DEFERRABLE のいずれか。

参考

- db2_column_privileges()
- db2_columns()
- db2_primary_keys()
- db2_procedure_columns()
- db2_procedures()
- db2_special_columns()
- db2_statistics()
- db2_table_privileges()
- db2_tables()

---

db2_free_result

(PECL ibm_db2:1.0-1.6.2)

db2_free_result — 結果セットに関連付けられたリソースを開放する

説明

bool **db2_free_result** (resource $stmt)

結果セットに関連付けられたシステムリソースおよびデータベースリソースを 開放します。これらのリソースはスクリプトの終了時に暗黙的に開放されますが、 スクリプトの終了前に **db2_free_result()** をコールすることで、明示的に結果セットリソースを開放することができます。

パラメータ

stmt

有効なステートメントリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- db2_free_stmt()

---

db2_free_stmt

(PECL ibm_db2:1.0-1.6.2)

db2_free_stmt — 指定されたステートメントリソースに関連付けられたリソースを開放する

説明

bool **db2_free_stmt** (resource $stmt)

ステートメントリソースに関連付けられたシステムリソースおよびデータベースリソースを 開放します。これらのリソースはスクリプトの終了時に暗黙的に開放されますが、 スクリプトの終了前に **db2_stmt_result()** をコールすることで、明示的にステートメントリソースを開放することができます。

パラメータ

stmt

有効なステートメントリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- db2_free_result()

---

db2_get_option

(PECL ibm_db2:1.6.0-1.6.2)

db2_get_option — ステートメントリソースあるいは接続リソースからオプションの値を取得する

説明

string **db2_get_option** (resource $resource , string $option)

ステートメントリソースあるいは接続リソースから、指定したオプションの値を取得します。

パラメータ

resource

A valid statement resource as returned from db2_prepare() が返す有効なステートメントリソース、 あるいは db2_connect() や db2_pconnect() が返す有効な接続リソース。

option

ステートメントや接続の、有効なオプション。 以下の新しいオプションが、ibm_db2 バージョン 1.6.0 以降で使用可能です。 これらは、有用な情報を提供します。これらの情報は、 **db2_get_option()** によって取得します。

> **注意**: **注意** 以前のバージョンの ibm_db2 では、これらの新しいオプションはサポートしていません。
> 各オプションの値を設定する際、 サーバによっては指定したすべての内容を処理できないことがあります。 その場合、値が切り詰められます。
> 指定したオプションが正しく変換されてホストシステムに送信されることを確実にするには、 A から Z までの文字と 0 から 9 までの数字、そしてアンダースコア (_) とピリオド (.) のみを使用するようにします。

userid

> *SQL_ATTR_INFO_USERID* - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのユーザ

ID として使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 16 文字までの長さをサポートしています。 このユーザ ID を、認証時に使用するユーザ ID と混同しないでください。 これは識別のためだけに使用するものであり、認証には用いられません。

acctstr

SQL_ATTR_INFO_ACCTSTR - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのアカウンティング・ストリングとして使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 200 文字までの長さをサポートしています。

applname

SQL_ATTR_INFO_APPLNAME - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのアプリケーション名として使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 32 文字までの長さをサポートしています。

wrkstnname

SQL_ATTR_INFO_WRKSTNNAME - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのワークステーション名として使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 18 文字までの長さをサポートしています。

次の表は、どのオプションがどのリソース型で使用できるのかをまとめたものです。

リソースとパラメータの対応

キー	値	リソース型		
		接続	ステートメント	結果セット
userid	*SQL_ATTR_INFO_USERID*	X	X	-
acctstr	*SQL_ATTR_INFO_ACCTSTR*	X	X	-
applname	*SQL_ATTR_INFO_APPLNAME*	X	X	-
wrkstnname	*SQL_ATTR_INFO_WRKSTNNAME*	X	X	-

返り値

成功した場合に接続属性の現在の設定内容、 失敗した場合に **FALSE** を返します。

例

Example#1 接続リソースのパラメータの設定および取得

```
<?php
/* データベースへの接続パラメータ */
$database = 'SAMPLE';
$user     = 'db2inst1';
$password = 'ibmdb2';

/* 接続リソースの取得 */
$conn = db2_connect($database, $user, $password);

echo "Client attributes passed through conection string:¥n";

/* Create the associative options array with valid key-value pairs */
/* Assign the attributes through connection string */
/* Access the options specified */
$options1 = array('userid' => 'db2inst1');
$conn1 = db2_connect($database, $user, $password, $options1);
$val = db2_get_option($conn1, 'userid');
echo $val . "¥n";

$options2 = array('acctstr' => 'account');
$conn2 = db2_connect($database, $user, $password, $options2);
$val = db2_get_option($conn2, 'acctstr');
echo $val . "¥n";

$options3 = array('applname' => 'myapp');
$conn3 = db2_connect($database, $user, $password, $options3);
$val = db2_get_option($conn3, 'applname');
echo $val . "¥n";

$options4 = array('wrkstnname' => 'workstation');
$conn4 = db2_connect($database, $user, $password, $options4);
$val = db2_get_option($conn4, 'wrkstnname');
echo $val . "¥n";

echo "Client attributes passed post-conection:¥n";

/* Create the associative options array with valid key-value pairs */
/* Assign the attributes after a connection is made */
/* Access the options specified */
$options5 = array('userid' => 'db2inst1');
$conn5 = db2_connect($database, $user, $password);
$rc = db2_set_option($conn5, $options5, 1);
$val = db2_get_option($conn5, 'userid');
echo $val . "¥n";

$options6 = array('acctstr' => 'account');
$conn6 = db2_connect($database, $user, $password);
$rc = db2_set_option($conn6, $options6, 1);
```

```
$val = db2_get_option($conn6, 'acctstr');
echo $val . "¥n";

$options7 = array('applname' => 'myapp');
$conn7 = db2_connect($database, $user, $password);
$rc = db2_set_option($conn7, $options7, 1);
$val = db2_get_option($conn7, 'applname');
echo $val . "¥n";

$options8 = array('wrkstnname' => 'workstation');
$conn8 = db2_connect($database, $user, $password);
$rc = db2_set_option($conn8, $options8, 1);
$val = db2_get_option($conn8, 'wrkstnname');
echo $val . "¥n";
?>
```

上の例の出力は以下となります。

```
Client attributes passed through conection string:
db2inst1
account
myapp
workstation
Client attributes passed post-connection:
db2inst1
account
myapp
workstation
```

参考

- db2_connect()
- db2_cursor_type()
- db2_exec()
- db2_set_option()
- db2_pconnect()
- db2_prepare()

---

db2_lob_read

(PECL ibm_db2:1.6.0-1.6.2)

db2_lob_read — LOB ファイルから、ユーザが定義したサイズの内容を取得する

説明

string **db2_lob_read** (resource $stmt , int $colnum , int $length)

db2_lob_read() により、 結果セットの指定したカラムから、指定したサイズの LOB データを取得します。

パラメータ

stmt

LOB データを含む有効な *stmt* リソース。

colnum

stmt リソースの結果セット内の、有効なカラム番号。

length

stmt リソースから取得する LOB データのサイズ。

返り値

指定したサイズのデータを返します。データが取得できない場合は **FALSE** を返します。

例

Example#1 さまざまな型のデータの取得

```
<?php

/* データベースへの接続パラメータ */
$db = 'SAMPLE';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続リソースの取得 */
$conn = db2_connect($db,$username,$password);

if ($conn) {
    $drop = 'DROP TABLE clob_stream';
    $result = @db2_exec( $conn, $drop );

    $create = 'CREATE TABLE clob_stream (id INTEGER, my_clob CLOB)';
```

```
    $result = db2_exec( $conn, $create );

    $variable = "";
    $stmt = db2_prepare($conn, "INSERT INTO clob_stream (id,my_clob) VALUES (1, ?)");
    $variable = "THIS IS A CLOB TEST. THIS IS A CLOB TEST.";
    db2_bind_param($stmt, 1, "variable", DB2_PARAM_IN);
    db2_execute($stmt);

    $sql = "SELECT id,my_clob FROM clob_stream";
    $result = db2_prepare($conn, $sql);
    db2_execute($result);
    db2_fetch_row($result);
    $i = 0;
    /* LOB データの読み込み */
    while ($data = db2_lob_read($result, 2, 6)) {
        echo "Loop $i: $data¥n";
        $i = $i + 1;
    }

    $drop = 'DROP TABLE blob_stream';
    $result = @db2_exec( $conn, $drop );

    $create = 'CREATE TABLE blob_stream (id INTEGER, my_blob CLOB)';
    $result = db2_exec( $conn, $create );

    $variable = "";
    $stmt = db2_prepare($conn, "INSERT INTO blob_stream (id,my_blob) VALUES (1, ?)");
    $variable = "THIS IS A BLOB TEST. THIS IS A BLOB TEST.";
    db2_bind_param($stmt, 1, "variable", DB2_PARAM_IN);
    db2_execute($stmt);

    $sql = "SELECT id,my_blob FROM blob_stream";
    $result = db2_prepare($conn, $sql);
    db2_execute($result);
    db2_fetch_row($result);
    $i = 0;
    /* LOB データの読み込み */
    while ($data = db2_lob_read($result, 2, 6)) {
        echo "Loop $i: $data¥n";
        $i = $i + 1;
    }
} else {
    echo '接続に失敗しました: ' . db2_conn_errormsg();
}

?>
```

上の例の出力は以下となります。

```
Loop 0: THIS I
Loop 1: S A CL
Loop 2: OB TES
Loop 3: T. THI
Loop 4: S IS A
Loop 5:  CLOB
Loop 6: TEST.
Loop 0: THIS I
Loop 1: S A BL
Loop 2: OB TES
Loop 3: T. THI
Loop 4: S IS A
Loop 5:  BLOB
Loop 6: TEST.
```

参考

- db2_bind_param()
- db2_exec()
- db2_execute()
- db2_fetch_row()
- db2_prepare()
- db2_result()

---

db2_next_result

(PECL ibm_db2:1.0-1.6.2)

db2_next_result — ストアドプロシージャから、次の結果セットを要求する

説明

```
resource db2_next_result ( resource $stmt )
```

ストアドプロシージャは、複数の結果セットを返すことができます。 単純な SELECT 文が返す結果を処理するのと同じように最初の結果セットを扱った後でそれ以降の結果セットを処理する際には、 **db2_next_result()** 関数をコールして その返り値を PHP 変数に代入しなければなりません。

パラメータ

stmt

db2_exec() あるいは db2_execute() から返されるプリペアドステートメント。

返り値

ストアドプロシージャが別の結果セットを返している場合には 次の結果セットを含む新しいステートメントリソース、 別の結果セットを返していない場合には **FALSE** を返します。

例

Example#1 複数の結果セットを返すストアドプロシージャをコールする

次の例では、3 つの結果セットを返すストアドプロシージャをコールします。 最初の結果セットは CALL 文を起動したのと同じステートメントリソースから 直接取得できますが、2 番目および 3 番目の結果セットは **db2_next_result()** が返すステートメントリソースから取得します。

```
<?php
$conn = db2_connect($database, $user, $password);

if ($conn) {
  $stmt = db2_exec($conn, 'CALL multiResults()');

  print "最初の結果セットを取得します\n";
  while ($row = db2_fetch_array($stmt)) {
    var_dump($row);
  }

  print "\n2 番目の結果セットを取得します\n";
  $res = db2_next_result($stmt);
  if ($res) {
    while ($row = db2_fetch_array($res)) {
      var_dump($row);
    }
  }

  print "\n3 番目の結果セットを取得します\n";
  $res2 = db2_next_result($stmt);
  if ($res2) {
    while ($row = db2_fetch_array($res2)) {
      var_dump($row);
    }
  }

  db2_close($conn);
}
?>
```

上の例の出力は以下となります。

```
最初の結果セットを取得します
array(2) {
  [0]=>
  string(16) "Bubbles         "
  [1]=>
  int(3)
}
array(2) {
  [0]=>
  string(16) "Gizmo           "
  [1]=>
  int(4)
}

2 番目の結果セットを取得します
array(4) {
  [0]=>
  string(16) "Sweater         "
  [1]=>
  int(6)
  [2]=>
  string(5) "llama"
  [3]=>
  string(6) "150.00"
}
array(4) {
  [0]=>
  string(16) "Smarty          "
  [1]=>
  int(2)
  [2]=>
  string(5) "horse"
  [3]=>
  string(6) "350.00"
}

3 番目の結果セットを取得します
array(1) {
  [0]=>
  string(16) "Bubbles         "
}
array(1) {
  [0]=>
  string(16) "Gizmo           "
}
```

db2_num_fields

(PECL ibm_db2:1.0-1.6.2)

db2_num_fields — 結果セットに含まれるフィールドの数を返す

説明

int **db2_num_fields** (resource $stmt)

結果セットに含まれるフィールドの数を返します。動的に生成されたクエリが返す 結果セットを処理する場合、あるいはストアドプロシージャから返された結果セットを 処理する場合など、結果セットの使用法がアプリケーションにわからない場合に有用です。

パラメータ

stmt

結果セットを含む有効なステートメントリソース。

返り値

指定したステートメントリソースに関連付けられた結果セット内の フィールドの数を表す整数値を返します。ステートメントリソースが 不正な値の場合には **FALSE** を返します。

例

Example#1 結果セットのフィールド数の取得

以下の例では、結果セットが返すフィールドの数を取得する方法を説明します。

```
<?php

$sql = "SELECT id, name, breed, weight FROM animals ORDER BY breed";
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, $sql);
$columns = db2_num_fields($stmt);

echo "結果セット内には {$columns} つのカラムがあります。";
?>
```

上の例の出力は以下となります。

```
結果セット内には 4 つのカラムがあります。
```

参考

- db2_execute()
- db2_field_display_size()
- db2_field_name()
- db2_field_num()
- db2_field_precision()
- db2_field_scale()
- db2_field_type()
- db2_field_width()

---

db2_num_rows

(PECL ibm_db2:1.0-1.6.2)

db2_num_rows — SQL 文によって変更された行の数を返す

説明

int **db2_num_rows** (resource $stmt)

SQL 文によって削除、挿入あるいは更新された行の数を返します。

SELECT 文が返す行の数を取得するには、調べたい SELECT 文と同じ述部を持つ SELECT COUNT(*) を発行し、その値を取得します。

SELECT 文が返す行の数を調べて 0 の場合には別の処理を行うようなロジックを アプリケーションで使用している場合、その代わりに db2_fetch_assoc()、db2_fetch_both()、 db2_fetch_array() あるいは db2_fetch_row() を使用して最初の行を取得し、 結果が **FALSE** の場合には別の処理を行うという方法を検討してください。

> **注意**: スクロール可能なカーソルを使用して SELECT 文を発行した場合は、 **db2_num_rows()** は SELECT 文の結果の行数を返します。 しかし、スクロール可能なカーソルを使用すると、 アプリケーションのパフォーマンスが急激に低下します。 そのため、SELECT 文の結果の行数を知りたいというだけの理由で スクロール可能なカーソルを使用するのは避けましょう。 前進のみのカーソルを使用し、SELECT COUNT(*) をコールするか フェッチ関数の返す boolean 値を調べるほうが、 同じ機能をずっと高速に実現できます。

パラメータ

stmt

結果セットを含む有効な *stmt* リソース。

返り値

指定したステートメントハンドルによって発行された、 直近の SQL 文によって変更された行の数を返します。

---

db2_pconnect

(PECL ibm_db2:1.0-1.6.2)

db2_pconnect — データベースへの持続的接続を返す

説明

resource **db2_pconnect** (string $database , string $username , string $password [, array $options])

IBM DB2 Universal Database、IBM Cloudscape あるいは Apache Derby データベースへの持続的接続を返します。 持続的接続についての詳細な情報は、 持続的データベース接続 を参照ください。

持続的接続に対して db2_close() をコールすると、 常に **TRUE** を返します。しかし DB2 クライアント接続はオープンされたままであり、 条件に一致する次の **db2_pconnect()** リクエストを待ち続けます。

パラメータ

database

DB2 クライアントカタログ内のデータベースエイリアス。

username

データベースに接続するユーザ名。

password

データベースに接続するパスワード。

options

接続の振る舞いを指定する接続オプションの連想配列。 使用可能なキーは以下のとおりです。

autocommit

DB2_AUTOCOMMIT_ON を渡すと、 この接続ハンドルで自動コミットを有効にします。

DB2_AUTOCOMMIT_OFF を渡すと、 この接続ハンドルで自動コミットを無効にします。

DB2_ATTR_CASE

DB2_CASE_NATURAL を渡すと、 カラム名の大文字小文字を変換せずに返します。

DB2_CASE_LOWER を渡すと、 カラム名を小文字に変換して返します。

DB2_CASE_UPPER を渡すと、 カラム名を大文字に変換して返します。

CURSOR

DB2_FORWARD_ONLY を渡すと、 ステートメントリソースで前進のみのカーソルを使用します。 これはデフォルトのカーソル型であり、 すべてのデータベースサーバでサポートされています。

DB2_SCROLLABLE を渡すと、 ステートメントリソースでスクロール可能なカーソルを使用します。 このモードでは結果セット内の行へのランダムアクセスが可能となりますが、 現在は IBM DB2 Universal Database でしかサポートされていません。

返り値

接続に成功した場合は接続ハンドルリソースを返します。 パラメータ *database* 、*username* および *password* に完全に一致する接続が 既に存在した場合、**db2_pconnect()** はそれを再利用します。接続に失敗した場合は **db2_pconnect()** は **FALSE** を返します。

例

Example#1 db2_pconnect() の例

以下の例で、最初に **db2_pconnect()** をコールした際には新しい持続的接続リソースを返します。 二度目に **db2_pconnect()** をコールすると、最初の持続的接続のリソースが再利用されます。

```
<?php
$database = 'SAMPLE';
$user = 'db2inst1';
$password = 'ibmdb2';

$pconn = db2_pconnect($database, $user, $password);

if ($pconn) {
    echo "持続的接続に成功しました。";
}
else {
    echo "持続的接続に失敗しました。";
}

$pconn2 = db2_pconnect($database, $user, $password);
if ($pconn) {
    echo "2 回目の持続的接続に成功しました。";
}
```

```
else {
    echo "2 回目の持続的接続に失敗しました。";
}
?>
```

上の例の出力は以下となります。

```
持続的接続に成功しました。
2 回目の持続的接続に成功しました。
```

参考

- db2_connect()

---

db2_prepare

(PECL ibm_db2:1.0-1.6.2)

db2_prepare — 実行する SQL 文を準備する

説明

```
resource db2_prepare ( resource $connection , string $statement [, array $options ] )
```

db2_prepare() は、プリペアドステートメントを作成します。 このプリペアドステートメントには、入力パラメータ・出力パラメータ あるいは入出力パラメータを表すパラメータマーカ (?) を含めることができます。プリペアドステートメントにパラメータを渡すには db2_bind_param() を使用します。 入力パラメータの場合についてのみ、db2_execute() に渡す配列で指定することも可能です。

アプリケーション内でプリペアドステートメントを使用する利点は、 以下の 3 つです。 application:

- *パフォーマンス*: 事前に文を準備しておくと、 その文によってデータを取得するための実行計画をデータベースサーバが 最適化します。その後 db2_execute() でプリペアドステートメントを実行する際にはこの実行計画が再利用され、 実行時に実行計画を毎回作成することによるオーバーヘッドを避けられます。
- *セキュリティ*: 事前に文を準備する際に、 入力値にパラメータマーカを含めることができます。 入力値にプレースホルダを使用してプリペアドステートメントを実行すると、 入力値の型がカラム定義あるいはパラメータ定義と一致することを データベースサーバがチェックします。
- *高機能*: パラメータマーカの機能は、 単に入力値をプリペアドステートメントに渡すだけではありません。 db2_bind_param() を使用すると、 ストアドプロシージャの OUT パラメータおよび INOUT パラメータから 値を取得することも可能です。

パラメータ

connection

db2_connect() あるいは db2_pconnect() が返した有効なデータベース接続リソース。

statement

ひとつ以上のパラメータマーカを含む SQL 文。

options

文のオプションを含む連想配列。 データベースサーバがその機能をサポートしている場合に、 このパラメータを使用してスクロール可能なカーソルの使用を 要求することができます。

cursor

DB2_FORWARD_ONLY を渡すと、 この SQL 文で前進のみのカーソルを使用することを要求します。 これはデフォルトのカーソル型であり、すべてのデータベースサーバで サポートされています。また、スクロール可能なカーソルに比べて 非常に高速になります。

DB2_SCROLLABLE を渡すと、 この SQL 文でスクロール可能なカーソルを使用することを要求します。 このカーソル型を使用すると、データベースサーバから 行の並び順を気にせずにデータを取得できるようになります。 しかし、この型は DB2 サーバでしかサポートされておらず、前進のみのカーソルに比べて非常に低速です。

返り値

SQL 文のパースに成功し、データベースサーバ内で正しく準備された場合に ステートメントリソースを返します。データベースサーバがエラーを返した場合に **FALSE** を返します。返されたエラーの詳細を調べるには、 db2_stmt_error() あるいは db2_stmt_errormsg() をコールします。

例

Example#1 パラメータマーカを使用した SQL 文の準備と実行

以下の例では、4 つのパラメータマーカを含む INSERT 文を準備し、 入力値の配列を含む配列を順に処理しながら db2_execute() に値を渡します。

```
<?php
$animals = array(
     array(0, 'cat', 'Pook', 3.2),
     array(1, 'dog', 'Peaches', 12.3),
     array(2, 'horse', 'Smarty', 350.0),
);

$insert = 'INSERT INTO animals (id, breed, name, weight)
     VALUES (?, ?, ?, ?)';
$stmt = db2_prepare($conn, $insert);
if ($stmt) {
     foreach ($animals as $animal) {
         $result = db2_execute($stmt, $animal);
```

```
    }
}
?>
```

参考

- db2_bind_param()
- db2_execute()
- db2_stmt_error()
- db2_stmt_errormsg()

---

db2_primary_keys

(PECL ibm_db2:1.0-1.6.2)

db2_primary_keys — テーブルの主キーを含む結果セットを返す

説明

resource **db2_primary_keys** (resource $connection , string $qualifier , string $schema , string $table-name)

テーブルの主キーを含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

テーブルを含むスキーマ。*schema* が **NULL** の場合、**db2_primary_keys()** は現在の接続のスキーマを使用します。

table-name

テーブルの名前。

返り値

指定したテーブルの主キーを含む結果セットのステートメントリソースを返します。 結果セットは、以下のカラムで構成されています。

カラム名	説明
TABLE_CAT	主キーを含むテーブルのカタログの名前。 テーブルがカタログを保持していない場合は NULL。
TABLE_SCHEM	主キーを含むテーブルのスキーマの名前。
TABLE_NAME	主キーを含むテーブルの名前。
COLUMN_NAME	主キーを含むカラムの名前。
KEY_SEQ	1 から始まる数字で表した、キー内のカラムの位置。
PK_NAME	主キーの名前。

参考

- db2_column_privileges()
- db2_columns()
- db2_foreign_keys()
- db2_procedure_columns()
- db2_procedures()
- db2_special_columns()
- db2_statistics()
- db2_table_privileges()
- db2_tables()

---

db2_procedure_columns

(PECL ibm_db2:1.0-1.6.2)

db2_procedure_columns — ストアドプロシージャのパラメータを含む結果セットを返す

説明

resource **db2_procedure_columns** (resource $connection , string $qualifier , string $schema , string $procedure , string $parameter)

ストアドプロシージャのパラメータを含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には NULL あるいは空の文字列を渡します。

schema

プロシージャを含むスキーマ。このパラメータでは、ワイルドカードとして _ および % を含む検索パターンを使用可能です。

procedure

プロシージャの名前。このパラメータでは、ワイルドカードとして _ および % を含む検索パターンを使用可能です。

parameter

パラメータの名前。このパラメータでは、ワイルドカードとして _ および % を含む検索パターンを使用可能です。 このパラメータが NULL の場合、指定したストアドプロシージャの すべてのパラメータが返されます。

返り値

指定したパラメータに一致するストアドプロシージャのパラメータ情報を含む結果セットの ステートメントリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	説明	
PROCEDURE_CAT	プロシージャを含むカタログ。テーブルがカタログを保持していない場合は NULL。	
PROCEDURE_SCHEM	ストアドプロシージャを含むスキーマの名前。	
PROCEDURE_NAME	プロシージャの名前。	
COLUMN_NAME	パラメータの名前。	
COLUMN_TYPE	パラメータの型を表す整数値。	
	返される値	**パラメータの型**
	1 (SQL_PARAM_INPUT)	入力 (IN) パラメータ。
	2 (SQL_PARAM_INPUT_OUTPUT)	入力/出力 (INOUT) パラメータ。
	3 (SQL_PARAM_OUTPUT)	出力 (OUT) パラメータ。
DATA_TYPE	パラメータの SQL データ型を表す整数値。	
TYPE_NAME	パラメータのデータ型を表す文字列。	
COLUMN_SIZE	パラメータのサイズを表す整数値。	
BUFFER_LENGTH	このパラメータのデータを保存するために必要な最大バイト数。	
DECIMAL_DIGITS	パラメータの位取り。位取りが適用できない場合は NULL。	
NUM_PREC_RADIX	*10* (正確な数値データ型を表す)、 *2* (概数データ型を表す)、 あるいは NULL (基数が適用できないデータ型を表す) のいずれか。	
NULLABLE	パラメータが null 値をとることができるかどうかを表す整数値。	
REMARKS	パラメータの説明。	
COLUMN_DEF	パラメータのデフォルト値。	
SQL_DATA_TYPE	パラメータのサイズを表す整数値。	
SQL_DATETIME_SUB	datetime 型のコードを表す整数値、 あるいはこれが適用できない SQL データ型である場合に NULL。	
CHAR_OCTET_LENGTH	文字型のパラメータにおける最大のオクテット数。 シングルバイト文字セットのデータの場合、これは COLUMN_SIZE に一致します。文字型でないカラムの場合は NULL となります。	
ORDINAL_POSITION	CALL 文の中でのパラメータの位置を表す、1 から始まるインデックス。	
IS_NULLABLE	パラメータが null 値をとることができるかどうかを表す文字列。 'YES' の場合は null 値をとることができ、 'NO' の場合はできません。	

参考

- db2_column_privileges()
- db2_columns()
- db2_foreign_keys()
- db2_primary_keys()
- db2_procedures()
- db2_special_columns()
- db2_statistics()
- db2_table_privileges()
- db2_tables()

db2_procedures

(PECL ibm_db2:1.0-1.6.2)

db2_procedures — データベース内に登録されているストアドプロシージャの一覧を含む結果セットを返す

説明

resource **db2_procedures** (resource $connection , string $qualifier , string $schema , string $procedure)

データベース内に登録されているストアドプロシージャの一覧を含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

プロシージャを含むスキーマ。このパラメータでは、ワイルドカードとして _ および % を含む検索パターンを使用可能です。

procedure

プロシージャの名前。このパラメータでは、ワイルドカードとして _ および % を含む検索パターンを使用可能です。

返り値

指定したパラメータに一致するストアドプロシージャの情報を含む結果セットの ステートメントリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	説明
PROCEDURE_CAT	プロシージャを含むカタログ。テーブルがカタログを保持していない場合は **NULL**。
PROCEDURE_SCHEM	ストアドプロシージャを含むスキーマの名前。
PROCEDURE_NAME	プロシージャの名前。
NUM_INPUT_PARAMS	ストアドプロシージャの入力 (IN) パラメータの数。
NUM_OUTPUT_PARAMS	ストアドプロシージャの出力 (OUT) パラメータの数。
NUM_RESULT_SETS	ストアドプロシージャが返す結果セットの数。
REMARKS	ストアドプロシージャのコメント。
PROCEDURE_TYPE	常に *1* を返します。これは、 ストアドプロシージャが返り値を返さないことを意味します。

参考

- db2_column_privileges()
- db2_columns()
- db2_foreign_keys()
- db2_primary_keys()
- db2_procedure_columns()
- db2_special_columns()
- db2_statistics()
- db2_table_privileges()
- db2_tables()

---

db2_result

(PECL ibm_db2:1.0-1.6.2)

db2_result — 結果セットの行からひとつのカラムを返す

説明

mixed **db2_result** (resource $stmt , mixed $column)

db2_result() を使用して、 結果セットの現在の行から指定したカラムの値を取得します。 結果セットポインタの位置を指定するため、**db2_result()** のコール前には db2_fetch_row() をコールする必要があります。

パラメータ

stmt

有効な *stmt* リソース。

column

結果セット内のカラムの位置を表す 0 から始まるインデックス、 あるいはカラム名を表す文字列。

返り値

指定したフィールドが結果セットに存在する場合にそのフィールドの値を返します。 フィールドが存在しない場合には NULL を返し、警告を発生させます。

例

Example#1 db2_result() の例

以下の例では、db2_fetch_row() で結果セットを 順に処理し、**db2_result()** で結果セットから カラムを取得する方法を説明します。

```
<?php
$sql = 'SELECT name, breed FROM animals WHERE weight < ?';
$stmt = db2_prepare($conn, $sql);
db2_execute($stmt, array(10));
while (db2_fetch_row($stmt)) {
    $name = db2_result($stmt, 0);
    $breed = db2_result($stmt, 'BREED');
    print "$name $breed";
}
?>
```

上の例の出力は以下となります。

```
cat Pook
gold fish Bubbles
budgerigar Gizmo
goat Rickety Ride
```

参考

- db2_fetch_array()
- db2_fetch_assoc()
- db2_fetch_both()
- db2_fetch_object()
- db2_fetch_row()

---

db2_rollback

(PECL ibm_db2:1.0-1.6.2)

db2_rollback — トランザクションをロールバックする

説明

bool **db2_rollback** (resource $connection)

指定した接続リソース上で実行中のトランザクションをロールバックし、 新しいトランザクションを開始します。PHP アプリケーションのデフォルトは AUTOCOMMIT モードなので、接続リソースに対して AUTOCOMMIT を無効にしていない限り **db2_rollback()** は何の意味もありません。

> **注意**: 指定した接続が持続的接続であった場合、持続的接続を使用している すべてのアプリケーションで実行中のトランザクションがロールバックされます。 そのため、トランザクションが必要なアプリケーションでは 持続的接続の使用は推奨されません。

パラメータ

connection

db2_connect() あるいは db2_pconnect() が返した有効なデータベース接続リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 DELETE 文のロールバック

以下の例では、テーブルの行数を数えた後にデータベース接続の AUTOCOMMIT を無効にし、テーブルのすべての行を削除した上で、 行数が *0* となっていることを確認します。 それから、**db2_rollback()** を発行して再度行数を取得し、 DELETE 文の発行前の状態に戻っていることを確認します。これにより、トランザクションのロールバックが正常に行われたことを示します。

```
<?php
$conn = db2_connect($database, $user, $password);

if ($conn) {
    $stmt = db2_exec($conn, "SELECT count(*) FROM animals");
    $res = db2_fetch_array( $stmt );
    echo $res[0] . "¥n";

    // AUTOCOMMIT を無効にします
    db2_autocommit($conn, DB2_AUTOCOMMIT_OFF);

    // ANIMALS のすべての行を削除します
    db2_exec($conn, "DELETE FROM animals");
```

```
    $stmt = db2_exec($conn, "SELECT count(*) FROM animals");
    $res = db2_fetch_array( $stmt );
    echo $res[0] . "¥n";

    // DELETE 文をロールバックします
    db2_rollback( $conn );

    $stmt = db2_exec( $conn, "SELECT count(*) FROM animals" );
    $res = db2_fetch_array( $stmt );
    echo $res[0] . "¥n";
    db2_close($conn);
}
?>
```

上の例の出力は以下となります。

```
7
0
7
```

参考

- db2_autocommit()
- db2_commit()

---

db2_server_info

(PECL ibm_db2:1.1.1-1.6.2)

db2_server_info — DB2 データベースサーバの情報をプロパティに保持するオブジェクトを返す

説明

object **db2_server_info** (resource $connection)

この関数は、IBM DB2、Cloudscape あるいは Apache Derby データベースサーバについての情報を 読み取り専用のプロパティに保持するオブジェクトを返します。 以下の表は、データベースサーバのプロパティの一覧です。

データベースサーバのプロパティ

プロパティ名	返り値の型	説明
DBMS_NAME	string	接続中のデータベースサーバの名前。DB2 サーバの場合は、 *DB2* の後にサーバが稼動している OS の名前が続きます。
DBMS_VER	string	データベースサーバのバージョン。"MM.mm.uuuu" という形式の文字列で、 *MM* がメジャーバージョン、 *mm* がマイナーバージョン、 そして *uuuu* がアップデートを表します。 例えば "08.02.0001" は メジャーバージョン 8、マイナーバージョン 2、アップデート 1 を表します。
DB_CODEPAGE	int	接続中のデータベースのコードページ。
DB_NAME	string	接続中のデータベースの名前。
DFT_ISOLATION	string	サーバがサポートしているデフォルトのトランザクション分離レベル。 UR Uncommitted read: 変更内容は、他のトランザクションからも すぐに見えるようになります。 CS Cursor stability: あるトランザクションで読み込まれた行を、 別のトランザクションから変更してコミットすることができます。 RS Read stability: 検索条件あるいは実行中のトランザクションに 一致する行を、トランザクションから追加したり削除したりできます。 RR Repeatable read: 実行中のトランザクションに影響を受けるデータは、 他のトランザクションからは見えません。 NC No commit: 変更内容は、操作が終了した時点で見えるようになります。 明示的なコミットやロールバックはできません。
IDENTIFIER_QUOTE_CHAR	string	識別子を区切るための文字。
INST_NAME	string	データベースを含むデータベースサーバのインスタンス名。
ISOLATION_OPTION	array	データベースサーバがサポートする分離オプション。 分離オプションについては DFT_ISOLATION プロパティの説明を参照ください。
KEYWORDS	array	データベースサーバの予約語の配列。
LIKE_ESCAPE_CLAUSE	bool	データベースサーバがワイルドカード文字 % および _ の使用をサポートしている場合に **TRUE**、 これらのワイルドカードをサポートしていない場合に **FALSE**。
MAX_COL_NAME_LEN	int	データベースサーバがサポートするカラム名の最大バイト数。
MAX_IDENTIFIER_LEN	int	データベースサーバがサポートする SQL 識別子の最大文字数。
MAX_INDEX_SIZE	int	データベースサーバがサポートするインデックスつきカラムの最大バイト数。
MAX_PROC_NAME_LEN	int	データベースサーバがサポートするプロシージャ名の最大バイト数。
MAX_ROW_SIZE	int	データベースサーバがサポートするベーステーブルの行の最大バイト数。
MAX_SCHEMA_NAME_LEN	int	データベースサーバがサポートするスキーマ名の最大バイト数。
MAX_STATEMENT_LEN	int	データベースサーバがサポートする SQL 文の最大バイト数。
MAX_TABLE_NAME_LEN	int	データベースサーバがサポートするテーブル名の最大バイト数。
NON_NULLABLE_COLUMNS	bool	データベースサーバがカラムの NOT NULL 定義をサポートしている場合に **TRUE**、サポートしていない場合に **FALSE**。
PROCEDURES	bool	データベースサーバがストアドプロシージャをコールするための CALL 文をサポートしている場合に **TRUE**、サポートしていない場合に **FALSE**。
SPECIAL_CHARS	string	識別子として使用可能な文字のうち、 a-Z、0-9 およびアンダースコア以外の文字。
SQL_CONFORMANCE	string	データベースサーバの ANSI/ISO SQL-92 仕様への対応レベル。 ENTRY SQL-92 エントリレベルに準拠しています。 FIPS127

プロパティ名	返り値の型	説明

パラメータ

connection

アクティブな DB2 クライアント接続を指定します。

返り値

成功した場合にオブジェクト、失敗した場合に **FALSE** を返します。

例

Example#1 db2_server_info() の例

サーバの情報を取得するには、有効なデータベース接続リソースを **db2_server_info()** に渡す必要があります。

```
<?php

$conn = db2_connect('sample', 'db2inst1', 'ibmdb2');

$server = db2_server_info( $conn );

if ($server) {
     echo "DBMS_NAME:  ";                  var_dump( $server->DBMS_NAME  );
     echo "DBMS_VER:  ";                   var_dump( $server->DBMS_VER  );
     echo "DB_CODEPAGE:  ";                var_dump( $server->DB_CODEPAGE  );
     echo "DB_NAME:  ";                    var_dump( $server->DB_NAME  );
     echo "INST_NAME:  ";                  var_dump( $server->INST_NAME  );
     echo "SPECIAL_CHARS:  ";              var_dump( $server->SPECIAL_CHARS  );
     echo "KEYWORDS:  ";                   var_dump( sizeof($server->KEYWORDS) );
     echo "DFT_ISOLATION:  ";              var_dump( $server->DFT_ISOLATION  );
     echo "ISOLATION_OPTION:  ";
     $il = '';
      foreach( $server->ISOLATION_OPTION  as $opt )
     {
        $il .= $opt." ";
     }
     var_dump( $il );
     echo "SQL_CONFORMANCE:  ";            var_dump( $server->SQL_CONFORMANCE  );
     echo "PROCEDURES:  ";                 var_dump( $server->PROCEDURES  );
     echo "IDENTIFIER_QUOTE_CHAR:  ";      var_dump( $server->IDENTIFIER_QUOTE_CHAR  );
     echo "LIKE_ESCAPE_CLAUSE:  ";         var_dump( $server->LIKE_ESCAPE_CLAUSE  );
     echo "MAX_COL_NAME_LEN:  ";           var_dump( $server->MAX_COL_NAME_LEN  );
     echo "MAX_ROW_SIZE:  ";               var_dump( $server->MAX_ROW_SIZE  );
     echo "MAX_IDENTIFIER_LEN:  ";         var_dump( $server->MAX_IDENTIFIER_LEN  );
     echo "MAX_INDEX_SIZE:  ";             var_dump( $server->MAX_INDEX_SIZE  );
     echo "MAX_PROC_NAME_LEN:  ";          var_dump( $server->MAX_PROC_NAME_LEN  );
     echo "MAX_SCHEMA_NAME_LEN:  ";        var_dump( $server->MAX_SCHEMA_NAME_LEN  );
     echo "MAX_STATEMENT_LEN:  ";          var_dump( $server->MAX_STATEMENT_LEN  );
     echo "MAX_TABLE_NAME_LEN:  ";         var_dump( $server->MAX_TABLE_NAME_LEN  );
     echo "NON_NULLABLE_COLUMNS:  ";       var_dump( $server->NON_NULLABLE_COLUMNS  );

     db2_close($conn);
}
?>
```

上の例の出力は以下となります。

```
DBMS_NAME:  string(9)  "DB2/LINUX"
DBMS_VER:  string(10)  "08.02.0000"
DB_CODEPAGE:  int(1208)
DB_NAME:  string(6)  "SAMPLE"
INST_NAME:  string(8)  "db2inst1"
SPECIAL_CHARS:  string(2)  "@#"
KEYWORDS:  int(179)
DFT_ISOLATION:  string(2)  "CS"
ISOLATION_OPTION:  string(12)  "UR  CS  RS  RR  "
SQL_CONFORMANCE:  string(7)  "FIPS127"
PROCEDURES:  bool(true)
IDENTIFIER_QUOTE_CHAR:  string(1)  """
LIKE_ESCAPE_CLAUSE:  bool(true)
MAX_COL_NAME_LEN:  int(30)
MAX_ROW_SIZE:  int(32677)
MAX_IDENTIFIER_LEN:  int(18)
MAX_INDEX_SIZE:  int(1024)
MAX_PROC_NAME_LEN:  int(128)
MAX_SCHEMA_NAME_LEN:  int(30)
MAX_STATEMENT_LEN:  int(2097152)
MAX_TABLE_NAME_LEN:  int(128)
NON_NULLABLE_COLUMNS:  bool(true)
```

参考

- db2_client_info()

---

db2_set_option

(PECL ibm_db2:1.0-1.6.2)

db2_set_option — 接続リソースあるいはステートメントリソースのオプションを設定する

説明

bool **db2_set_option** (resource $resource , array $options , int $type)

ステートメントリソースあるいは接続リソースのオプションを設定します。 結果セットリソースのオプションを設定することはできません。

パラメータ

resource

db2_prepare() が返す有効なステートメントリソースか、 あるいは db2_connect() や db2_pconnect() が返す有効な接続リソース。

options

ステートメントあるいは接続のオプションを含む連想配列。 このパラメータは、自動コミットの値を変更したりカーソルの型を（スクロール可能、あるいは前進のみに）変更したり、結果セットに表示されるカラム名を（小文字、大文字、あるいは元のままに）設定したりする際に使用します。

autocommit

DB2_AUTOCOMMIT_ON を渡すと、 指定した接続リソースの自動コミットをオンにします。

DB2_AUTOCOMMIT_OFF を渡すと、 指定した接続リソースの自動コミットをオフにします。

cursor

DB2_FORWARD_ONLY を渡すと、 ステートメントリソースに前進のみのカーソルを指定します。 これはデフォルトのカーソル型であり、 すべてのデータベースサーバでサポートされています。

DB2_SCROLLABLE を渡すと、 ステートメントリソースにスクロール可能なカーソルを指定します。 このカーソルは、順番どおり以外の方法で結果セットの行にアクセス可能です。 しかし、IBM DB2 Universal Database でしかサポートされていません。

binmode

DB2_BINARY を渡すと、 バイナリデータがそのままの形式で返されるようになります。 これはデフォルトのモードです。*php.ini* で *ibm_db2.binmode=1* とするのと同じことです。

DB2_CONVERT を渡すと、 バイナリデータを十六進エンコーディングで変換して返します。 これは、*php.ini* で *ibm_db2.binmode=2* とするのと同じことです。

DB2_PASSTHRU を渡すと、 バイナリデータを **NULL** に変換するようになります。これは *php.ini* で *ibm_db2.binmode=3* とするのと同じことです。

db2_attr_case

DB2_CASE_LOWER を渡すと、 結果セットのカラム名を小文字で返します。

DB2_CASE_UPPER を渡すと、 結果セットのカラム名を大文字で返します。

DB2_CASE_NATURAL を渡すと、 結果セットのカラム名をそのまま何もせずに返します。

deferred_prepare

DB2_DEFERRED_PREPARE_ON を渡すと、 指定したステートメントリソースについて遅延プリペアを有効にします。

DB2_DEFERRED_PREPARE_OFF を渡すと、 指定したステートメントリソースについて遅延プリペアを無効にします。

以下の新しい i5/OS オプションは、ibm_db2 のバージョン 1.5.1 以降で使用可能です。

注意: **注意** それ以前のバージョンの ibm_db2 は、これらの新しい i5 オプションをサポートしていません。

i5_fetch_only

DB2_I5_FETCH_ON - カーソルは読み込み専用となり、場所を指定しての更新や削除には使用できません。 これは、*SQL_ATTR_FOR_FETCH_ONLY* が *SQL_FALSE* に設定されていない場合のデフォルト設定です。

DB2_I5_FETCH_OFF - カーソルは、場所を指定しての更新や削除に使用できるようになります。

以下の新しいオプションが、ibm_db2 バージョン 1.6.0 以降で使用可能です。 これらは、有用な情報を提供します。これらの情報は、db2_get_option() によって取得します。

注意: **注意** 以前のバージョンの ibm_db2 では、これらの新しいオプションはサポートしていません。
各オプションの値を設定する際、 サーバによっては指定したすべての内容を処理できないことがあります。 その場合、値が切り詰められます。
指定したオプションが正しく変換されてホストシステムに送信されることを確実にするには、 A から Z までの文字と 0 から 9 までの数字、そしてアンダースコア（_）とピリオド（.）のみを使用するようにします。

userid

SQL_ATTR_INFO_USERID - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのユーザ ID として使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 16 文字までの長さをサポートしています。 このユーザ ID を、認証時に使用するユーザ ID と混同しないでください。 これは識別のためだけに使用するものであり、認証には用いられません。

acctstr

SQL_ATTR_INFO_ACCTSTR - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのアカウンティング・ストリングとして使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 200 文字までの長さをサポートしています。

applname

SQL_ATTR_INFO_APPLNAME - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのアプリケーション名として使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 32 文字までの長さをサポートしています。

wrkstnname

SQL_ATTR_INFO_WRKSTNNAME - ヌル終端の文字列へのポインタで、DB2 接続の際にホストデータベースサーバに送信する クライアントのワークステーション名として使用します。

注意: **注意** z/OS および OS/390 サーバ版の DB2 は 18 文字までの長さをサポートしています。

type

この関数に渡すリソースの型を、整数値で指定します。 リソースの型とこの値は必ず一致していなければなりません。

1 は、 接続リソースがこの関数に渡されることを表します。

1 以外の整数を指定すると、 ステートメントリソースがこの関数に渡されることを表します。

次の表は、それぞれのオプションがどのリソース型で使用可能かをまとめたものです。

リソースとパラメータの対応

キー	値	リソースの型		
		接続	ステートメント	結果セット
autocommit	*DB2_AUTOCOMMIT_ON*	X	-	-
autocommit	*DB2_AUTOCOMMIT_OFF*	X	-	-
cursor	*DB2_SCROLLABLE*	-	X	-
cursor	*DB2_FORWARD_ONLY*	-	X	-
binmode	*DB2_BINARY*	X	X	-
binmode	*DB2_CONVERT*	X	X	-
binmode	*DB2_PASSTHRU*	X	X	-
db2_attr_case	*DB2_CASE_LOWER*	X	X	-
db2_attr_case	*DB2_CASE_UPPER*	X	X	-
db2_attr_case	*DB2_CASE_NATURAL*	X	X	-
deferred_prepare	*DB2_DEFERRED_PREPARE_ON*	-	X	-
deferred_prepare	*DB2_DEFERRED_PREPARE_OFF*	-	X	-
i5_fetch_only	*DB2_I5_FETCH_ON*	-	X	-
i5_fetch_only	*DB2_I5_FETCH_OFF*	-	X	-
userid	*SQL_ATTR_INFO_USERID*	X	X	-
acctstr	*SQL_ATTR_INFO_ACCTSTR*	X	X	-
applname	*SQL_ATTR_INFO_APPLNAME*	X	X	-
wrkstnname	*SQL_ATTR_INFO_WRKSTNNAME*	X	X	-

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 接続リソースにひとつのパラメータを設定する

```
<?php
/* データベース接続用のパラメータ */
$database = 'SAMPLE';
$hostname = 'localhost';
$port = 50000;
$protocol = 'TCPIP';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続文字列 */
$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;";
$conn_string .= "HOSTNAME=$hostname;PORT=$port;PROTOCOL=$protocol;";
$conn_string .= "UID=$username;PWD=$password;";

/* 接続リソースを取得します */
$conn = db2_connect($conn_string, '', '');

/* 有効なキーと値を使用して、オプションの連想配列を作成します */
$options = array('autocommit' => DB2_AUTOCOMMIT_ON);

/* 正しいリソースとオプションの配列、そして型を表す値を指定して関数をコールします */
$result = db2_set_option($conn, $options, 1);

/* すべてのオプションが正しく設定できたかどうかを調べます */
if($result)
{
  echo 'オプションの設定に成功しました';
}
```

```
else
{
  echo 'オプションが設定できませんでした';
}
?>
```

上の例の出力は以下となります。

```
オプションの設定に成功しました
```

Example#2 接続リソースに複数のパラメータを設定する

```
<?php
/* データベース接続用のパラメータ */
$database = 'SAMPLE';
$hostname = 'localhost';
$port = 50000;
$protocol = 'TCPIP';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続文字列 */
$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;";
$conn_string .= "HOSTNAME=$hostname;PORT=$port;PROTOCOL=$protocol;";
$conn_string .= "UID=$username;PWD=$password;";

/* 接続リソースを取得します */
$conn = db2_connect($conn_string, '', '');

/* 有効なキーと値を使用して、オプションの連想配列を作成します */
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF,
                 'binmode' => DB2_PASSTHRU,
              'db2_attr_case' => DB2_CASE_UPPER,
                 'cursor' => DB2_SCROLLABLE);

/* 正しいリソースとオプションの配列、そして型を表す値を指定して関数をコールします */
$result = db2_set_option($conn, $options, 1);

/* すべてのオプションが正しく設定できたかどうかを調べます */
if($result)
{
  echo 'オプションの設定に成功しました';
}
else
{
  echo 'オプションが設定できませんでした';
}
?>
```

上の例の出力は以下となります。

```
オプションの設定に成功しました
```

Example#3 複数のパラメータを間違ったキーで設定する

```
<?php
/* データベース接続用のパラメータ */
$database = 'SAMPLE';
$hostname = 'localhost';
$port = 50000;
$protocol = 'TCPIP';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続文字列 */
$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;";
$conn_string .= "HOSTNAME=$hostname;PORT=$port;PROTOCOL=$protocol;";
$conn_string .= "UID=$username;PWD=$password;";

/* 接続リソースを取得します */
$conn = db2_connect($conn_string, '', '');

/* 有効なキーと値を使用して、オプションの連想配列を作成します */
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF,
              'MY_INVALID_KEY' => DB2_PASSTHRU,
               'db2_attr_case' => DB2_CASE_UPPER,
                     'cursor' => DB2_SCROLLABLE);

/* 正しいリソースとオプションの配列、そして型を表す値を指定して関数をコールします */
$result = db2_set_option($conn, $options, 1);

/* すべてのオプションが正しく設定できたかどうかを調べます */
if($result)
{
  echo 'オプションの設定に成功しました';
}
else
{
  echo 'オプションが設定できませんでした';
}
?>
```

上の例の出力は以下となります。

```
オプションが設定できませんでした
```

Example#4 複数のパラメータを間違った値で設定する

```
<?php
/* データベース接続用のパラメータ */
```

```
$database = 'SAMPLE';
$hostname = 'localhost';
$port = 50000;
$protocol = 'TCPIP';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続文字列 */
$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;";
$conn_string .= "HOSTNAME=$hostname;PORT=$port;PROTOCOL=$protocol;";
$conn_string .= "UID=$username;PWD=$password;";

/* 接続リソースを取得します */
$conn = db2_connect($conn_string, '', '');

/* 有効なキーと値を使用して、オプションの連想配列を作成します */
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF,
                 'binmode' => 'INVALID_VALUE',
            'db2_attr_case' => DB2_CASE_UPPER,
                  'cursor' => DB2_SCROLLABLE);

/* 正しいリソースとオプションの配列、そして型を表す値を指定して関数をコールします */
$result = db2_set_option($conn, $options, 1);

/* すべてのオプションが正しく設定できたかどうかを調べます */
if($result)
{
  echo 'オプションの設定に成功しました';
}
else
{
  echo 'オプションが設定できませんでした';
}
?>
```

上の例の出力は以下となります。

```
オプションが設定できませんでした
```

Example#5 複数のパラメータを接続リソースに設定する（間違った型を指定する）

```
<?php
/* データベース接続用のパラメータ */
$database = 'SAMPLE';
$hostname = 'localhost';
$port = 50000;
$protocol = 'TCPIP';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続文字列 */
$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;";
$conn_string .= "HOSTNAME=$hostname;PORT=$port;PROTOCOL=$protocol;";
$conn_string .= "UID=$username;PWD=$password;";

/* 接続リソースを取得します */
$conn = db2_connect($conn_string, '', '');

/* 有効なキーと値を使用して、オプションの連想配列を作成します */
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF,
                 'binmode' => DB2_PASSTHRU,
            'db2_attr_case' => DB2_CASE_UPPER,
                  'cursor' => DB2_SCROLLABLE);

/* 正しいリソースとオプションの配列、そして間違った型を表す値を指定して関数をコールします */
$result = db2_set_option($conn, $options, 2);

/* すべてのオプションが正しく設定できたかどうかを調べます */
if($result)
{
  echo 'オプションの設定に成功しました';
}
else
{
  echo 'オプションが設定できませんでした';
}
?>
```

上の例の出力は以下となります。

```
オプションが設定できませんでした
```

Example#6 複数のパラメータを間違ったリソースに設定する

```
<?php
/* データベース接続用のパラメータ */
$database = 'SAMPLE';
$hostname = 'localhost';
$port = 50000;
$protocol = 'TCPIP';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続文字列 */
$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;";
$conn_string .= "HOSTNAME=$hostname;PORT=$port;PROTOCOL=$protocol;";
$conn_string .= "UID=$username;PWD=$password;";

/* 接続リソースを取得します */
$conn = db2_connect($conn_string, '', '');
```

```
/* 有効なキーと値を使用して、オプションの連想配列を作成します */
$options = array('autocommit' => DB2_AUTOCOMMIT_OFF,
                 'binmode' => DB2_PASSTHRU,
              'db2_attr_case' => DB2_CASE_UPPER,
                 'cursor' => DB2_SCROLLABLE);

$stmt = db2_prepare($conn, 'SELECT * FROM EMPLOYEE');

/* 間違ったリソース、そして正しいオプションの配列と型を表す値を指定して関数をコールします */
$result = db2_set_option($stmt, $options, 1);

/* すべてのオプションが正しく設定できたかどうかを調べます */
if($result)
{
  echo 'オプションの設定に成功しました';
}
else
{
  echo 'オプションが設定できませんでした';
}
?>
```

上の例の出力は以下となります。

```
オプションが設定できませんでした
```

Example#7 すべてひとつにまとめる

```
<?php
/* データベース接続用のパラメータ */
$database = 'SAMPLE';
$hostname = 'localhost';
$port = 50000;
$protocol = 'TCPIP';
$username = 'db2inst1';
$password = 'ibmdb2';

/* 接続文字列 */
$conn_string = "DRIVER={IBM DB2 ODBC DRIVER};DATABASE=$database;";
$conn_string .= "HOSTNAME=$hostname;PORT=$port;PROTOCOL=$protocol;";
$conn_string .= "UID=$username;PWD=$password;";

/* 接続リソースを取得します */
$conn = db2_connect($conn_string, '', '');

/* 有効なキーと値を使用して、オプションの連想配列を作成します */
$options = array('db2_attr_case' => DB2_CASE_LOWER,
                 'cursor' => DB2_SCROLLABLE);

$stmt = db2_prepare($conn, 'SELECT * FROM EMPLOYEE WHERE EMPNO = ? OR EMPNO = ?');

/* 正しいリソースとオプションの配列、そして型を表す値を指定して関数をコールします */
$option_result = db2_set_option($stmt, $options, 2);
$result = db2_execute($stmt, array('000130', '000140'));

/* スクロール可能なカーソルのため、1 行目より先に 2 行目を取得します */
print_r(db2_fetch_assoc($stmt, 2));
print '<br /><br />';
print_r(db2_fetch_assoc($stmt, 1));

?>
```

上の例の出力は以下となります。

```
Array
(
     [empno] => 000140
     [firstnme] => HEATHER
     [midinit] => A
     [lastname] => NICHOLLS
     [workdept] => C01
     [phoneno] => 1793
     [hiredate] => 1976-12-15
     [job] => ANALYST
     [edlevel] => 18
     [sex] => F
     [birthdate] => 1946-01-19
     [salary] => 28420.00
     [bonus] => 600.00
     [comm] => 2274.00
)

Array
(
     [empno] => 000130
     [firstnme] => DELORES
     [midinit] => M
     [lastname] => QUINTANA
     [workdept] => C01
     [phoneno] => 4578
     [hiredate] => 1971-07-28
     [job] => ANALYST
     [edlevel] => 16
     [sex] => F
     [birthdate] => 1925-09-15
     [salary] => 23800.00
     [bonus] => 500.00
     [comm] => 1904.00
)
```

Example#8 i5/OS カーソルは読み込み専用

```php
<?php
  $conn = db2_connect("", "", "", array("i5_lib"=>"nobody"));
  $stmt = db2_prepare($conn, 'select * from names where first = ?');
  $name = "first2";
  db2_bind_param($stmt, 1, "name", DB2_PARAM_IN);
  $options = array("i5_fetch_only"=>DB2_I5_FETCH_ON);
  db2_set_option($stmt,$options,0);
  if (db2_execute($stmt)) {
    while ($row = db2_fetch_array($stmt)) {
      echo "{$row[0]}  {$row[1]}";
    }
  }
?>
```

上の例の出力は以下となります。

```
first2  last2
```

参考

- db2_connect()
- db2_pconnect()
- db2_exec()
- db2_prepare()
- db2_cursor_type()

---

db2_special_columns

(PECL ibm_db2:1.0-1.6.2)

db2_special_columns — テーブルのユニーク行 ID カラムを含む結果セットを返す

説明

resource **db2_special_columns** (resource $connection , string $qualifier , string $schema , string $table_name , int $scope)

テーブルのユニーク行 ID カラムを含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

テーブルを含むスキーマ。

table_name

テーブルの名前。

scope

ユニーク行 ID の有効期間の最小値を表す整数値。 以下の値のいずれかとなります。

整数値	SQL 定数	説明
0	SQL_SCOPE_CURROW	行 ID は、カーソルがその行にある場合にのみ有効です。
1	SQL_SCOPE_TRANSACTION	行 ID は、そのトランザクションの間のみ有効です。
2	SQL_SCOPE_SESSION	行 ID は、その接続の間のみ有効です。

返り値

テーブルのユニーク行 ID 情報を含む結果セットの ステートメントリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	整数値	SQL 定数	説明
SCOPE	0	SQL_SCOPE_CURROW	行 ID は、カーソルがその行にある場合にのみ有効です。
	1	SQL_SCOPE_TRANSACTION	行 ID は、そのトランザクションの間のみ有効です。
	2	SQL_SCOPE_SESSION	行 ID は、その接続の間のみ有効です。
COLUMN_NAME	ユニークカラムの名前。		
DATA_TYPE	カラムの SQL データ型。		
TYPE_NAME	カラムの SQL データ型を文字列で表したもの。		
COLUMN_SIZE	カラムのサイズを表す整数値。		

カラム名	説明
BUFFER_LENGTH	このカラムのデータを保存するために必要な最大のバイト数。
DECIMAL_DIGITS	カラムの位取り。位取りが適用できない場合は **NULL**。
NUM_PREC_RADIX	*10* (正確な数値データ型を表す)、 *2* (概数データ型を表す)、 あるいは **NULL** (基数が適用できないデータ型を表す) のいずれか。
PSEUDO_COLUMN	常に 1 を返します。

参考

- *db2_column_privileges()*
- *db2_columns()*
- *db2_foreign_keys()*
- *db2_primary_keys()*
- *db2_procedure_columns()*
- *db2_procedures()*
- *db2_statistics()*
- *db2_table_privileges()*
- *db2_tables()*

---

db2_statistics

(PECL ibm_db2:1.0-1.6.2)

db2_statistics — インデックスの情報およびテーブルの統計情報を含む結果セットを返す

説明

*resource **db2_statistics** (resource $connection , string $qualifier , string $schema , string $table-name , bool $unique)*

インデックスの情報およびテーブルの統計情報を含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

*OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。*

schema

*対象となるテーブルを含むスキーマ。 このパラメータを **NULL** にすると、現在のユーザのスキーマについての 統計およびインデックスが返されます。*

table_name

テーブルの名前。

unique

返されるインデックスの型を表す整数値。

0

テーブルのユニークインデックスについての情報のみを返します。

1

テーブルのすべてのインデックスについての情報を返します。

返り値

指定したパラメータに一致するテーブルの統計およびインデックスを含む ステートメントリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	説明
TABLE_CAT	テーブルを含むカタログの名前。テーブルがカタログを保持していない場合は **NULL**。
TABLE_SCHEM	テーブルを含むスキーマの名前。
TABLE_NAME	テーブルの名前。
	インデックスが一意でなければならないかどうか、 あるいはその行がテーブルの統計情報であるかどうかを表す整数値。

返り値	パラメータの型
NON_UNIQUE	0 (SQL_FALSE) インデックスは、重複した値をとることができます。
	1 (SQL_TRUE) インデックスは、一意でなければなりません。
	NULL この行は、テーブル自身についての統計情報です。

カラム名	説明
INDEX_QUALIFIER	インデックスの資格を満たすため、INDEX_NAME の先頭に付加しなければならない修飾子を表す文字列。
INDEX_NAME	インデックスの名前を表す文字列。
TYPE	結果セットのこの行に含まれる情報の型を表す整数値。 返り値 / パラメータの型 0 (SQL_TABLE_STAT) この行は、テーブル自身の統計情報を含みます。 1 (SQL_INDEX_CLUSTERED) この行は、クラスタ化インデックスについての情報を含みます。 2 (SQL_INDEX_HASH) この行は、ハッシュインデックスについての情報を含みます。 3 (SQL_INDEX_OTHER) この行は、クラスタ化もハッシュも行われていない インデックスについての情報を含みます。
ORDINAL_POSITION	インデックス内の、1 から始まるカラムの位置。 その行がテーブル自身についての統計情報を含んでいる場合は **NULL**。
COLUMN_NAME	インデックス内のカラムの名前。 その行がテーブル自身についての統計情報を含んでいる場合は **NULL**。
ASC_OR_DESC	カラムの並び順が昇順の場合は *A*、 降順の場合は *D*、 その行がテーブル自身についての統計情報を含んでいる場合は **NULL**。
CARDINALITY	その行がインデックスの情報を含んでいる場合、このカラムには インデックス内の一意な値の数を表す整数値が含まれます。 その行がテーブル自身についての情報を含んでいる場合、 このカラムにはテーブルの行数を表す整数値が含まれます。
PAGES	その行がインデックスの情報を含んでいる場合、このカラムには インデックスを保存するために使用しているページ数を表す整数値が含まれます。 その行がテーブル自身についての情報を含んでいる場合、このカラムには テーブルを保存するために使用しているページ数を表す整数値が含まれます。
FILTER_CONDITION	常に **NULL**。

参考

- *db2_column_privileges()*
- *db2_columns()*
- *db2_foreign_keys()*
- *db2_primary_keys()*
- *db2_procedure_columns()*
- *db2_procedures()*
- *db2_special_columns()*
- *db2_table_privileges()*
- *db2_tables()*

---

db2_stmt_error

(PECL ibm_db2:1.0-1.6.2)

db2_stmt_error — SQL 文が返す SQLSTATE を含む文字列を返す

説明

*string **db2_stmt_error** ([resource $stmt])*

SQL 文が返す SQLSTATE を含む文字列を返します。

***db2_stmt_error()** の引数にステートメントリソースを 渡さなかった場合は、直近で (例えば db2_prepare() あるいは db2_exec() を使用して) 返そうとしたステートメントリソースに関連する SQLSTATE を返します。*

*SQLSTATE の値の意味を調べるには、DB2 コマンドラインプロセッサのプロンプトで 次のコマンドを発行します。 **db2 '? sqlstate-value '** また、db2_stmt_errormsg() をコールして、 明示的なエラーメッセージと、それに関連する SQLCODE の値を取得することも可能です。*

パラメータ

stmt

有効なステートメントリソース。

返り値

SQLSTATE を含む文字列を返します。

参考

- *db2_conn_error()*
- *db2_conn_errormsg()*
- *db2_stmt_errormsg()*

---

db2_stmt_errormsg

(PECL ibm_db2:1.0-1.6.2)

db2_stmt_errormsg — 直近の SQL 文のエラーメッセージを含む文字列を返す

説明

string **db2_stmt_errormsg** ([resource $stmt])

直近の SQL 文のエラーメッセージを含む文字列を返します。

db2_stmt_errormsg() の引数にステートメントリソースを 渡さなかった場合は、直近で (例えば db2_prepare() あるいは db2_exec() を使用して) 返そうとしたステートメントリソースに関連するエラーメッセージを返します。

パラメータ

stmt

有効なステートメントリソース。

返り値

SQL 文の発行により直近に発生したエラーを表す エラーメッセージおよび SQLSTATE を返します。

参考

- db2_conn_error()
- db2_conn_errormsg()
- db2_stmt_error()

---

db2_table_privileges

(PECL ibm_db2:1.0-1.6.2)

db2_table_privileges — データベース内のテーブルおよび関連する権限情報を含む結果セットを返す

説明

resource **db2_table_privileges** (resource $connection [, string $qualifier [, string $schema [, string $table_name]]])

データベース内のテーブルおよび関連する権限情報を含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

テーブルを含むスキーマ。このパラメータには、ワイルドカードとして _ および % を含む検索パターンを指定することができます。

table_name

テーブルの名前。このパラメータには、ワイルドカードとして _ および % を含む検索パターンを指定することができます。

返り値

指定したパラメータに一致するテーブルの権限情報を含むステートメントリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	説明
TABLE_CAT	テーブルを含むカタログの名前。テーブルがカタログを保持していない場合は **NULL**。
TABLE_SCHEM	テーブルを含むスキーマの名前。
TABLE_NAME	テーブルの名前。
GRANTOR	その権限を与えたユーザの認証 ID。
GRANTEE	その権限を与えられたユーザの認証 ID。
PRIVILEGE	与えられた権限。ALTER、 CONTROL、DELETE、INDEX、INSERT、REFERENCES、SELECT あるいは UPDATE のいずれかです。
IS_GRANTABLE	grantee が、この権限を他のユーザに与えることができるかどうかを、 文字列 "YES" あるいは "NO" で表す。

参考

- db2_column_privileges()
- db2_columns()

- db2_foreign_keys()
- db2_primary_keys()
- db2_procedure_columns()
- db2_procedures()
- db2_special_columns()
- db2_statistics()
- db2_tables()

---

db2_tables

(PECL ibm_db2:1.0-1.6.2)

db2_tables — データベース内のテーブルおよび関連するメタデータを含む結果セットを返す

説明

resource **db2_tables** (resource $connection [, string $qualifier [, string $schema [, string $table-name [, string $table-type]]]])

データベース内のテーブルおよび関連するメタデータを含む結果セットを返します。

パラメータ

connection

IBM DB2、Cloudscape あるいは Apache Derby データベースへの有効な接続。

qualifier

OS/390 あるいは z/OS サーバ上で稼動している DB2 データベースの修飾子。 その他のデータベースの場合には **NULL** あるいは空の文字列を渡します。

schema

テーブルを含むスキーマ。このパラメータでは、ワイルドカードとして _ および % を含む検索パターンを使用可能です。

table-name

テーブルの名前。このパラメータでは、ワイルドカードとして _ および % を含む検索パターンを使用可能です。

table-type

テーブル型の識別子をカンマで区切ったリスト。 すべてのテーブル型を指定するには、**NULL** あるいは空の文字列を指定します。 有効なテーブル型識別子は ALIAS、HIERARCHY TABLE、INOPERATIVE VIEW、NICKNAME、 MATERIALIZED QUERY TABLE、SYSTEM TABLE、TABLE、TYPED TABLE、TYPED VIEW および VIEW です。

返り値

指定したパラメータに一致するテーブルの情報を含む結果セットのリソースを返します。 行の内容は、以下のカラムで構成されています。

カラム名	説明
TABLE_CAT	テーブルを含むカタログ。テーブルがカタログを保持していない場合は **NULL**。
TABLE_SCHEM	テーブルを含むスキーマの名前。
TABLE_NAME	テーブルの名前。
TABLE_TYPE	テーブルの型識別子。
REMARKS	テーブルの説明。

参考

- db2_column_privileges()
- db2_columns()
- db2_foreign_keys()
- db2_primary_keys()
- db2_procedure_columns()
- db2_procedures()
- db2_special_columns()
- db2_statistics()
- db2_table_privileges()

目次

- db2_column_privileges — テーブルのカラムおよび関連する権限情報を含む結果セットを返す
- db2_columns — テーブルのカラムおよび関連するメタデータを含む結果セットを返す
- db2_commit — トランザクションをコミットする
- db2_conn_error — 直近の接続から返された SQLSTATE を含む文字列を返す
- db2_conn_errormsg — 直近の接続エラーメッセージおよび SQLCODE を返す
- db2_connect — データベースへの接続を返す
- db2_cursor_type — ステートメントリソースが使用しているカーソルの型を返す
- db2_escape_string — 特定の文字をエスケープする
- db2_exec — SQL 文を直接実行する
- db2_execute — プリペアドステートメントを実行する
- db2_fetch_array — 結果セット内の行を表す、カラム位置をインデックスとする配列を返す
- db2_fetch_assoc — 結果セット内の行を表す、カラム名をインデックスとする配列を返す
- db2_fetch_both — 結果セット内の行を表す、 カラム位置およびカラム名の両方をインデックスとする配列を返す
- db2_fetch_object — 結果セット内の行を表す、カラムをプロパティとするオブジェクトを返す
- db2_fetch_row — 結果セットポインタを次の行あるいは要求された行に設定する
- db2_field_display_size — カラムを表示するために必要な最大のバイト数を返す
- db2_field_name — 結果セット内のカラムの名前を返す
- db2_field_num — 結果セット内の指定したカラムの位置を返す
- db2_field_precision — 結果セット内の指定したカラムの精度を返す
- db2_field_scale — 結果セット内の指定したカラムの位取りを返す
- db2_field_type — 結果セット内の指定したカラムのデータ型を返す
- db2_field_width — 結果セット内の指定したカラムの現在値の幅を返す
- db2_foreign_keys — テーブルの外部キーを含む結果セットを返す
- db2_free_result — 結果セットに関連付けられたリソースを開放する
- db2_free_stmt — 指定されたステートメントリソースに関連付けられたリソースを開放する
- db2_get_option — ステートメントリソースあるいは接続リソースからオプションの値を取得する
- db2_lob_read — LOB ファイルから、ユーザが定義したサイズの内容を取得する
- db2_next_result — ストアドプロシージャから、次の結果セットを要求する
- db2_num_fields — 結果セットに含まれるフィールドの数を返す
- db2_num_rows — SQL 文によって変更された行の数を返す
- db2_pconnect — データベースへの持続的接続を返す
- db2_prepare — 実行する SQL 文を準備する
- db2_primary_keys — テーブルの主キーを含む結果セットを返す
- db2_procedure_columns — ストアドプロシージャのパラメータを含む結果セットを返す
- db2_procedures — データベース内に登録されているストアドプロシージャの一覧を含む結果セットを返す
- db2_result — 結果セットの行からひとつのカラムを返す
- db2_rollback — トランザクションをロールバックする
- db2_server_info — DB2 データベースサーバの情報をプロパティに保持するオブジェクトを返す
- db2_set_option — 接続リソースあるいはステートメントリソースのオプションを設定する
- db2_special_columns — テーブルのユニーク行 ID カラムを含む結果セットを返す
- db2_statistics — インデックスの情報およびテーブルの統計情報を含む結果セットを返す
- db2_stmt_error — SQL 文が返す SQLSTATE を含む文字列を返す
- db2_stmt_errormsg — 直近の SQL 文のエラーメッセージを含む文字列を返す
- db2_table_privileges — データベース内のテーブルおよび関連する権限情報を含む結果セットを返す
- db2_tables — データベース内のテーブルおよび関連するメタデータを含む結果セットを返す

---

iconv 関数

導入

このモジュールには、iconv による文字集合の変換機能へのインターフェースが含まれています。 このモジュールにより、ローカルな文字集合で表された文字列を、Unicode 文字集合のような 他の文字集合で表わされた文字列に変換することができます。 サポートされる文字集合は、使用するシステムの iconv の実装に依存します。 いくつかのシステムでは iconv 関数は意図した通りに動作しない可能性があることに注意してください。 この場合は、» GNU libiconv ライブラリをインストール すると良いでしょう。 このライブラリの出力は、多くの場合、より妥当なものとなります。

PHP 5.0.0以降、この拡張モジュールには、 多言語スクリプトを書く際に有用な種々のユーティリティが附属しています。 この新機能については、以下のセクションを参照してください。

要件

使用しているシステムが、最近のPOSIX対応システムのどれかである場合には、標準C 言語ライブラリの中にiconv機能が含まれているため、何かをする必要はありません。 そうでない場合は、 » libiconvライブラリを入手して使用する システムにインストールする必要があります。

インストール手順

この関数を使用するには、 PHPバイナリを以下のオプションを指定して 構築する必要があります。 --with-iconv[=DIR]

注意: **Windows® ユーザへの注意** Windows®環境でこのモジュールを有効にするには、 PHP/Win32バイナリパッケージに附属する

iconv.dllまたは iconv-1.3.dll (4.2.1より前)という名前の DLLファイルを環境変数PATHで指定した ディレクトリまたは Windows® のシステムディレクトリにコピーする 必要があります。 このモジュールは、PHP 5 で PHP の一部として組み込まれました。 iconv.dll や php_iconv.dll は もはや必要ありません。

実行時設定

php.ini の設定により動作が変化します。

Iconv 設定オプション

名前	デフォルト	変更の可否	変更履歴
iconv.input_encoding	"ISO-8859-1"	PHP_INI_ALL	PHP 4.0.5 以降で使用可能です。
iconv.output_encoding	"ISO-8859-1"	PHP_INI_ALL	PHP 4.0.5 以降で使用可能です。
iconv.internal_encoding	"ISO-8859-1"	PHP_INI_ALL	PHP 4.0.5 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

警告

(IBM AIX のように)いくつかのシステムでは "ISO-8859-1" ではなく "ISO8859-1" を使用することがあります。そのような場合は、設定オプションや 関数のパラメータにはこの値を使用しなければなりません。

注意: 設定オプション iconv.input_encoding は、 今のところ一切利用されていません。

リソース型

リソース型は定義されていません。

定義済み定数

PHP 4.3.0以降、この拡張モジュールで使用される iconv 実装の種類を実行時に調べることができます。

iconv定数

名前	型	説明
ICONV_IMPL	string	実装の名前
ICONV_VERSION	string	実装のバージョン

注意: これらの定数を用いて実装に依存するスクリプトを書くことは全く推奨されません。

PHP 5.0.0以降、以下の定数も利用可能です。

PHP 5.0.0以降で利用可能なiconv定数

名前	型	説明
ICONV_MIME_DECODE_STRICT	integer	iconv_mime_decode()で使用されるビットマスク
ICONV_MIME_DECODE_CONTINUE_ON_ERROR	integer	iconv_mime_decode()で使用されるビットマスク

参考

GNU Recode 関数も参照してください。

---

iconv_get_encoding

(PHP 4 >= 4.0.5, PHP 5)

iconv_get_encoding — iconv 拡張モジュールの内部設定変数を取得する

説明

mixed **iconv_get_encoding** ([string $type])

iconv 拡張モジュールの内部設定変数を取得します。

パラメータ

type

オプション type の値は以下のどれかとすることができます。

- all
- input_encoding
- output_encoding
- internal_encoding

返り値

成功時に内部設定変数の現在の設定、失敗時に **FALSE** を返します。

type が省略されるか、"all"が指定された場合、 **iconv_get_encoding()** は上記変数全てを格納した 配列を返します。

例

Example#1 iconv_get_encoding() の例

```
<pre>
<?php
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
var_dump(iconv_get_encoding('all'));
?>
</pre>
```

上の例の出力は以下となります。

```
Array
(
      [input_encoding] => ISO-8859-1
      [output_encoding] => ISO-8859-1
      [internal_encoding] => UTF-8
)
```

参考

- iconv_set_encoding()
- ob_iconv_handler()

---

iconv_mime_decode_headers

(PHP 5)

iconv_mime_decode_headers — 複数の MIME ヘッダフィールドを一度にデコードする

説明

array **iconv_mime_decode_headers** (string $encoded_headers [, int $mode [, string $charset]])

成功した場合は連想配列を返します。その中身には encoded_headers で指定した MIME ヘッダフィールドがすべて含まれています。 デコード中にエラーが発生した場合は **FALSE** を返します。

連想配列の個々のキーがフィールド名を表し、対応する要素がフィールドの 値を表します。同じ名前のフィールドが複数存在する場合は、**iconv_mime_decode_headers()** が自動的に連番つきの 配列をつくり、出現順にその配列に入れられます。

パラメータ

encoded_headers

エンコードされたヘッダを表す文字列。

mode

mode は、**iconv_mime_decode_headers()** が不正な形式の MIME ヘッダフィールドに遭遇した場合の 振る舞いを定義します。以下のビットマスクの組み合わせで指定が可能です。

iconv_mime_decode_headers() で指定できるビットマスク

値	定数名	説明
1	ICONV_MIME_DECODE_STRICT	指定すると、ヘッダは » RFC2047 で定義されている標準に完全準拠する形式でデコードされます。 このオプションはデフォルトでは無効になっています。なぜなら、世の中には おかしなメールソフトが多く存在し、それらは規格に従わずに間違った MIME ヘッダを生成するからです。
2	ICONV_MIME_DECODE_CONTINUE_ON_ERROR	指定すると、**iconv_mime_decode_headers()** は文法的なエラーを無視し、デコード作業を継続します。

charset

オプションの charset パラメータは、結果の 文字セットを指定します。指定されなかった場合は iconv.internal_encoding が用いられます。

例

Example#1 iconv_mime_decode_headers() の例

```
<?php
$headers_string = <<<EOF
Subject: =?UTF-8?B?UHLDvGZ1bmcgUHLDvGZ1bmc=?=
To: example@example.com
Date: Thu, 1 Jan 1970 00:00:00 +0000
Message-Id: <example@example.com>
Received: from localhost (localhost [127.0.0.1]) by localhost
      with SMTP id example for <example@example.com>;
      Thu, 1 Jan 1970 00:00:00 +0000 (UTC)
       (envelope-from example-return-0000-example=example.com@example.com)
Received: (qmail 0 invoked by uid 65534); 1 Thu 2003 00:00:00 +0000

EOF;

$headers = iconv_mime_decode_headers($headers_string, 0, "ISO-8859-1");
print_r($headers);
?>
```

上の例の出力は以下となります。

```
Array
(
      [Subject] => Pr?ung Pr?ung
      [To] => example@example.com
      [Date] => Thu, 1 Jan 1970 00:00:00 +0000
       [Message-Id] => <example@example.com>
      [Received] => Array
         (
                  [0] => from localhost (localhost [127.0.0.1]) by localhost with SMTP id example for <example@example.com>
                [1] => (qmail 0 invoked by uid 65534); 1 Thu 2003 00:00:00 +0000
         )

)
```

参考

- iconv_mime_decode()
- mb_decode_mimeheader()
- imap_mime_header_decode()
- imap_base64()
- imap_qprint()

---

iconv_mime_decode

(PHP 5)

iconv_mime_decode — MIME ヘッダフィールドをデコードする

説明

string **iconv_mime_decode** (string $encoded_header [, int $mode [, string $charset]])

MIME ヘッダフィールドをデコードします。

パラメータ

encoded_header

エンコードされたヘッダを表す文字列。

mode

mode は、**iconv_mime_decode()** が不正な形式の MIME ヘッダフィールドに遭遇した場合の 振る舞いを定義します。以下のビットマスクの組み合わせで指定が可能です。

iconv_mime_decode() で指定できるビットマスク

値	定数名	説明
1	ICONV_MIME_DECODE_STRICT	指定すると、ヘッダは » RFC2047 で定義されている標準に完全準拠する形式でデコードされます。 このオプションはデフォルトでは無効になっています。なぜなら、世の中には おかしなメールソフトが多く存在し、それらは規格に従わずに間違った MIME ヘッダを生成するからです。
2	ICONV_MIME_DECODE_CONTINUE_ON_ERROR	指定すると、iconv_mime_decode_headers() は文法的なエラーを無視し、デコード作業を継続します。

charset

オプションの charset パラメータは、結果の 文字セットを指定します。指定されなかった場合は iconv.internal_encoding が用いられます。

返り値

成功した場合はデコードされた MIME フィールドを、 デコード中にエラーが発生した場合は **FALSE** を返します。

例

Example#1 iconv_mime_decode() の例

```
<?php
// この結果は "Subject: Prüfung Prüfung" となります。
echo iconv_mime_decode("Subject:  =?UTF-8?B?UHLDvGZ1bmcgUHLDvGZ1bmc=?=",
                       0, "ISO-8859-1");
?>
```

参考

- iconv_mime_decode_headers()
- mb_decode_mimeheader()
- imap_mime_header_decode()
- imap_base64()
- imap_qprint()

---

iconv_mime_encode

(PHP 5)

iconv_mime_encode — MIME ヘッダフィールドを作成する

説明

string **iconv_mime_encode** (string $field_name , string $field_value [, array $preferences])

有効な MIME ヘッダフィールドを作成し、返します。 これは以下のような形式になります。

Subject: =?ISO-8859-1?Q?Pr=FCfung_f=FCr?= Entwerfen von einer MIME kopfzeile

上の例では "Subject" がフィールド名、"=?ISO-8859-1?..." で始まる部分が フィールドの値となります。

パラメータ

field_name

フィールド名。

field_value

フィールドの値。

preferences

iconv_mime_encode() の振る舞いを変更するには、 ここで設定項目を含む連想配列を指定します。 **iconv_mime_encode()** でサポートされている項目は 以下のとおりです。項目名の大文字・小文字は区別されることに注意してください。

iconv_mime_encode() でサポートされる設定項目

項目	型	説明	デフォルト値	例
scheme	string	フィールドの値のエンコード方法を指定します。"B" か "Q" の どちらかを指定することになるでしょう。"B" は base64 エンコードを、また "Q" は quoted-printable エンコードを表します。	B	B
input-charset	string	第 1 パラメータ field_name と第 2 パラメータ field_value の文字セットを指定します。 指定されなかった場合は、**iconv_mime_encode()** は ini 設定 iconv.internal_encoding であると仮定します。	iconv.internal_encoding	ISO-8859-1
output-charset	string	MIME ヘッダを作成する文字セットを指定します。 指定されなかった場合は、input-charset と同じ値が用いられます。	iconv.internal_encoding	UTF-8
line-length	integer	ヘッダ行の長さの最大値を指定します。もし結果がこの値より 長くなった場合は、 » RFC2822 - Internet Message Format に基づいてヘッダを "折りたたんで" 複数行に分割します。 指定されなかった場合は、長さは 76 文字に限定されます。	76	996
line-break-chars	string	長いヘッダフィールドに対して "折りたたみ" 処理が行われる場合に 個々の行の後ろに付加される文字列を指定します。 指定されなかった場合は、"\r\n" (CR LF) が用いられます。 このパラメータは、input-charset の値にかかわらず 常に ASCII 文字列として扱われることに注意してください。	¥r¥n	¥n

返り値

成功した場合はエンコードした MIME フィールド、 エンコード時にエラーが発生した場合は **FALSE** を返します。

例

Example#1 iconv_mime_encode() の例:

```
<?php
$preferences = array(
    "input-charset" => "ISO-8859-1",
    "output-charset" => "UTF-8",
    "line-length" => 76,
    "line-break-chars" => "¥n"
);
$preferences["scheme"] = "Q";
// この結果は "Subject: =?UTF-8?Q?Pr=C3=BCfung_Pr=C3=BCfung?=" となります。
echo iconv_mime_encode("Subject", "Prüfung Prüfung", $preferences);

$preferences["scheme"] = "B";
// この結果は "Subject: =?UTF-8?B?UHLDvGZ1bmcgUHLDvGZ1bmc=?=" となります。
echo iconv_mime_encode("Subject", "Prüfung Prüfung", $preferences);
?>
```

参考

- imap_binary()
- mb_encode_mimeheader()
- imap_8bit()

---

iconv_set_encoding

(PHP 4 >= 4.0.5, PHP 5)

iconv_set_encoding — 文字エンコーディング変換用の設定を行なう

説明

bool **iconv_set_encoding** (string $type , string $charset)

type で指定された内部設定変数の値を、 charset に変更します。

パラメータ

type

type には以下の値のどれかを指定できます。

- input_encoding
- output_encoding
- internal_encoding

charset

文字セット。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 iconv_set_encoding() の例

```
<?php
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
?>
```

参考

- iconv_get_encoding()
- ob_iconv_handler()

---

iconv_strlen

(PHP 5)

iconv_strlen — 文字列の文字数を返す

説明

int **iconv_strlen** (string $str [, string $charset])

strlen() とは違い、**iconv_strlen()** は与えられたバイト列 str 中に現れる文字の数を 指定された文字セットに基づいて数えます。この結果は、必ずしも文字列の バイト数と一致するとは限りません。

パラメータ

str

文字列。

charset

charset パラメータが指定されなかった場合、 str のエンコードは iconv.internal_encoding であると判断されます。

返り値

str の文字数を返します。

参考

- strlen()
- mb_strlen()

---

iconv_strpos

(PHP 5)

iconv_strpos — 文字列が最初に現れる場所を見つける

説明

int **iconv_strpos** (string $haystack , string $needle [, int $offset [, string $charset]])

needle が haystack の中で最初に現れる位置を探します。

strpos() の返り値は needle が見つかった位置の 先頭からのバイト数でしたが、それとは異なり iconv_strpos() の返り値は needle が見つかった 位置の先頭からの文字数となります。文字数は charset で指定された文字セットに基づいて 数えられます。

パラメータ

haystack

文字列全体。

needle

検索する文字列。

offset

オプションの offset パラメータは 検索を開始する位置を指定します。

charset

charset パラメータが指定されなかった場合、 string のエンコードは iconv.internal_encoding であると判断されます。

haystack や needle が文字列でない場合、文字列に変換され、文字が並んだ値として適用されます。

返り値

needle が haystack の中で最初に現れる位置を探します。

もし needle が見つからなかった場合、 iconv_strpos() は FALSE を返します。

警告

この関数は論理値 FALSE を返す可能性がありますが、FALSE として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

参考

- strpos()
- iconv_strrpos()
- mb_strpos()

---

iconv_strrpos

(PHP 5)

iconv_strrpos — 文字列が最後に現れる場所を見つける

説明

int **iconv_strrpos** (string $haystack , string $needle [, string $charset])

strrpos() の返り値は needle が見つかった位置の 先頭からのバイト数でしたが、それとは異なり iconv_strrpos() の返り値は needle が見つかった 位置の先頭からの文字数となります。

パラメータ

haystack

文字列全体。

needle

検索する文字列。

charset

charset パラメータが指定されなかった場合、 string のエンコードは iconv.internal_encoding であると判断されます。

haystack や needle が文字列でない場合、文字列に変換され、文字が並んだ値として適用されます。

返り値

文字列 haystack の中で、 needle が最後に現れた位置を数字で返します。 文字数は charset で指定された文字セットに基づいて 数えられます。

もし needle が見つからなかった場合、 iconv_strrpos() は FALSE を返します。

警告

この関数は論理値 FALSE を返す可能性がありますが、FALSE として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

参考

- strrpos()
- iconv_strpos()

- mb_strrpos()

---

iconv_substr

(PHP 5)

iconv_substr — 文字列の一部を切り出す

説明

string **iconv_substr** (string $str , int $offset [, int $length [, string $charset]])

文字列 str の、 start と length で指定された一部を返します。

パラメータ

str

元の文字列。

offset

start が負でない場合、 **iconv_substr()** は str の start 番目の文字（ゼロから数えて）から切り出します。

start が負の場合、 **iconv_substr()** は str の最後から数えて start 番目の文字から切り出します。

length

length が指定され、かつ正である場合、 返される文字列は start から数えて最大 length 文字数分となります (string の長さに依存します)。

もし負の length が渡された場合に **iconv_substr()** が切り出す文字列は、 str の start 番目の文字からはじまり、文字列の最後から数えて length 文字分戻ったところまでとなります。 start も負の場合、開始位置は 上で説明した方式で計算されます。

charset

charset が指定されなかった場合、文字セットは ini 設定 iconv.internal_encoding で定義された値とみなされます。

offset や length のパラメータは、常に charset で定義された 文字セットにおける文字数と判断されることに注意してください。 一方、 substr() の場合はこれらの値を常に バイト数として判断します。

返り値

文字列 str の、 offset と length で指定された一部を返します。

もし str が start の文字列長より短い場合は、**FALSE** が返されます。

参考

- substr()
- mb_substr()
- mb_strcut()

---

iconv

(PHP 4 >= 4.0.5, PHP 5)

iconv — 文字列を指定した文字エンコーディングに変換する

説明

string **iconv** (string $in_charset , string $out_charset , string $str)

文字列 str の文字セットを in_charset から out_charset に変換します。

パラメータ

in_charset

入力文字セット。

out_charset

出力文字セット。

文字列 //TRANSLIT を out_charset に追加すると、翻字機能が有効になります。これは、指定された文字集合で 表せない文字を、見た目の似ている別の文字に置き換える機能です。 文字列 //IGNORE を追加すると、指定された文字集合で 表せない文字は黙って切り捨てられます。 それ以外の場合は、str の中に変換できない文字が 出現した時点で変換が打ち切られます。

str

変換する文字列。

返り値

変換された文字列、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 iconv() の例

```
<?php
echo iconv("ISO-8859-1", "UTF-8", "This is a test.");
?>
```

---

ob_iconv_handler

(PHP 4 >= 4.0.5, PHP 5)

ob_iconv_handler — 出力バッファハンドラとして文字エンコーディングを変換する

説明

string **ob_iconv_handler** (string $contents , int $status)

internal_encoding でエンコードされた文字列を output_encoding に変換します。

internal_encoding および output_encoding は、 iconv_set_encoding() または設定ファイル php.ini で定義されている必要があります。

パラメータ

このハンドラのパラメータについての情報は ob_start() を参照ください。

返り値

このハンドラの返り値についての情報は ob_start() を参照ください。

例

Example#1 ob_iconv_handler() の例

```
<?php
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
ob_start("ob_iconv_handler"); //  出力バッファリングを開始
?>
```

参考

- iconv_get_encoding()
- iconv_set_encoding()
- 出力制御関数 (output control)

目次

ID3 関数

導入

これらの関数は、ID3 タグの読み込みや操作を行います。 ID3 タグは MP3 ファイルで使用されており、曲のタイトルや 演奏者、アルバム名、ジャンル、年、トラック番号といった情報が 格納されています。

バージョン 0.2 以降では、ID3 v2.2+ タグのテキストフレームを 抽出できるようになりました。

要件

外部ライブラリを必要としません。

インストール手順

id3 は PECL の一部となっており、PEAR インストーラを用いて インストールすることができます。PHP を id3 サポートつきで コンパイルするには、ソースコードをダウンロードして php-src/ext/id3 に配置し、--enable-id3 を指定して PHP をコンパイルします。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済みの定数

ほとんどの id3 関数は、使用時にタグのバージョンを指定したり、タグの バージョンを結果として返したりします。タグのバージョンを指定するには、 これらの定数を使用してください。

ID3_V1_0 (integer)
ID3_V1_0 は、 ID3 V1.0 タグを利用する場合に用います。このタグには title、artist、album、genre、year および comment といったフィールドが含まれます。
ID3_V1_1 (integer)
ID3_V1_1 は、 ID3 V1.1 タグを利用する場合に用います。このタグには v1.0 タグのすべての情報に加えてトラック番号が含まれます。
ID3_V2_1 (integer)
ID3_V2_1 は、 ID3 V2.1 タグを利用する場合に用います。
ID3_V2_2 (integer)
ID3_V2_2 は、 ID3 V2.2 タグを利用する場合に用います。
ID3_V2_3 (integer)
ID3_V2_3 は、 ID3 V2.3 タグを利用する場合に用います。
ID3_V2_4 (integer)
ID3_V2_4 は、 ID3 V2.4 タグを利用する場合に用います。
ID3_BEST (integer)
ID3_BEST は、 どのバージョンのタグを使用するのかを id3 関数自身に決定させる場合に 用います。

---

id3_get_frame_long_name

(PECL id3:0.2)

id3_get_frame_long_name — ID3v2 フレームの長い名前を取得する

説明

string **id3_get_frame_long_name** (string $frameId)

id3_get_frame_long_name() は、 ID3v2 フレームの長い名前を返します。

パラメータ

frameId

ID3v2 フレーム。

返り値

フレームの長い名前、あるいはエラー時に **FALSE** を返します。

例

Example#1 id3_get_frame_long_name() の例

```
<?php
$longName = id3_get_frame_long_name("TOLY");
echo $longName;
?>
```

上の例の出力は以下となります。

```
Original lyricist(s)/text writer(s)
```

参考

- id3_get_frame_short_name()

---

id3_get_frame_short_name

(PECL id3:0.2)

id3_get_frame_short_name — ID3v2 フレームの短い名前を取得する

説明

string **id3_get_frame_short_name** (string $frameId)

id3_get_frame_short_name() は、 ID3v2 フレームの短い名前を返します。

パラメータ

frameId

ID3v2 フレーム。

返り値

フレームの短い名前、あるいはエラー時に **FALSE** *を返します。*

id3_get_short_name() *が返す値は、* id3_get_tag() *の返す配列で用いられます。*

例

Example#1 id3_get_frame_short_name() の例

```
<?php
$shortName = id3_get_frame_short_name("TOLY");
echo $shortName;
?>
```

上の例の出力は以下となります。

```
originalLyricist
```

参考

- id3_get_frame_long_name()

---

id3_get_genre_id

(PECL id3:0.1-0.2)

id3_get_genre_id — ジャンルの ID を取得する

説明

int **id3_get_genre_id** *(string $genre)*

id3_get_genre_id() *は、ジャンルの ID を返します。*

パラメータ

genre

0 から 147 までの整数値。

返り値

ジャンルの ID あるいはエラー時に **FALSE** *を返します。*

例

Example#1 id3_get_genre_id() の例

```
<?php
$id = id3_get_genre_id("Alternative");
echo $id;
?>
```

上の例の出力は以下となります。

```
20
```

参考

- id3_get_genre_name()
- id3_get_genre_list()

---

id3_get_genre_list

(PECL id3:0.1-0.2)

id3_get_genre_list — 使用可能なジャンルの一覧を取得する

説明

array **id3_get_genre_list** *(void)*

id3_get_genre_list() は、ID3 タグに格納されている すべてのジャンルを配列で返します。この一覧は Eric Kemp によって作成されたもので、後に WinAmp によって拡張されました。

この関数は、選択可能なジャンルの一覧を利用者に提供するのに便利です。 ID3 タグを更新する際には、ジャンルは 0 から 147 までの整数値で 指定しなければなりません。

返り値

ID3 タグに格納されているすべてのジャンルを含む配列を返します。

例

Example#1 id3_get_genre_list() の例

```
<?php
$genres = id3_get_genre_list();
print_r($genres);
?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => Blues
     [1] => Classic Rock
     [2] => Country
     [3] => Dance
     [4] => Disco
     [5] => Funk
     [6] => Grunge
     [7] => Hip-Hop
     [8] => Jazz
     [9] => Metal
     [10] => New Age
     [11] => Oldies
     [12] => Other
     [13] => Pop
     [14] => R&B
     [15] => Rap
     [16] => Reggae
     [17] => Rock
     [18] => Techno
     [19] => Industrial
     [20] => Alternative
     [21] => Ska
     [22] => Death Metal
     [23] => Pranks
     [24] => Soundtrack
     [25] => Euro-Techno
     [26] => Ambient
     [27] => Trip-Hop
     [28] => Vocal
     [29] => Jazz+Funk
     [30] => Fusion
     [31] => Trance
     [32] => Classical
     [33] => Instrumental
     [34] => Acid
     [35] => House
     [36] => Game
     [37] => Sound Clip
     [38] => Gospel
     [39] => Noise
     [40] => Alternative Rock
     [41] => Bass
     [42] => Soul
     [43] => Punk
     [44] => Space
     [45] => Meditative
     [46] => Instrumental Pop
     [47] => Instrumental Rock
     [48] => Ethnic
     [49] => Gothic
     [50] => Darkwave
     [51] => Techno-Industrial
     [52] => Electronic
     [53] => Pop-Folk
     [54] => Eurodance
     [55] => Dream
     [56] => Southern Rock
     [57] => Comedy
     [58] => Cult
     [59] => Gangsta
     [60] => Top 40
     [61] => Christian Rap
     [62] => Pop/Funk
     [63] => Jungle
     [64] => Native US
     [65] => Cabaret
     [66] => New Wave
     [67] => Psychadelic
     [68] => Rave
     [69] => Showtunes
     [70] => Trailer
     [71] => Lo-Fi
     [72] => Tribal
     [73] => Acid Punk
     [74] => Acid Jazz
     [75] => Polka
```

```
    [76] => Retro
    [77] => Musical
    [78] => Rock & Roll
    [79] => Hard Rock
    [80] => Folk
    [81] => Folk-Rock
    [82] => National Folk
    [83] => Swing
    [84] => Fast Fusion
    [85] => Bebob
    [86] => Latin
    [87] => Revival
    [88] => Celtic
    [89] => Bluegrass
    [90] => Avantgarde
    [91] => Gothic Rock
    [92] => Progressive Rock
    [93] => Psychedelic Rock
    [94] => Symphonic Rock
    [95] => Slow Rock
    [96] => Big Band
    [97] => Chorus
    [98] => Easy Listening
    [99] => Acoustic
    [100] => Humour
    [101] => Speech
    [102] => Chanson
    [103] => Opera
    [104] => Chamber Music
    [105] => Sonata
    [106] => Symphony
    [107] => Booty Bass
    [108] => Primus
    [109] => Porn Groove
    [110] => Satire
    [111] => Slow Jam
    [112] => Club
    [113] => Tango
    [114] => Samba
    [115] => Folklore
    [116] => Ballad
    [117] => Power Ballad
    [118] => Rhytmic Soul
    [119] => Freestyle
    [120] => Duet
    [121] => Punk Rock
    [122] => Drum Solo
    [123] => Acapella
    [124] => Euro-House
    [125] => Dance Hall
    [126] => Goa
    [127] => Drum & Bass
    [128] => Club-House
    [129] => Hardcore
    [130] => Terror
    [131] => Indie
    [132] => BritPop
    [133] => Negerpunk
    [134] => Polsk Punk
    [135] => Beat
    [136] => Christian Gangsta
    [137] => Heavy Metal
    [138] => Black Metal
    [139] => Crossover
    [140] => Contemporary C
    [141] => Christian Rock
    [142] => Merengue
    [143] => Salsa
    [144] => Thrash Metal
    [145] => Anime
    [146] => JPop
    [147] => SynthPop
)
```

参考

- id3_get_genre_name()
- id3_get_genre_id()

---

id3_get_genre_name

(PECL id3:0.1-0.2)

id3_get_genre_name — ジャンル ID に対応する名前を取得する

説明

string **id3_get_genre_name** (int $genre_id)

id3_get_genre_name() は、ジャンル ID に対応する名前を返します。

パラメータ

genre_id

0 から 147 までの整数値。

返り値

名前を文字列で返します。

例

Example#1 id3_get_genre_name() の例

```
<?php
$genre = id3_get_genre_name(20);
echo $genre;
?>
```

上の例の出力は以下となります。

```
Alternative
```

参考

- id3_get_genre_list()
- id3_get_genre_id()

---

id3_get_tag

(PECL id3:0.1-0.2)

id3_get_tag — ID3 タグに含まれるすべての情報を取得する

説明

array **id3_get_tag** *(string $filename [, int $version])*

id3_get_tag() *は、指定したファイルの ID3 タグに 含まれるすべての情報を取得するために使用されます。*

パラメータ

filename

MP3 ファイルへのパス。

ファイル名のかわりに、ストリームリソースを渡すことも可能です。

version

MP3 ファイルがバージョン 1.x およびバージョン 2.x の両方のタグを 含んでいる場合に、タグのバージョンを指定します。

*バージョン 0.2 以降、**id3_get_tag()** は ID3 タグのバージョン 2.2、2.3 および 2.4 にも対応するようになりました。 これらのタグに関する情報を取得するには、2 番目の引数に ID3_V2_2、ID3_V2_3 あるいは ID3_V2_4 のいずれかの定数を指定します。 ID3 v2.x タグは、ID3 v1.x タグに比べてはるかに多くの情報を MP3 ファイルに含めることが可能です。*

返り値

title や artist といったキーを含む連想配列を返します。

genre には 0 から 147 までの整数値が格納されています。 id3_get_genre_name() を使用することで、 この数値をジャンル名に変換することが可能です。

例

Example#1 id3_get_tag() の例

```
<?php
$tag = id3_get_tag( "path/to/example.mp3" );
print_r($tag);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [title] => DN-38416
     [artist] => Re:¥Legion
     [album] => Reflections
     [year] => 2004
     [genre] => 19
)
```

Example#2 id3_get_tag() の例

```
<?php
$tag = id3_get_tag( "path/to/example2.mp3", ID3_V2_3 );
print_r($tag);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [copyright] => Dirty Mac
     [originalArtist] => Dirty Mac
     [composer] => Marcus G□ ze
     [artist] => Dirty Mac
     [title] => Little Big Man
     [album] => Demo-Tape
     [track] => 5/12
     [genre] => (17)Rock
     [year] => 2001
)
```

参考

- id3_set_tag()
- id3_remove_tag()
- id3_get_version()

---

id3_get_version

(PECL id3:0.1-0.2)

id3_get_version — ID3 タグのバージョンを取得する

説明

```
int id3_get_version ( string $filename )
```

id3_get_version() は、MP3 ファイル内の ID3 タグのバージョンを取得します。

あるファイルが ID3 v1.1 タグを含んでいるのなら、そのファイルは 常に 1.0 タグも含んでいます。というのも、バージョン 1.1 は単に 1.0 を拡張したものであるからです。

パラメータ

filename

MP3 ファイルへのパス。

ファイル名のかわりに、ストリームリソースを渡すことも可能です。

返り値

MP3 ファイル内の ID3 タグのバージョンを返します。 ID3 v1.x のタグおよび v2.x のタグを同時に保持することも可能なので、 この関数の返す値を利用する際には 定義済みの定数 **ID3_V1_0**、 **ID3_V1_1** および **ID3_V2** とのビット比較を行わなければなりません。

例

Example#1 id3_get_version() の例

```
<?php
$version = id3_get_version( "path/to/example.mp3" );
if ($version & ID3_V1_0) {
    echo "1.x タグを含んでいます\n";
}
if ($version & ID3_V1_1) {
    echo "1.1 タグを含んでいます\n";
}
if ($version & ID3_V2) {
    echo "2.x タグを含んでいます\n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
1.x タグを含んでいます
1.1 タグを含んでいます
```

参考

- id3_set_tag()
- id3_get_tag()
- id3_remove_tag()

id3_remove_tag

(PECL id3:0.1-0.2)

id3_remove_tag — 既存の ID3 タグを削除する

説明

bool **id3_remove_tag** (string $filename [, int $version])

id3_remove_tag() は、ID3 タグに格納されている 情報を削除するために使用されます。

パラメータ

filename

MP3 ファイルへのパス。

ファイル名のかわりに、ストリームリソースを渡すことも可能です。

version

MP3 ファイルにはバージョン 1.x および 2.x のタグを両方含められるので、 ここでタグのバージョンを指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 id3_remove_tag() の例

```
<?php
$result = id3_remove_tag( "path/to/example.mp3", ID3_V1_0 );
if ($result === true) {
    echo "タグが正常に削除されました\n";
}
?>
```

ファイルが書き込み可能であり、かつ 1.0 タグを含んでいる場合は この例の出力は次のようになります。

タグが正常に削除されました

注意

> **注意**: 現時点では **id3_remove_tag()** がサポートしているのは バージョン 1.0 および 1.1 のみです。ファイルが 1.1 のタグを 含んでいるときに 1.0 のタグを削除しようとすると、この (1.1 の) タグが削除されます。というのも、v1.1 は単に 1.0 を拡張したものだからです。

参考

- id3_set_tag()
- id3_get_tag()
- id3_get_version()

id3_set_tag

(PECL id3:0.1-0.2)

id3_set_tag — ID3 タグに格納されている情報を更新する

説明

bool **id3_set_tag** (string $filename , array $tag [, int $version])

id3_set_tag() は、ID3 タグに格納されている情報を 変更するために使用されます。タグが存在しない場合は、新たに追加されます。

パラメータ

filename

MP3 ファイルへのパス。

ファイル名のかわりに、ストリームリソースを渡すことも可能です。

tag

タグのキーと値を含む連想配列。

連想配列の中で使用できるキーは以下のようになります。

連想配列内のキー

キー	とりうる値	使用可能なバージョン
title	最大 30 文字までの文字列	v1.0, v1.1
artist	最大 30 文字までの文字列	v1.0, v1.1
album	最大 30 文字までの文字列	v1.0, v1.1
year	4 桁の数値	v1.0, v1.1
genre	0 から 147 までの整数値	v1.0, v1.1
comment	最大 30 文字 (v1.1 では 28 文字) までの文字列	v1.0, v1.1
track	0 から 255 までの整数値	v1.1

version

MP3 ファイルにはバージョン 1.x および 2.x のタグを両方含められるので、 ここでタグのバージョンを指定します。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 id3_set_tag() の例

```
<?php
$data = array(
                "title" => "Re:Start",
                "artist" => "Re:¥Legion",
                "comment" => "A nice track"
               );
$result = id3_set_tag( "path/to/example.mp3", $data, ID3_V1_0 );
if ($result === true) {
    echo "タグが正常に更新されました\n";
}
?>
```

ファイルが書き込み可能な場合、 この例の出力は次のようになります。

```
タグが正常に更新されました
```

注意

> **注意**: *現時点では **id3_set_tag()** がサポートしているのは バージョン 1.0 および 1.1 のみです。*

参考

- *id3_remove_tag()*
- *id3_get_tag()*
- *id3_get_version()*

目次

IIS 管理関数

導入

この » PECL 拡張モジュールは PHP にバンドルされていません。 この拡張モジュールは Win32 のみで利用可能で、マイクロソフトの インターネットインフォメーションサーバ(IIS)管理用関数を提供します。 この拡張モジュールには、Web サイトや仮想ディレクトリを作成する関数が 含まれており、同時にセキュリティやスクリプトマッピングの設定も 行うことが可能です。これらの関数は、PHP 4 で追加されました。

これらの関数を使用するには、php.ini の中で php_iisfunc.dll DLL を有効にする必要があります。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

IIS_READ (integer)
IIS_WRITE (integer)
IIS_EXECUTE (integer)
IIS_SCRIPT (integer)
IIS_ANONYMOUS (integer)
IIS_BASIC (integer)
IIS_NTLM (integer)
IIS_STARTING (integer)
IIS_STOPPED (integer)
IIS_PAUSED (integer)
IIS_RUNNING (integer)

---

iis_add_server

(No version information available, might be only in CVS)

iis_add_server — 新規に仮想 Web サーバを作成する

説明

int **iis_add_server** (string $path , string $comment , string $server_ip , int $port , string $host_name , int $rights , int $start_server)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_get_dir_security

(No version information available, might be only in CVS)

iis_get_dir_security — ディレクトリのセキュリティを取得する

説明

int **iis_get_dir_security** (int $server_instance , string $virtual_path)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_get_script_map

(No version information available, might be only in CVS)

iis_get_script_map — 指定した拡張子に関して仮想ディレクトリにおけるスクリプトマッピングを取得する

説明

string **iis_get_script_map** (int $server_instance , string $virtual_path , string $script_extension)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_get_server_by_comment

(No version information available, might be only in CVS)

iis_get_server_by_comment — 指定したコメントのインスタンス番号を返す

説明

int **iis_get_server_by_comment** (string $comment)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_get_server_by_path

(No version information available, might be only in CVS)

iis_get_server_by_path — 指定したパスのインスタンス番号を返す

説明

int **iis_get_server_by_path** (string $path)

IIS の各仮想サーバには、インスタンス番号が付けられています。 **iis_get_server_by_path()** は、ルートディレクトリへの 実際のパスからインスタンス番号を検索します。

パラメータ

path

ルートディレクトリへのパス。

返り値

サーバのインスタンス番号を返します。

---

iis_get_server_rights

(No version information available, might be only in CVS)

iis_get_server_rights — サーバの権限を取得する

説明

int **iis_get_server_rights** (int $server_instance , string $virtual_path)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_get_service_state

(No version information available, might be only in CVS)

iis_get_service_state — サービス ID で指定したサービスの状態を取得する

説明

int **iis_get_service_state** (string $service_id)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_remove_server

(No version information available, might be only in CVS)

iis_remove_server — サーバインスタンスで指定した仮想 Web サーバを削除する

説明

int **iis_remove_server** (int $server_instance)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_set_app_settings

(No version information available, might be only in CVS)

iis_set_app_settings — 仮想ディレクトリでのアプリケーションスコープを作成する

説明

int **iis_set_app_settings** (int $server_instance , string $virtual_path , string $application_scope)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

iis_set_dir_security

(No version information available, might be only in CVS)

iis_set_dir_security — ディレクトリのセキュリティを設定する

説明

int **iis_set_dir_security** (int $server_instance , string $virtual_path , int $directory_flags)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

iis_set_script_map

(No version information available, might be only in CVS)

iis_set_script_map — 仮想ディレクトリにスクリプトマッピングを設定する

説明

int **iis_set_script_map** (int $server_instance , string $virtual_path , string $script_extension , string $engine_path , int $allow_scripting)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

iis_set_server_rights

(No version information available, might be only in CVS)

iis_set_server_rights — サーバの権限を設定する

説明

int **iis_set_server_rights** (int $server_instance , string $virtual_path , int $directory_flags)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

iis_start_server

(No version information available, might be only in CVS)

iis_start_server — 仮想 Web サーバを起動する

説明

int **iis_start_server** (int $server_instance)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

iis_start_service

(No version information available, might be only in CVS)

iis_start_service — サービス ID で指定したサービスを起動する

説明

int **iis_start_service** (string $service_id)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

iis_stop_server

(No version information available, might be only in CVS)

iis_stop_server — 仮想 Web サーバを停止する

説明

int **iis_stop_server** (int $server_instance)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iis_stop_service

(No version information available, might be only in CVS)

iis_stop_service — サービス ID で指定したサービスを停止する

説明

int **iis_stop_service** (string $service_id)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

イメージ関数(image)

導入

PHPができることは、HTML出力を生成することだけではありません。PHP は、多くの異なるイメージ形式でイメージファイルを作成したり、操作 したりすることもできます。このイメージ形式には、gif, png, jpg, wbmp, xpmが含まれます。さらに便利なことに、PHPはイメージストリー ムを直接ブラウザに出力することができます。 これを動作されるには、イメージ関数のGDライブラリを指定してPHPをコ ンパイル必要があります。使用したいイメージ形式によっては、GDとPHP は他のライブラリも必要とする可能性があります。

PHPのイメージ関数により JPEG、GIF、 PNG、SWF、 TIFF、JPEG2000イメージの 大きさを知ることができます。

exif 拡張モジュール を使用すると、 JPEG や TIFF 画像のヘッダに保存された情報を扱うことができます。 これにより、デジタルカメラが作成したメタデータを読み込むことができます。 exif 関数は GD ライブラリを必要としません。

> **注意**: イメージの読み込み、書き込み、修正の機能を拡張するには、 要求の節を呼んでください。 デジタルカメラで撮影した画像のメタデータを読み込むには、 上で説明した exif 拡張モジュール が必要です。

要件

GDライブラリ(» http://www.libgd.org/で取得可能)があれば、画像の作成と操 作を行なうこともできます。

操作できるイメージの形式は、インストールされている GDとこれらのイメージフォーマットにアクセスする ためにGDが必要とするその他のライブラリに依存し ます。gd-1.6より前のバージョンのgdは、GIFイメージ画像をサポートし ていますが、PNGはサポートしていません。一方、gd-1.6以降でgd-2.0.28未満の バージョンはPNGをサポートし、GIFはサポートしていません。 GIFサポートは、gd-2.0.28で再度有効になりました。

> **注意**: PHP 4.3以降、GD libの付属版が付属しています。この付属版にはαブ レンディングのようないくつかの機能が追加されています。 この版のコードは、より管理が行き届き、安定しているため、 外部ライブラリの代わりに使用されるべきです。

> **注意**: PHP 6.0.0 では GD 1.x はサポートされなくなりました。 GD 2.0.33 以降が必要となります。

より多くのイメージ形式を処理するために GD を拡張したいと思うかもしれません。

サポートされるイメージ形式

イメージ形式	ダウンロードするライブラリ	注記
gif		gd-1.6より以前またはgd-2.0.28以降のバージョンのGDでのみサポートされます。 *読み込みのみ*のGIFサポートは、PHP 4.3.0 とバンドルされたGDライブラリの組合せで利用可能です。 *書き込み*のサポートは、PHP 4.3.9およびPHP 5.0.1以降で利用可能です。
jpeg-6b	» ftp://ftp.uu.net/graphics/jpeg/	(PHP のビルド前に) jpeg-v6b ライブラリをビルドする際、configure で --enable-shared オプションを指定する必要があります。 そうしないと、PHP をビルドする際の configure で *libjpeg.(a¦so) not found* というエラーが発生します。
png	» http://www.libpng.org/pub/png/libpng.html	gd-1.6以降のバージョンのGDでのみサポートされます。
xpm	» ftp://metalab.unc.edu/pub/Linux/libs/X/!INDEX.html	Xウインドウ環境をインストールしている場合、 このライブラリを既に利用可能と思われます。

別の種類のフォントを処理できるようにGDを拡張したいと思うかもしれ ません。以下にサポートされるフォントライブラリを示します。

サポートされるフォントライブラリ

フォントライブラリ	ダウンロード	注記
FreeType 1.x	» http://www.freetype.org/	PHP 6.0.0 でサポートされなくなりました。
FreeType 2	» http://www.freetype.org/	
T1lib	» ftp://sunsite.unc.edu/pub/Linux/libs/graphics/)	Postscript Type 1 フォントをサポートします。

インストール手順

PHPでGDサポートを有効にするには、configure に--with-gd[=DIR]を指定します。ただし、DIRは GDのベースインストールディレクトリです。PHPにバンドルされている推奨のGDライブラリを使用するには --with-gdを指定します。 GD ライブラリをコンパイルするには、libpng と libjpeg が必要です。

Windowsの場合、GD2 DLL php_gd2.dllをphp.iniに エクステンションとして指定してください。 GD1 DLL php_gd.dllはPHP4.3.2で削除されました。 imagecreatetruecolor()のようにTrueカラーが優先される関数 についてはGD2が必須です。

PHP3においてGDサポートを無効にするには --without-gdを指定してください。

より多くの画像フォーマットを扱えるようにGDの能力を高めるには、 --with-XXXXのような形のオプションを指定します。

サポートされる画像フォーマット

画像フォーマット	configure オプション
jpeg-6b	jpeg-6b をサポートするには --with-jpeg-dir=DIR を指定します。
png	PNGをサポートするには--with-png-dir=DIR を指定します。ただし、libpngはzlibライブラリを 必要とするため、--with-zlib-dir[=DIR] もconfigureオプションに追加する必要があります。
xpm	XPMをサポートするには--with-xpm-dir=DIR を指定します。必要なライブラリをconfigureが見つけられなかった場合は X11ライブラリのパスを追加してください。

***注意**: libpng と共に PHP をコンパイルする際、GD ライブラリとリンクされる同じバージョンを使用する必要があります。*

GDがより多くのフォントを扱えるようにするには --with-XXXXのような形のオプションを指定します。

サポートされるフォントライブラリ

フォントライブラリ	configure オプション
FreeType 1.x	FreeType 1.x をサポートするには --with-ttf[=DIR]を指定します。
FreeType 2	FreeType 2 をサポートするには --with-freetype-dir=DIRを指定します。
T1lib	T1lib (Postscript Type 1 フォント) をサポートするには --with-t1lib[=DIR]を指定します。
ネイティブ TrueType 文字列関数	ネイティブな TrueType 文字列関数 をサポートするには --enable-gd-native-ttfを指定します。

実行時設定

php.ini の設定により動作が変化します。

Image 設定オプション

名前	デフォルト	変更可能	Changelog
gd.jpeg_ignore_warning	"0"	PHP_INI_ALL	PHP 5.1.3 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

gd.jpeg_ignore_warning bool

jpeg2wbmp() や *imagecreatefromjpeg()* が出す警告を無視します。

exif の設定ディレクティブも参照ください。

警告

イメージ関数は非常にメモリを消費します。 memory_limit を十分大きな値にしておくようにしましょう。

リソース型

この拡張モジュールでは、画像 ID および フォント ID の二種類のリソースを定義しています。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

GD_VERSION (*string*)
PHP をコンパイルしたときの GD のバージョン (PHP 5.2.4 以降で利用可能)。
GD_MAJOR_VERSION (*integer*)
PHP をコンパイルしたときの GD のメジャーバージョン (PHP 5.2.4 以降で利用可能)。
GD_MINOR_VERSION (*integer*)
PHP をコンパイルしたときの GD のマイナーバージョン (PHP 5.2.4 以降で利用可能)。
GD_RELEASE_VERSION (*integer*)
PHP をコンパイルしたときの GD のリリースバージョン (PHP 5.2.4 以降で利用可能)。
GD_EXTRA_VERSION (*string*)
PHP をコンパイルしたときの GD の "追加" バージョン (beta/rc..) (PHP 5.2.4 以降で利用可能)。
IMG_GIF (*integer*)
imagetypes() の返り値として使用。
IMG_JPG (*integer*)
imagetypes() の返り値として使用。
IMG_JPEG (*integer*)
imagetypes() の返り値として使用。

> ***注意***: *この定数の値は **IMAGE_JPG** と同じです。*

IMG_PNG (*integer*)
imagetypes() の返り値として使用。
IMG_WBMP (*integer*)
imagetypes() の返り値として使用。
IMG_XPM (*integer*)
imagetypes() の返り値として使用。
IMG_COLOR_TILED (*integer*)
imagecolorallocate() あるいは imagecolorallocatealpha() で割り当てた色のかわりに使用できる、特別な色オプション。
IMG_COLOR_STYLED (*integer*)
imagecolorallocate() あるいは imagecolorallocatealpha() で割り当てた色のかわりに使用できる、特別な色オプション。
IMG_COLOR_BRUSHED (*integer*)
imagecolorallocate() あるいは imagecolorallocatealpha() で割り当てた色のかわりに使用できる、特別な色オプション。
IMG_COLOR_STYLEDBRUSHED (*integer*)
imagecolorallocate() あるいは imagecolorallocatealpha() で割り当てた色のかわりに使用できる、特別な色オプション。
IMG_COLOR_TRANSPARENT (*integer*)
imagecolorallocate() あるいは imagecolorallocatealpha() で割り当てた色のかわりに使用できる、特別な色オプション。
IMG_ARC_ROUNDED (*integer*)
imagefilledarc() 関数で使用するスタイル定数。

> ***注意***: *この定数の値は **IMG_ARC_PIE** と同じです。*

IMG_ARC_PIE (*integer*)
imagefilledarc() 関数で使用するスタイル定数。
IMG_ARC_CHORD (*integer*)
imagefilledarc() 関数で使用するスタイル定数。
IMG_ARC_NOFILL (*integer*)
imagefilledarc() 関数で使用するスタイル定数。
IMG_ARC_EDGED (*integer*)
imagefilledarc() 関数で使用するスタイル定数。
IMG_GD2_RAW (*integer*)
imagegd2() 関数で使用する型定数。
IMG_GD2_COMPRESSED (*integer*)
imagegd2() 関数で使用する型定数。
IMG_EFFECT_REPLACE (*integer*)
imagelayereffect() 関数で使用するアルファブレンディング効果。
IMG_EFFECT_ALPHABLEND (*integer*)
imagelayereffect() 関数で使用するアルファブレンディング効果。
IMG_EFFECT_NORMAL (*integer*)
imagelayereffect() 関数で使用するアルファブレンディング効果。
IMG_EFFECT_OVERLAY (*integer*)
imagelayereffect() 関数で使用するアルファブレンディング効果。
IMG_FILTER_NEGATE (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_GRAYSCALE (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_BRIGHTNESS (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_CONTRAST (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_COLORIZE (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_EDGEDETECT (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_GAUSSIAN_BLUR (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_SELECTIVE_BLUR (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_EMBOSS (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_MEAN_REMOVAL (*integer*)
imagefilter() 関数で使用する特別な GD フィルタ。
IMG_FILTER_SMOOTH (*integer*)

imagefilter() 関数で使用する特別な GD フィルタ。
IMAGETYPE_GIF (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_JPEG (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_PNG (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_SWF (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_PSD (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_BMP (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_WBMP (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_XBM (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_TIFF_II (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_TIFF_MM (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_IFF (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_JB2 (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_JPC (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_JP2 (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_JPX (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_SWC (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数。
IMAGETYPE_ICO (integer)
image_type_to_mime_type() および image_type_to_extension() 関数で使用する画像形式定数 (PHP 5.3.0 以降で利用可能)。
PNG_NO_FILTER (integer)
imagepng() 関数で使用する特別な PNG フィルタ。
PNG_FILTER_NONE (integer)
imagepng() 関数で使用する特別な PNG フィルタ。
PNG_FILTER_SUB (integer)
imagepng() 関数で使用する特別な PNG フィルタ。
PNG_FILTER_UP (integer)
imagepng() 関数で使用する特別な PNG フィルタ。
PNG_FILTER_AVG (integer)
imagepng() 関数で使用する特別な PNG フィルタ。
PNG_FILTER_PAETH (integer)
imagepng() 関数で使用する特別な PNG フィルタ。
PNG_ALL_FILTERS (integer)
imagepng() 関数で使用する特別な PNG フィルタ。

例

Example#1 PHPによるPNGの生成

```
<?php

header("Content-type: image/png");
$string  = $_GET['text'];
$im      = imagecreatefrompng("images/button1.png");
$orange  = imagecolorallocate($im, 220, 210, 60);
$px      = (imagesx($im) - 7.5 * strlen($string)) / 2;
imagestring($im, 3, $px, 9, $string, $orange);
imagepng($im);
imagedestroy($im);

?>
```

この例のスクリプトは、<img src="button.php?text"> のようなタグによりあるページからコールされるものです。 上のbutton.phpスクリプトは、この "text" 文字列を引数と し、この場合は "images/button1.png" である基本イメージ の最上部にこの文字列を描いた後、描画後ののイメージを出力します。 この方法は、ボタンのテキストを変更する度に新規のボタンを生成する 必要があるのを回避するために便利な手法です。 この方法により、動的にイメージボタンを生成できます。

---

gd_info

(PHP 4 >= 4.3.0, PHP 5)

gd_info — 現在インストールされているGDライブラリに関する情報を取得する

説明

array **gd_info** (void)

インストールされている GD ライブラリのバージョンとスペックに関する情報を取得します。

返り値

連想配列を返します。

gd_info() が返す配列の要素

属性	意味
GD Version	string。インストールされている *libgd*のバージョン
Freetype Support	boolean value. **TRUE** の場合Freetypeサポートはインストールされている
Freetype Linkage	string値。Freetypeのリンク方法の情報。 'with freetype', 'with TTF library', 'with unknown library'など。 *Freetype Support* が**TRUE**にセットされている場合にのみ この要素が定義されます。
T1Lib Support	boolean値。 *T1Lib*サポートが含まれている場合に**TRUE**
GIF Read Support	boolean値。 *GIF*画像の *読み込み* がサポートされている場合に**TRUE**
GIF Create Support	boolean値。 *GIF*画像の *生成* がサポートされている場合に**TRUE**
JPG Support	boolean値。 *JPG*サポートが含まれている場合に**TRUE**
PNG Support	boolean値。 *PNG*サポートが含まれている場合に**TRUE**
WBMP Support	boolean値。 *WBMP*サポートが含まれている場合に**TRUE**
XBM Support	boolean値。 *XBM*サポートが含まれている倍に**TRUE**

例

Example#1 gd_info() の使用法

```
<?php
var_dump(gd_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
array(9) {
   ["GD Version"]=>
   string(24) "bundled (2.0 compatible)"
   ["FreeType Support"]=>
   bool(false)
   ["T1Lib Support"]=>
   bool(false)
   ["GIF Read Support"]=>
   bool(true)
   ["GIF Create Support"]=>
   bool(false)
   ["JPG Support"]=>
   bool(false)
   ["PNG Support"]=>
   bool(true)
   ["WBMP Support"]=>
   bool(true)
   ["XBM Support"]=>
   bool(false)
}
```

参考

- *imagepng()*
- *imagejpeg()*
- *imagegif()*
- *imagewbmp()*
- *imagetypes()*

---

getimagesize

(PHP 4, PHP 5)

getimagesize — 画像の大きさを取得する

説明

array ***getimagesize*** *(string $filename [, array &$imageinfo])*

getimagesize() *関数は任意の画像ファイルの大きさを決定し、 ファイルの型と高さ/幅を表す文字列を返します。これらは HTML の IMG タグや HTTP の content type として使用できます。*

getimagesize() *は、 imageinfo パラメータで追加情報を返します。*

> ***注意***: *JPC と JP2 は異なるビット深度のコンポーネントを含むことが可能なことに 注意してください。この場合 "bits" に対する値*

は、 最も大きい深度になります。また、JP2 ファイルは 複数の JPEG 2000 コードストリームを含む場合があります。この場合、**getimagesize()** はファイルのルートから 最初に遭遇するコードストリームに対する値を返します。

注意: アイコンに関する情報は、ビットレートが最大のアイコンから取得します。

パラメータ

filename

このパラメータは、情報を取得したいファイルの名前を指定します。 ローカルファイルへの参照、あるいは（設定で許可されているなら）ストリームを用いたリモートファイルへの参照を指定できます。

imageinfo

オプションのパラメータで、 画像ファイルから何らかの拡張情報を引き出すことが可能です。 現在、この変数にはJPGファイルについて 異なった複数のAPPマーカーが連想配列として返されます。いくつかのプログラムは、 これらのAPPマーカーを画像の中の埋込テキストの情報として使用します。最も一般的な例は、マーカーAPP13に埋込IPTC » http://www.iptc.org/情報が返されることです。 バイナリ形式のマーカーAPP13を読むことができるよう処理するために iptcparse() 関数を使用することができます。

返り値

5 つの要素からなる配列を返します。

0 番目および 1 番目の要素は、それぞれ画像の幅と高さを表します。

注意: 形式によっては、画像を含まないものや複数の画像を含むものがあります。 これらの場合、**getimagesize()** は画像のサイズを適切に決定することができません。このような場合、 **getimagesize()** が返す幅と高さはいずれもゼロとなります。

2 番目の要素は IMAGETYPE_XXX 定数のひとつで、 画像の形式を表します。

3 番目の要素はIMGタグで直接利用できる文字列 height="yyy" width="xxx" です。

mimeは画像のMIMEタイプに一致します。この情報は 画像とともに正しい HTTP Content-type ヘッダを転送するために使用できます。

Example#1 getimagesize() および MIME 型

```
<?php
$size = getimagesize($filename);
$fp = fopen($filename, "rb");
if ($size && $fp) {
    header("Content-type: {$size['mime']}");
    fpassthru($fp);
    exit;
} else {
    // エラー
}
?>
```

channels は RGB 画像の場合に 3、CMYK 画像の場合に 4 です。 bitsはカラーの数です。 しかし、これらの値の存在はちょっと 混乱気味です。例えば、GIFは常に1ピクセルあたり3チャンネル を使用しますが、グローバルカラーテーブルを使ったアニメーションGIFの ピクセルあたりのビット数を計算することはできません。

失敗した場合には **FALSE** を返します。

エラー / 例外

もし filename のイメージにアクセスできない場合、 もしくは有効な画像でない場合、**getimagesize()** は **FALSE** を返し、**E_WARNING** レベルのエラーを発生させます。 読み込み時にエラーが発生した場合は、 **getimagesize()** は **E_NOTICE** レベルのエラーを発生させます。

変更履歴

バージョン	説明
5.3.0	アイコンに対応しました。
5.2.3	読み込み時にエラーが発生した場合のエラーレベルが、 **E_WARNING** から **E_NOTICE** に変わりました。
4.3.2	JPC, JP2, JPX, JB2, XBM, WBMP のサポートが追加されました。
4.3.2	*imageinfo* パラメータで JPEG 2000 のサポートが追加されました。
4.3.0	*bits* と *channels* が他の画像形式でも使用できるようになりました。
4.3.0	*mime* が追加されました。
4.3.0	SWC のサポートが追加されました。
4.2.0	TIFF のサポートが追加されました。
4.0.5	URL のサポートが追加されました。

例

Example#2 getimagesize （ファイル）

```
<?php
list($width, $height, $type, $attr) = getimagesize("img/flag.jpg");
echo "<img src=¥"img/flag.jpg¥" $attr alt=¥"getimagesize() example¥" />";
?>
```

Example#3 getimagesize （URL）

```
<?php
$size = getimagesize("http://www.example.com/gifs/logo.gif");

// ファイル名にスペースが含まれる場合は適切にエンコードしてください
$size = getimagesize("http://www.example.com/gifs/lo%20go.gif");
```

```
?>
```

Example#4 IPTC を返す getimagesize()

```
<?php
$size = getimagesize("testimg.jpg", $info);
if (isset($info["APP13"])) {
    $iptc = iptcparse($info["APP13"]);
    var_dump($iptc);
}
?>
```

注意

注意: **getimagesize()** 関数は GD 画像ライブラリを必要としません。

参考

- image_type_to_mime_type()
- exif_imagetype()
- exif_read_data()
- exif_thumbnail()

---

image_type_to_extension

(PHP 5)

image_type_to_extension — 画像形式からファイルの拡張子を取得する

説明

string **image_type_to_extension** (int $imagetype [, bool $include_dot])

指定した定数 IMAGETYPE_XXX に対応する拡張子を返します。

パラメータ

imagetype

IMAGETYPE_XXX 定数のいずれかひとつ。

include_dot

拡張子の前にドットをつけるかどうか。デフォルトは **TRUE**。

返り値

指定した型に対応する拡張子を文字列で返します。

---

image_type_to_mime_type

(PHP 4 >= 4.3.0, PHP 5)

image_type_to_mime_type — getimagesize, exif_read_data, exif_thumbnail, exif_imagetypeから返される 画像形式のMIMEタイプを取得する

説明

string **image_type_to_mime_type** (int $imagetype)

image_type_to_mime_type() は IMAGETYPE 定数で指定される Mime-Type を取得します。

パラメータ

imagetype

IMAGETYPE_XXX 定数のいずれか。

返り値

戻り値は次のとおりです。

戻り値の定数

imagetype	戻り値
IMAGETYPE_GIF	*image/gif*
IMAGETYPE_JPEG	*image/jpeg*
IMAGETYPE_PNG	*image/png*
IMAGETYPE_SWF	*application/x-shockwave-flash*
IMAGETYPE_PSD	*image/psd*
IMAGETYPE_BMP	*image/bmp*
IMAGETYPE_TIFF_II (intel byte order)	*image/tiff*
IMAGETYPE_TIFF_MM (motorola byte order)	*image/tiff*
IMAGETYPE_JPC	*application/octet-stream*
IMAGETYPE_JP2	*image/jp2*
IMAGETYPE_JPX	*application/octet-stream*
IMAGETYPE_JB2	*application/octet-stream*
IMAGETYPE_SWC	*application/x-shockwave-flash*
IMAGETYPE_IFF	*image/iff*
IMAGETYPE_WBMP	*image/vnd.wap.wbmp*
IMAGETYPE_XBM	*image/xbm*
IMAGETYPE_ICO	*image/vnd.microsoft.icon*

例

Example#1 image_type_to_mime_type （ファイル）

```
<?php
header("Content-type: " . image_type_to_mime_type(IMAGETYPE_PNG));
?>
```

注意

> ***注意**: この関数は GD ライブラリを必要としません。*

参考

- *getimagesize()*
- *exif_imagetype()*
- *exif_read_data()*
- *exif_thumbnail()*

---

image2wbmp

(PHP 4 >= 4.0.5, PHP 5)

image2wbmp — ブラウザまたはファイルにイメージを出力する

説明

*bool **image2wbmp** (resource $image [, string $filename [, int $threshold]])*

***image2wbmp()** は、イメージ im から filename に WBMP ファイルを作成します。*

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存られるファイルへのパス。指定しなかった場合は、生の画像ストリームが直接出力されます。

threshold

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 image2wbmp() の例

```php
<?php

$file = 'php.png';
$image = imagecreatefrompng($file);

header('Content-type: ' . image_type_to_mime_type(IMAGETYPE_WBMP));
image2wbmp($image); // ストリームを直接出力します。

?>
```

注意

注意: WBMP サポートは、GD-1.8 以降で PHP をコンパイルした場合のみ利用可能です。

参考

- imagewbmp()

---

imagealphablending

(PHP 4 >= 4.0.6, PHP 5)

imagealphablending — イメージのブレンドモードを設定する

説明

bool **imagealphablending** (resource $image , bool $blendmode)

imagealphablending() によりTrueColorイメージに 二つの異なる描画モードを使用可能となります。ブレンドモードでは、 全ての描画関数に指定される色のalphaチャネル要素として使用され、 例えばimagesetpixel()では背景色の透過割合を定 義します。結果として、gd は描画色に関してその点に存在する色を自動 的にブレンドし、イメージに結果を保存します。結果のピクセルは、透 明になります。ブレンドモードでない場合、描画色は形式的にそのアル ファチャネル情報にコピーされ、出力ピクセルを置換します。ブレンド モードは、パレットイメージを描画している際には使用できません。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

blendmode

ブレンドモードを有効にするかどうか。デフォルトは **FALSE** です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

---

imageantialias

(PHP 4 >= 4.3.2, PHP 5)

imageantialias — アンチエイリアス機能を使用すべきかどうかを判断する

説明

bool **imageantialias** (resource $image , bool $on)

直線や多角形を高速に描画するためのアンチエイリアス機能を有効にします。 アルファコンポーネントはサポートしていません。ダイレクトブレンド操作を 使用します。truecolor 画像に対してのみ動作します。

thickness および styled はサポートしていません。

背景色が透明な場合にアンチエイリアス機能を使用すると、予期せぬ結果に 終わることがあります。ブレンドメソッドでは、背景色が使用されます。アルファコンポーネントをサポートしていないため、アルファコンポーネントに 基づいたアンチエイリアス手法は使用できません。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

on

アンチエイリアスを有効にするかどうか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数は、PHP がバンドル版の GD ライブラリでコンパイルされている場合のみ使用可能です。

参考

- imagecreatetruecolor()

---

imagearc

(PHP 4, PHP 5)

imagearc — 部分楕円を描画する

説明

bool **imagearc** (resource $image , int $cx , int $cy , int $width , int $height , int $start , int $end , int $color)

imagearc() は、指定した座標を中心とする円弧を描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

cx

中心の x 座標。

cy

中心の y 座標。

width

円弧の幅。

height

円弧の高さ。

start

始点の角度。

end

終点の角度。 0° は 3 時の位置で、そこから時計回りの方向に円弧が描かれます。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagearc() による円の描画

```php
<?php

// 200*200 の画像を作成します
$img = imagecreatetruecolor(200, 200);

// 色を設定します
$white  = imagecolorallocate($img, 255, 255, 255);
$red    = imagecolorallocate($img, 255,   0,   0);
$green  = imagecolorallocate($img,   0, 255,   0);
$blue   = imagecolorallocate($img,   0,   0, 255);

// 頭を描きます
imagearc($img, 100, 100, 200, 200,  0, 360, $white);
// 口を描きます
imagearc($img, 100, 100, 150, 150, 25, 155, $red);
// 左右の目を描きます
imagearc($img,  60,  75,  50,  50,  0, 360, $green);
imagearc($img, 140,  75,  50,  50,  0, 360, $blue);

// 画像をブラウザに出力します
header("Content-type:  image/png");
imagepng($img);

// メモリを解放します
imagedestroy($img);

?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagefilledarc()
- imageellipse()
- imagefilledellipse()

---

imagechar

(PHP 4, PHP 5)

imagechar — 水平に文字を描画する

説明

bool **imagechar** (resource $image , int $font , int $x , int $y , string $c , int $color)

imagechar() は、画像 im 上の座標 x ,y (左上が 0, 0) に color 色で c の最初の文字を描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか (数字が大きなほうが、より大きいフォントに対応します)、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

x

始点の x 座標。

y

始点の y 座標。

c

描画する文字。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagechar() の例

```
<?php

$im = imagecreate(100, 100);

$string = 'PHP';

$bg = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);

// 左上の角に黒で "P" を表示します
imagechar($im, 1, 0, 0, $string, $black);

header('Content-type: image/png');
imagepng($im);

?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagecharup()
- imageloadfont()

---

imagecharup

(PHP 4, PHP 5)

imagecharup — 垂直に文字を描画する

説明

bool **imagecharup** (resource $image , int $font , int $x , int $y , string $c , int $color)

指定した画像 image の指定した位置に、 文字 c を垂直に描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか（数字が大きなほうが、より大きいフォントに対応します）、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

x

始点の x 座標。

y

始点の y 座標。

c

描画する文字。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagecharup() の例

```
<?php

$im = imagecreate(100, 100);

$string = 'Note that the first letter is a N';

$bg = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);

// 白地に黒の "Z" を表示します
imagecharup($im, 3, 10, 10, $string, $black);

header('Content-type: image/png');
imagepng($im);

?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagechar()
- imageloadfont()

---

imagecolorallocate

(PHP 4, PHP 5)

imagecolorallocate — 画像で使用する色を作成する

説明

int **imagecolorallocate** (resource $image , int $red , int $green , int $blue)

指定した RGB を配色とする色の ID を返します。

imagecolorallocate() は image で表される画像上で使用する各々の色を作成する際にコールする必要があります。

> **注意**: **imagecolorallocate()** の最初のコールで パレットをもとにした画像 （imagecreate() を使用して作成した画像）で背景色がセットされます。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

これらの値は 0 から 255 までの整数か、あるいは 0x00 から 0XFF までの 16 進数です。

返り値

色の ID、あるいは作成に失敗したい場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.3 より前	作成に失敗した場合には -1 を返します。

例

Example#1 imagecolorallocate() の例

```
<?php

$im = imagecreate(100, 100);

// 背景色を赤にします
$background = imagecolorallocate($im, 255, 0, 0);

// その他の色を設定します
$white = imagecolorallocate($im, 255, 255, 255);
$black = imagecolorallocate($im, 0, 0, 0);

// 十六進で指定します
$white = imagecolorallocate($im, 0xFF, 0xFF, 0xFF);
$black = imagecolorallocate($im, 0x00, 0x00, 0x00);

?>
```

参考

- imagecolorallocatealpha()
- imagecolordeallocate()

---

imagecolorallocatealpha

(PHP 4 >= 4.3.2, PHP 5)

imagecolorallocatealpha — 画像で使用する色を透過度を指定して作成する

説明

int **imagecolorallocatealpha** (resource $image , int $red , int $green , int $blue , int $alpha)

imagecolorallocatealpha() は、 透明度を指定するパラメータ alpha が追加されている以外は imagecolorallocate() と等価です。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

alpha

0 から 127 までの値。 0 は完全に不透明な状態。 127 は完全に透明な状態を表します。

色のパラメータは、0 から 255 までの整数値か 0x00 から 0xFF までの十六進値を指定します。

返り値

色 ID、あるいは作成に失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.3 より前	作成に失敗した場合には -1 を返します。

例

Example#1 imagecolorallocatealpha() の使用例

```
<?php
$size = 300;
$image=imagecreatetruecolor($size, $size);

// 白い背景で黒いふちどりにします
$back = imagecolorallocate($image, 255, 255, 255);
$border = imagecolorallocate($image, 0, 0, 0);
imagefilledrectangle($image, 0, 0, $size - 1, $size - 1, $back);
imagerectangle($image, 0, 0, $size - 1, $size - 1, $border);

$yellow_x = 100;
$yellow_y = 75;
$red_x    = 120;
$red_y    = 165;
$blue_x   = 187;
$blue_y   = 125;
$radius   = 150;

// alpha 値を指定して色を作成します
$yellow = imagecolorallocatealpha($image, 255, 255, 0, 75);
$red    = imagecolorallocatealpha($image, 255, 0, 0, 75);
$blue   = imagecolorallocatealpha($image, 0, 0, 255, 75);

// 3つの重なる円を描きます
imagefilledellipse($image, $yellow_x, $yellow_y, $radius, $radius, $yellow);
imagefilledellipse($image, $red_x, $red_y, $radius, $radius, $red);
imagefilledellipse($image, $blue_x, $blue_y, $radius, $radius, $blue);

// 正しいヘッダを出力するのを忘れないように!
header('Content-type: image/png');

// 最後に、結果を出力します
imagepng($image);
imagedestroy($image);
?>
```

上の例の出力は、たとえば以下のようになります。

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

参考

- imagecolorallocate()
- imagecolordeallocate()

---

imagecolorat

(PHP 4, PHP 5)

imagecolorat — ピクセルの色のインデックスを取得する

説明

int **imagecolorat** (resource $image , int $x , int $y)

image で指定された画像上の 特定位置にあるピクセルの色のインデックスを返します。

PHP が GD ライブラリ 2.0 以上とともにコンパイルされておりかつ画像が True カラーイメージである場合、この関数はそのピクセルの RGB 値を整数で返します。 赤、緑、青のそれぞれの値にアクセスするにはビットシフトとマスキングを利用してください:

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

x

点の x 座標。

y

点の y 座標。

返り値

色のインデックスを返します。

例

Example#1 個々の RGB 値へのアクセス

```
<?php
$im = imagecreatefrompng("php.png");
$rgb = imagecolorat($im, 10, 15);
$r = ($rgb >> 16) & 0xFF;
$g = ($rgb >> 8) & 0xFF;
$b = $rgb & 0xFF;
?>
```

上の例の出力は、たとえば以下のようになります。

```
int(119)
int(123)
int(180)
```

参考

- imagecolorset()
- imagecolorsforindex()

---

imagecolorclosest

(PHP 4, PHP 5)

imagecolorclosest — 指定した色に最も近い色のインデックスを取得する

説明

int **imagecolorclosest** (resource $image , int $red , int $green , int $blue)

指定した RGB 値に「近い」 画像パレット中の色のインデックスを返します。

指定した色とパレット上の各色の「距離」は、 RGB 値が三次元空間上の点の座標を表すと考えて計算します。

もし ファイルからイメージを生成した場合、イメージに使用されている色だけが 解決されます。パレットだけに存在する色は解決されません。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

色のパラメータは、0 から 255 までの整数値か 0x00 から 0xFF までの十六進値を指定します。

返り値

画像パレット内で、指定した色にいちばん近い色のインデックスを返します。

参考

- imagecolorexact()

---

imagecolorclosestalpha

(PHP 4 >= 4.0.6, PHP 5)

imagecolorclosestalpha — 指定した色+アルファ値に最も近い色のインデックスを取得する

説明

int **imagecolorclosestalpha** (resource $image , int $red , int $green , int $blue , int $alpha)

指定した RGB 値と alpha レベルに 「近い」画像パレット中の色のインデックスを返します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

alpha

0 から 127 までの値。 0 は完全に不透明な状態。 127 は完全に透明な状態を表します。

色のパラメータは、0 から 255 までの整数値か 0x00 から 0xFF までの十六進値を指定します。

返り値

画像パレット内で、指定した色にいちばん近い色のインデックスを返します。

注意

> ***注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。*

参考

- *imagecolorexactalpha()*

---

imagecolorclosesthwb

(PHP 4 >= 4.0.1, PHP 5)

imagecolorclosesthwb — 指定した色に最も近い色合い、白、黒を有する色のインデックスを得る

説明

*int **imagecolorclosesthwb** (resource $image , int $red , int $green , int $blue)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

---

imagecolordeallocate

(PHP 4, PHP 5)

imagecolordeallocate — イメージの色リソースを開放する

説明

*bool **imagecolordeallocate** (resource $image , int $color)*

imagecolorallocate() または imagecolorallocatealpha() で確保された色を開放します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

color

色 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagecolordeallocate() の使用法

```
<?php
$white = imagecolorallocate($im, 255, 255, 255);
imagecolordeallocate($im, $white);
?>
```

参考

- imagecolorallocate()
- imagecolorallocatealpha()

---

imagecolorexact

(PHP 4, PHP 5)

imagecolorexact — 指定した色のインデックスを取得する

説明

int **imagecolorexact** (resource $image , int $red , int $green , int $blue)

画像パレット中の特定の色のインデックスを返します。

もし ファイルからイメージを生成した場合、イメージに使用されている色だけが 解決されます。パレットだけに存在する色は解決されません。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

返り値

指定した色の、パレット内でのインデックスを返します。 画像パレット中に色が存在しない場合は -1 を返します。

参考

- imagecolorclosest()

---

imagecolorexactalpha

(PHP 4 >= 4.0.6, PHP 5)

imagecolorexactalpha — 指定した色+アルファ値のインデックスを取得する

説明

int **imagecolorexactalpha** (resource $image , int $red , int $green , int $blue , int $alpha)

イメージのパレットで指定した色+アルファ値のインデックスを返します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

alpha

0 から 127 までの値。 0 は完全に不透明な状態。 127 は完全に透明な状態を表します。

色のパラメータは、0 から 255 までの整数値か 0x00 から 0xFF までの十六進値を指定します。

返り値

イメージのパレットにおける、指定した色+アルファ値のインデックスを返します。 指定した色がイメージのパレットに存在しない場合、-1が返されます。

注意

> ***注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。*

参考

- *imagecolorclosestalpha()*

---

imagecolormatch

(PHP 4 >= 4.3.0, PHP 5)

imagecolormatch — パレットイメージの色を True カラーイメージに近づける

説明

*bool **imagecolormatch** (resource $image1 , resource $image2)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

image1

truecolor イメージリンクリソース。

image2

パレットイメージリンクリソース。 image1 と同じ大きさの画像をさします。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

注意

> ***注意**: この関数は、PHP がバンドル版の GD ライブラリでコンパイルされている場合のみ使用可能です。*
>
> ***注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。*

参考

- *imagecreatetruecolor()*

---

imagecolorresolve

(PHP 4, PHP 5)

imagecolorresolve — 指定した色または出来るだけ近い色のインデックスを得る

説明

*int **imagecolorresolve** (resource $image , int $red , int $green , int $blue)*

この関数は、指定した色に関するインデックスを常に返します。 その色そのものまたは出来るだけ近い色へのインデックスが返されます。

もし ファイルからイメージを生成した場合、イメージに使用されている色だけが 解決されます。パレットだけに存在する色は解決されません。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

返り値

色インデックスを返します。

参考

- *imagecolorclosest()*

---

imagecolorresolvealpha

(PHP 4 >= 4.0.6, PHP 5)

imagecolorresolvealpha — 指定した色+アルファ値または最も近い色のインデックスを取得する

説明

*int **imagecolorresolvealpha** (resource $image , int $red , int $green , int $blue , int $alpha)*

この関数は、指定した色のインデックスを必ず返します。そうでない場合、 正確な色または最も近い別の色のどちらかを返します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

alpha

0 から 127 までの値。 0 は完全に不透明な状態。 127 は完全に透明な状態を表します。

色のパラメータは、0 から 255 までの整数値か 0x00 から 0xFF までの十六進値を指定します。

返り値

色インデックスを返します。

注意

> ***注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。*

参考

- *imagecolorclosestalpha()*

---

imagecolorset

(PHP 4, PHP 5)

imagecolorset — 指定したパレットインデックスの色を設定する

説明

*void **imagecolorset** (resource $image , int $index , int $red , int $green , int $blue)*

この関数は、パレット上で指定したインデックス index を 指定した色 color に設定します。 実際にぬりつぶしを実行するオーバーヘッドなしにパレット上の色を使用する画像をぬりつぶしたような効果を得る ことができ、便利です。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

index

パレットのインデックス。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

返り値

値を返しません。

参考

- imagecolorat()

---

imagecolorsforindex

(PHP 4, PHP 5)

imagecolorsforindex — カラーインデックスからカラーを取得する

説明

array **imagecolorsforindex** (resource $image , int $index)

指定したインデックスに対する色を取得します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

index

返り値

指定したカラーインデックス index に対して適当な値からなる red、green、blue をキーとする連想配列を返します。

例

Example#1 imagecolorsforindex() の例

```php
<?php

// 画像をオープンします
$im = imagecreatefrompng('nexen.png');

// 色を取得します
$start_x = 40;
$start_y = 50;
$color_index = imagecolorat($im, $start_x, $start_y);

// 可読形式にします
$color_tran = imagecolorsforindex($im, $color_index);

// どんな内容でしょう?
print_r($color_tran);

?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [red] => 226
    [green] => 222
    [blue] => 252
    [alpha] => 0
)
```

参考

- imagecolorat()
- imagecolorexact()

---

imagecolorstotal

(PHP 4, PHP 5)

imagecolorstotal — 画像パレットの色数を検出する

説明

int **imagecolorstotal** (resource $image)

指定した画像パレットの色数を返します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

返り値

指定した画像パレットの色数を返します。 truecolor 画像の場合は 0 を返します。

参考

- imagecolorat()
- imagecolorsforindex()
- imageistruecolor()

---

imagecolortransparent

(PHP 4, PHP 5)

imagecolortransparent — 透明色を定義する

説明

int **imagecolortransparent** (resource $image [, int $color])

指定した画像 image 上の透明色を設定します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

color

imagecolorallocate() で指定した画像 ID。

返り値

新しい（あるいは指定されなかった場合は現在の）透明色の ID を返します。

注意

注意: 透過性は imagecopymerge() かつ True カラーの場合のみコピーされます。imagecopy() もしくはパレットイメージの場合はコピーされません。

注意: 透明色は画像のプロパティであり、色のプロパティではありません。 ある色を透明色と定義したら、画像上で既にその色で着色されてる領域も 透明になります。

---

imageconvolution

(PHP 5 >= 5.1.0)

imageconvolution — div および offset の係数を使用し、3x3 の畳み込み配列を適用する

説明

bool **imageconvolution** (resource $image , array $matrix , float $div , float $offset)

画像に畳み込み配列を適用します。 指定した係数とオフセットを使用します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

matrix

3x3 の配列。三つの float 値からなる三つの配列の配列。

div

畳み込み結果の除数。正規化で使用します。

offset

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 PHP.net ロゴのエンボス加工

```
<?php
$image = imagecreatefromgif('http://www.php.net/images/php.gif');

$emboss = array(array(2, 0, 0), array(0, -1, 0), array(0, 0, -1));
imageconvolution($image, $emboss, 1, 127);

header('Content-Type: image/png');
imagepng($image, null, 9);
?>
```

上の例の出力は以下となります。

Example#2 ガウス分布のぼかし

```
<?php
$image = imagecreatetruecolor(180,40);

// テキストを書き、画像にガウス分布のぼかしを適用します
imagestring($image, 5, 10, 8, 'Gaussian Blur Text', 0x00ff00);
$gaussian = array(array(1.0, 2.0, 1.0), array(2.0, 4.0, 2.0), array(1.0, 2.0, 1.0));
imageconvolution($image, $gaussian, 16, 0);

// 比較用にもう一度テキストを書きます
imagestring($image, 5, 10, 18, 'Gaussian Blur Text', 0x00ff00);

header('Content-Type: image/png');
imagepng($image, null, 9);
?>
```

上の例の出力は以下となります。

注意

> **注意**: *この関数は、PHP がバンドル版の GD ライブラリでコンパイルされている場合のみ使用可能です。*

参考

- *imagefilter()*

---

imagecopy

(PHP 4, PHP 5)

imagecopy — 画像の一部をコピーする

説明

*bool **imagecopy** (resource $dst_im , resource $src_im , int $dst_x , int $dst_y , int $src_x , int $src_y , int $src_w , int $src_h)*

src_im の一部、つまり、x,y座標 src_x , src_y を基準として幅src_w 、高さ src_h の領域をdst_im にコピーします。指定された領域は、x,y座標 dst_x ,dst_y にコピー されます。

パラメータ

dst_im

コピー先の画像リンクリソース。

src_im

コピー元の画像リンクリソース。

dst_x

コピー先の x 座標。

dst_y

コピー先の y 座標。

src_x

コピー元の x 座標。

src_y

コピー元の y 座標。

src_w

コピー元の幅。

src_h

コピー元の高さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

imagecopymerge

(PHP 4 >= 4.0.1, PHP 5)

imagecopymerge — イメージの一部をコピー、マージする

説明

bool **imagecopymerge** (resource $dst_im , resource $src_im , int $dst_x , int $dst_y , int $src_x , int $src_y , int $src_w , int $src_h , int $pct)

src_im のsrc_x , src_y で始まる幅 src_w 、高さsrc_h の領域を x,y座標で指定した dst_im にコピーします。 定義された部分は、x,y座標、dst_x 、 dst_y にコピーされます。

パラメータ

dst_im

コピー先の画像リンクリソース。

src_im

コピー元の画像リンクリソース。

dst_x

コピー先の x 座標。

dst_y

コピー先の y 座標。

src_x

コピー元の x 座標。

src_y

コピー元の y 座標。

src_w

コピー元の幅。

src_h

コピー元の高さ。

pct

二つの画像は、 0から100の範囲で指定した pct に基づきマージされます。 pct = 0 の時は、何も行われません。 100の場合、この関数の動作は、パレットイメージに対する imagecopy() と同じとなります。その上、 True カラーイメージに対するアルファ透過性を実装しています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

imagecopymergegray

(PHP 4 >= 4.0.6, PHP 5)

imagecopymergegray — グレースケールでイメージの一部をコピー、マージする

説明

bool **imagecopymergegray** (resource $dst_im , resource $src_im , int $dst_x , int $dst_y , int $src_x , int $src_y , int

$src_w , int $src_h , int $pct)

imagecopymergegray() は、 src_im の X,Y座標 src_x , src_y から 始まる幅src_w 、高さ src_h の領域を dst_im にコピーします。 定義された部分は、x、y座標dst_x 、 dst_y にコピーされます。

この関数は imagecopymerge() と同じですが、マー ジをする際に、コピー前にコピー先のピクセルをグレースケールに変換 することにより、コピー先のピクセルをコピー元の色相を維持するとこ ろが異なります。

パラメータ

dst_im

コピー先の画像リンクリソース。

src_im

コピー元の画像リンクリソース。

dst_x

コピー先の x 座標。

dst_y

コピー先の y 座標。

src_x

コピー元の x 座標。

src_y

コピー元の y 座標。

src_w

コピー元の幅。

src_h

コピー元の高さ。

pct

ふたつの画像は、 pct に基づきマージされます。この値の範囲は 0から100までです。pct = 0の場合、何も処理 は行われません。100の場合、この関数は、 imagecopy()と同じ処理を行います。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

imagecopyresampled

(PHP 4 >= 4.0.6, PHP 5)

imagecopyresampled — 再サンプリングを行いイメージの一部をコピー、伸縮する

説明

bool **imagecopyresampled** (resource $dst_image , resource $src_image , int $dst_x , int $dst_y , int $src_x , int $src_y , int $dst_w , int $dst_h , int $src_w , int $src_h)

imagecopyresampled() は、イメージの矩形の部分 を別のイメージにコピーします。同時にピクセル値を滑らかに補間を行い、 このため、特にサイズを小さくした場合には鮮明さが維持されます。

言い換えると、**imagecopyresampled()** は src_image の座標 (src_x ,src_y) にある 幅 src_w 、高さ src_h の矩形領域を受け取って、それを dst_image の座標 (dst_x ,dst_y) にある幅 dst_w 、高さ dst_h の矩形領域に配置します。

コピー元とコピー先の座標、幅、高さが異なる場合には、 適当なイメージ伸縮が行われます。座標は、左上を基準とします。 この関数は、同じイメージ内の領域にコピーする場合にも使用可能です (dst_image が src_image と同じ場合) が、領域が重なる場合の結果は予測できません。

パラメータ

dst_im

コピー先の画像リンクリソース。

src_im

コピー元の画像リンクリソース。

dst_x

コピー先の x 座標。

dst_y

コピー先の y 座標。

src_x

コピー元の x 座標。

src_y

コピー元の y 座標。

dst_w

コピー先の幅。

dst_h

コピー先の高さ。

src_w

コピー元の幅。

src_h

コピー元の高さ。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 単純な例

この例は、イメージをオリジナルの半分のサイズに再サンプルします。

```
<?php
// ファイル
$filename = 'test.jpg';
$percent = 0.5;

// コンテントタイプ
header('Content-type: image/jpeg');

// 新規サイズを取得します
list($width, $height) = getimagesize($filename);
$new_width = $width * $percent;
$new_height = $height * $percent;

// 再サンプル
$image_p = imagecreatetruecolor($new_width, $new_height);
$image = imagecreatefromjpeg($filename);
imagecopyresampled($image_p, $image, 0, 0, 0, 0, $new_width, $new_height, $width, $height);

// 出力
imagejpeg($image_p, null, 100);
?>
```

上の例の出力は、たとえば以下のようになります。

Example#2 イメージを均等に再サンプルする

この例は最大の幅もしくは高さが 200 ピクセルのイメージを表示します。

```
<?php
// ファイル
$filename = 'test.jpg';

// 最大の高さ・幅を設定します
$width = 200;
$height = 200;

// コンテントタイプ
header('Content-type: image/jpeg');

// 新規サイズを取得します
list($width_orig, $height_orig) = getimagesize($filename);

$ratio_orig = $width_orig/$height_orig;

if ($width/$height > $ratio_orig) {
   $width = $height*$ratio_orig;
} else {
   $height = $width/$ratio_orig;
}

// 再サンプル
$image_p = imagecreatetruecolor($width, $height);
$image = imagecreatefromjpeg($filename);
imagecopyresampled($image_p, $image, 0, 0, 0, 0, $width, $height, $width_orig, $height_orig);

// 出力
imagejpeg($image_p, null, 100);
?>
```

上の例の出力は、たとえば以下のようになります。

注意

注意: パレットイメージの制限(255+1色)による問題があります。 カラーの再サンプリングやフィルタリングには通常は255色以上の色が 必要となります。再サンプルするピクセルとその色を計算するために ある種の近似計算が使用されます。パレットに新しい色を割り当てよう として失敗すると、(理論的に)最も近い色が選択されます。 それは必ずしも常に可視色とは限りません。そのため、 空白(あるいは不可視な)といった不可思議な結果がもたらされます。 この問題を回避するには、imagecreatetruecolor()で 生成されるようなTrueカラーイメージを目的のイメージとして 使用してください。

参考

imagecopyresized()

---

imagecopyresized

(PHP 4, PHP 5)

imagecopyresized — 画像の一部をコピーしサイズを変更する

説明

```
bool imagecopyresized ( resource $dst_image , resource $src_image , int $dst_x , int $dst_y , int $src_x , int $src_y , int $dst_w , int $dst_h , int $src_w , int $src_h )
```

imagecopyresized() は指定した画像の矩形部分を 別の画像へコピーします。dst_image はコピー先の イメージ ID、src_image はコピー元のイメージ ID です。

言い換えると、**imagecopyresized()** は src_image の座標 (src_x ,src_y) にある 幅 src_w 、高さ src_h の矩形領域を受け取って、それを dst_image の座標 (dst_x ,dst_y) にある幅 dst_w 、高さ dst_h の矩形領域に配置します。

コピー先とコピー元の座標、幅、高さが異なった場合、画像の一部が 適当に伸縮されます。座標の原点は左上です。(仮に、 dst_image と src_image が 同一であれば)関数は領域のコピーに使うことができますが、領域が 重なったときの結果は予測できません。

パラメータ

dst_im

コピー先の画像リンクリソース。

src_im

コピー元の画像リンクリソース。

dst_x

コピー先の x 座標。

dst_y

コピー先の y 座標。

src_x

コピー元の x 座標。

src_y

コピー元の y 座標。

dst_w

コピー先の幅。

dst_h

コピー先の高さ。

src_w

コピー元の幅。

src_h

コピー元の高さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 イメージをリサイズする

この例はイメージを半分のサイズで表示します。

```
<?php
// ファイルと新規サイズ
$filename = 'test.jpg';
$percent = 0.5;

// コンテントタイプ
header('Content-type: image/jpeg');

// 新規サイズを取得します
list($width, $height) = getimagesize($filename);
```

```
$newwidth  = $width  * $percent;
$newheight  = $height  * $percent;

// 読み込み
$thumb  = imagecreatetruecolor($newwidth, $newheight);
$source  = imagecreatefromjpeg($filename);

// リサイズ
imagecopyresized($thumb, $source, 0, 0, 0, 0, $newwidth, $newheight, $width, $height);

// 出力
imagejpeg($thumb);
?>
```

上の例の出力は、たとえば以下のようになります。

イメージは半分サイズで出力されますが、 imagecopyresampled() を使用するとより良い品質になります。

注意

> **注意**: パレットイメージの制限(255+1 色)による問題があります。 カラーの再サンプリングやフィルタリングには通常は 255 色以上の色が 必要となります。再サンプルするピクセルとその色を計算するために ある種の近似計算が使用されます。パレットに新しい色を割り当てよう として失敗すると、(理論的に)最も近い色が選択されます。 それは必ずしも常に可視色とは限りません。そのため、 空白(あるいは不可視な)といった不可思議な結果がもたらされます。 この問題を回避するには、imagecreatetruecolor() で 生成されるような True カラーイメージを目的のイメージとして 使用してください。

参考

imagecopyresampled()

---

imagecreate

(PHP 4, PHP 5)

imagecreate — パレットを使用する新規画像を作成する

説明

resource **imagecreate** (int $width , int $height)

imagecreate() は、 指定した大きさの空の画像を表す画像 ID を返します。

imagecreatetruecolor() を使うことを推奨します。

パラメータ

width

画像の幅。

height

画像の高さ。

返り値

成功した場合に画像リソース ID、エラー時に **FALSE** を返します。

例

Example#1 新しい GD 画像ストリームの作成および画像の出力

```
<?php
header("Content-type:  image/png");
$im = @imagecreate(110, 20)
     or die("Cannot  Initialize  new  GD  image  stream");
$background_color  = imagecolorallocate($im, 0, 0, 0);
$text_color  = imagecolorallocate($im, 233, 14, 91);
imagestring($im, 1, 5, 5,  "A  Simple  Text  String", $text_color);
imagepng($im);
imagedestroy($im);
?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagedestroy()
- imagecreatetruecolor()

---

imagecreatefromgd2

(PHP 4 >= 4.0.7, PHP 5)

imagecreatefromgd2 — GD2 ファイルまたは URL から新規イメージを生成する

説明

resource **imagecreatefromgd2** (string $filename)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatefromgd2part

(PHP 4 >= 4.0.7, PHP 5)

imagecreatefromgd2part — GD2 ファイルまたは URL の指定した部分から新規イメージを生成する

説明

resource **imagecreatefromgd2part** (string $filename , int $srcX , int $srcY , int $width , int $height)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

srcX

srcY

width

height

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatefromgd

(PHP 4 >= 4.0.7, PHP 5)

imagecreatefromgd — GD ファイルまたは URL から新規イメージを生成する

説明

resource **imagecreatefromgd** (string $filename)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

注意

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatefromgif

(PHP 4, PHP 5)

imagecreatefromgif — ファイルまたは URL から新規画像を作成する

説明

```
resource imagecreatefromgif ( string $filename )
```

imagecreatefromgif() は、 指定したファイル名の画像を表す画像 ID を返します。

デバッグを簡単にするために、次の例ではエラー表示用 GIF を生成しています。

Example#1 作成時のエラーを処理する例

```
<?php
function LoadGif ($imgname)
{
    $im = @imagecreatefromgif ($imgname); /* オープンします */
    if (!$im) { /* 失敗したかどうかを調べます */
        $im = imagecreatetruecolor (150, 30); /* 空の画像を作成します */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
       /* エラーメッセージを出力します */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
header("Content-Type: image/gif");
$img = LoadGif("bogus.image");
imagegif($img);
?>
```

上の例の出力は、たとえば以下のようになります。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

GIF 画像へのパス。

返り値

成功した場合に画像リソース ID、エラー時に **FALSE** を返します。

注意

> **注意**: GIF サポートは GD ライブラリのバージョン 1.6 以降で削除され、 バージョン 2.0.28 で復活しました。 その間のバージョンでは、この関数を使用できません。

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatefromjpeg

(PHP 4, PHP 5)

imagecreatefromjpeg — ファイル又は URL から新規 JPEG 画像を作成する

説明

```
resource imagecreatefromjpeg ( string $filename )
```

imagecreatefromjpeg() は引数 filename から得られる画像を表すイメージIDを返します。

失敗した場合は、**imagecreatefromjpeg()** はエラーメッセージを出力します。この場合、残念なことに ブラウザ上のリンクは壊れてしまいます。デバッグを簡単にするために、 以下の例ではエラー表示用 JPEG を出力しています。

Example#1 作成時のエラーを処理する例

```
<?php
function LoadJpeg($imgname)
{
    $im = @imagecreatefromjpeg($imgname); /* オープンを試みます */
    if (!$im) { /* 失敗したかどうかを調べます */
        $im  = imagecreatetruecolor(150, 30); /* 空の画像を作成します */
        $bgc = imagecolorallocate($im, 255, 255, 255);
        $tc  = imagecolorallocate($im, 0, 0, 0);
        imagefilledrectangle($im, 0, 0, 150, 30, $bgc);
       /* エラーメッセージを出力します */
        imagestring($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
header("Content-Type: image/jpeg");
$img = LoadJpeg("bogus.image");
imagejpeg($img);
?>
```

上の例の出力は、たとえば以下のようになります。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

JPEG 画像へのパス。

返り値

成功した場合に画像リソース ID、エラー時に **FALSE** を返します。

注意

> **注意**: JPEG サポートは、 PHP が GD-1.8 か、それ以降に対して、コンパイルされている場合のみ使用可能です。

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatefrompng

(PHP 4, PHP 5)

imagecreatefrompng — ファイルまたは URL から新規 PNG 画像を作成する

説明

resource **imagecreatefrompng** (string $filename)

imagecreatefrompng()は引数 filename から得られる画像を表す画像 ID を返します。

imagecreatefrompng() はエラー時に空の文字列を 返します。エラーメッセージも出力されますが、この場合、残念なことに ブラウザ上のリンクは壊れてしまいます。 デバッグを簡単にするために以下の例ではエラー表示用 PNG を出力しています。

Example#1 作成時のエラーを処理する例

```
<?php
function LoadPNG($imgname)
{
    $im = @imagecreatefrompng($imgname); /* オープンします */
    if (!$im) { /* 失敗したかどうかを調べます */
        $im  = imagecreatetruecolor(150, 30); /* 空の画像を作成します */
        $bgc = imagecolorallocate($im, 255, 255, 255);
        $tc  = imagecolorallocate($im, 0, 0, 0);
        imagefilledrectangle($im, 0, 0, 150, 30, $bgc);
       /* エラーメッセージを出力します */
        imagestring($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
header("Content-Type: image/png");
$img = LoadPNG("bogus.image");
imagepng($img);
?>
```

上の例の出力は、たとえば以下のようになります。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

PNG 画像へのパス。

返り値

*成功した場合に画像リソース ID、エラー時に **FALSE** を返します。*

注意

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatefromstring

(PHP 4 >= 4.0.4, PHP 5)

imagecreatefromstring — 文字列の中のイメージストリームから新規イメージを作成する

説明

*resource **imagecreatefromstring** (string $data)*

***imagecreatefromstring()** は、指定した文字列から 得られたイメージを表すイメージ ID を返します。もし、PHP が JPEG, PNG, GIF, WBMP, GD2 をサポートするようビルドされている場合、 イメージの種類は自動的に判別されます。*

パラメータ

image

画像データを含む文字列。

返り値

*成功時にはイメージリソースが返されます。 イメージの種類がサポートされていない、データのフォーマットが識別できない、 イメージが壊れておりリロードできないなどの場合は **FALSE** が返されます。*

例

Example#1 imagecreatefromstring() の例

```
<?php
$data = 'iVBORw0KGgoAAAANSUhEUgAAABwAAAASCAMAAAB/2U7WAAAABl'
       . 'BMVEUAAAD///+l2Z/dAAAASUlEQVR4XqWQUQoAIAxC2/0vXZDr'
       . 'EX4IJTRkb7lobNUStXsB0jIXIAMSsQnWlsV+wULF4Avk9fLq2r'
       . '8a5HSE35Q3eO2XP1A1wQkZSgETvDtKdQAAAABJRU5ErkJggg==';
$data = base64_decode($data);

$im = imagecreatefromstring($data);
if ($im !== false) {
    header('Content-Type: image/png');
    imagepng($im);
}
else {
    echo 'エラーが発生しました。';
}
?>
```

上の例の出力は、たとえば以下のようになります。

参考

- *imagecreatefromjpeg()*
- *imagecreatefrompng()*
- *imagecreatefromgif()*
- *imagecreatetruecolor()*

---

imagecreatefromwbmp

(PHP 4 >= 4.0.1, PHP 5)

imagecreatefromwbmp — ファイルまたは URL から新規イメージを作成する

説明

resource **imagecreatefromwbmp** (string $filename)

imagecreatefromwbmp() は、指定したファイル名から 得られたイメージを表すイメージ ID を返します。

ImageCreateFromWBMP() は、エラー時に空の文字列を 返します。エラーメッセージも出力しますが、不幸にしてブラウザに 壊れたリンクとして表示されてしまいます。デバッグを容易にするために 次の例ではエラーを表示する WBMP を作成しています。

Example#1 作成時のエラーを処理する例

```
<?php
function LoadWBMP($imgname)
{
    $im = @imagecreatefromwbmp($imgname); /* オープンします */
    if (!$im) { /* 失敗したかどうかを調べます */
        $im  = imagecreatetruecolor (20, 20); /* 空の画像を作成します */
        $bgc = imagecolorallocate($im, 255, 255, 255);
        $tc  = imagecolorallocate($im, 0, 0, 0);
        imagefilledrectangle($im, 0, 0, 10, 10, $bgc);
       /* エラーメッセージを出力します */
        imagestring($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
?>
```

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

WBMP 画像へのパス。

返り値

成功した場合に画像リソース ID、エラー時に **FALSE** を返します。

注意

注意: WBMP サポートは、 PHP が GD-1.8 か、それ以降に対して、コンパイルされている場合のみ使用可能です。

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatefromxbm

(PHP 4 >= 4.0.1, PHP 5)

imagecreatefromxbm — ファイル又は URL から新規イメージを生成する

説明

resource **imagecreatefromxbm** (string $filename)

imagecreatefromxbm() は、指定した filename から得られたイメージを表すイメージIDを返します。

ヒント

fopen wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

パラメータ

filename

XBM 画像へのパス。

返り値

成功した場合に画像リソース ID、エラー時に **FALSE** を返します。

---

imagecreatefromxpm

(PHP 4 >= 4.0.1, PHP 5)

imagecreatefromxpm — ファイルまたは URL から新規イメージを生成する

説明

resource **imagecreatefromxpm** (string $filename)

imagecreatefromxpm() は、指定した filename から得られたイメージを表すイメージ ID を返します。

ヒント

fopen_wrappers が有効の場合、この関数のファイル名として URL を使用することができます。ファイル名の指定方法に関する詳細は fopen()、サポートされる URL プロトコルの種類については、(例えば)サポートされるプロトコル/ラッパー を参照してください。

パラメータ

filename

XPM 画像へのパス。

返り値

成功した場合に画像リソース ID、エラー時に **FALSE** を返します。

返り値

> **注意**: この関数は、PHP がバンドル版の GD ライブラリでコンパイルされている場合のみ使用可能です。

警告

PHP 4.3.0 より前のバージョンの Windows 版 PHP は、現在この関数に関してリモートファイルアクセス機能をサポートしていません。これは、allow_url_fopen を有効にした場合でも同様です。

---

imagecreatetruecolor

(PHP 4 >= 4.0.6, PHP 5)

imagecreatetruecolor — TrueColor イメージを新規に作成する

説明

resource **imagecreatetruecolor** (int $width , int $height)

imagecreatetruecolor() は、指定した大きさの黒い画像を表す画像 ID を返します。

この関数が定義されているかどうかは、PHP および GD のバージョンによって 変わります。PHP 4.0.6 から 4.1.x の場合、GD モジュールが読み込まれていれば この関数は常に存在しますが、GD2 がインストールされていない環境では PHP が致命的なエラーを発生して終了してしまいます。PHP 4.2.x では、 このような場合にエラーではなく警告を発生させます。他のバージョンの PHP では、 適切なバージョンの GD がインストールされている場合にのみ この関数が定義されます。

パラメータ

width

画像の幅。

height

画像の高さ。

返り値

成功した場合に画像リソース ID、エラー時に **FALSE** を返します。

例

Example#1 新規 GD イメージストリームの作成およびイメージの出力

```
<?php
header ("Content-type: image/png");
$im = @imagecreatetruecolor(120, 20)
      or die("Cannot Initialize new GD image stream");
$text_color = imagecolorallocate($im, 233, 14, 91);
imagestring($im, 1, 5, 5,  "A Simple Text String", $text_color);
imagepng($im);
imagedestroy($im);
?>
```

上の例の出力は、たとえば以下のようになります。

注意

> **注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。
>
> **注意**: この関数では GIF ファイルフォーマットは使用できません。

参考

- imagedestroy()

- imagecreate()

---

imagedashedline

(PHP 4, PHP 5)

imagedashedline — 破線を描画する

説明

bool **imagedashedline** (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color)

これは古い関数です。代わりに imagesetstyle() と imageline() の組み合せを使用してください。

---

imagedestroy

(PHP 4, PHP 5)

imagedestroy — 画像を破棄する

説明

bool **imagedestroy** (resource $image)

imagedestroy() は画像 image を保持するメモリを解放します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

imageellipse

(PHP 4 >= 4.0.6, PHP 5)

imageellipse — 楕円を描画する

説明

bool **imageellipse** (resource $image , int $cx , int $cy , int $width , int $height , int $color)

指定した座標を中心とする楕円を描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

cx

中心の x 座標。

cy

中心の y 座標。

width

楕円の幅。

height

楕円の高さ。

color

楕円の色を、 imagecolorallocate() で作成した画像 ID で指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imageellipse() の例

```php
<?php

// 空の画像を生成します
$image = imagecreatetruecolor(400, 300);

// 背景色を塗ります
$bg = imagecolorallocate($image, 0, 0, 0);

// 楕円の色を選択します
$col_ellipse = imagecolorallocate($image, 255, 255, 255);

// 楕円を描画します
imageellipse($image, 200, 150, 300, 200, $col_ellipse);

// 画像を出力します
header("Content-type: image/png");
imagepng($image);

?>
```

上の例の出力は、たとえば以下のようになります。

注意

注意: この関数は GD 2.0.2 以降を必要とします。

参考

- imagefilledellipse()
- imagearc()

---

imagefill

(PHP 4, PHP 5)

imagefill — 塗り潰す

説明

bool **imagefill** (resource $image , int $x , int $y , int $color)

指定した座標 (左上が 0, 0 です) から、指定した色 color で image を塗りつぶします。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

x

開始位置の x 座標。

y

開始位置の y 座標。

color

塗りつぶし色。 imagecolorallocate() で作成した色 ID です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagefill() の例

```php
<?php

$im = imagecreatetruecolor(100, 100);

// 背景色を赤に設定します
$red = imagecolorallocate($im, 255, 0, 0);
imagefill($im, 0, 0, $red);

header('Content-type: image/png');
imagepng($im);
imagedestroy($im);
?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagecolorallocate()

---

imagefilledarc

(PHP 4 >= 4.0.6, PHP 5)

imagefilledarc — 部分楕円を描画し、塗りつぶす

説明

bool **imagefilledarc** (resource $image , int $cx , int $cy , int $width , int $height , int $start , int $end , int $color , int $style)

指定した image の指定した座標を中心とする、 部分楕円を描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

cx

中心の x 座標。

cy

中心の y 座標。

width

弧の幅。

height

弧の高さ。

start

弧の開始角度。

end

弧の終了角度。 0° は三時の方向で、そこから時計回りに数えます。

color

imagecolorallocate() で作成した色 ID。

style

次の選択肢のビット和。

1. **IMG_ARC_PIE**
2. **IMG_ARC_CHORD**
3. **IMG_ARC_NOFILL**
4. **IMG_ARC_EDGED**

IMG_ARC_PIE および **IMG_ARC_CHORD** は相反します。 **IMG_ARC_CHORD** は、 開始角と終了角を直線で結ぶだけですが、**IMG_ARC_PIE** は、角を丸めます。**IMG_ARC_NOFILL** は、弧と弦が縁どられ塗りつぶされないことを指定します。 **IMG_ARC_EDGED** は、 **IMG_ARC_NOFILL** と共に指定することにより、 開始角と終端角は中心と結ばれます。これは、(塗りつぶすよりも) 「パイの切れ端」を縁どる良い方法です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 3D 風のパイを作成する

```
<?php

// 画像を作成します
$image  = imagecreatetruecolor(100, 100);

// 色を割り当てます
$white     = imagecolorallocate($image, 0xFF, 0xFF, 0xFF);
$gray      = imagecolorallocate($image, 0xC0, 0xC0, 0xC0);
$darkgray  = imagecolorallocate($image, 0x90, 0x90, 0x90);
$navy      = imagecolorallocate($image, 0x00, 0x00, 0x80);
$darknavy  = imagecolorallocate($image, 0x00, 0x00, 0x50);
$red       = imagecolorallocate($image, 0xFF, 0x00, 0x00);
$darkred   = imagecolorallocate($image, 0x90, 0x00, 0x00);

// 3D 効果を作成します
for ($i = 60; $i > 50; $i--) {
   imagefilledarc($image, 50, $i, 100, 50, 0, 45, $darknavy, IMG_ARC_PIE);
   imagefilledarc($image, 50, $i, 100, 50, 45, 75 , $darkgray, IMG_ARC_PIE);
   imagefilledarc($image, 50, $i, 100, 50, 75, 360 , $darkred, IMG_ARC_PIE);
}

imagefilledarc($image, 50, 50, 100, 50, 0, 45, $navy, IMG_ARC_PIE);
imagefilledarc($image, 50, 50, 100, 50, 45, 75 , $gray, IMG_ARC_PIE);
imagefilledarc($image, 50, 50, 100, 50, 75, 360 , $red, IMG_ARC_PIE);
```

```
// 画像を出力します
header('Content-type:  image/png');
imagepng($image);
imagedestroy($image);
?>
```

上の例の出力は、たとえば以下のようになります。

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

---

imagefilledellipse

(PHP 4 >= 4.0.6, PHP 5)

imagefilledellipse — 塗りつぶされた楕円を描画する

説明

bool **imagefilledellipse** (resource $image , int $cx , int $cy , int $width , int $height , int $color)

指定した image の指定した座標を中心として楕円を描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

cx

中心の x 座標。

cy

中心の y 座標。

width

楕円の幅。

height

楕円の高さ。

color

塗りつぶし色。 imagecolorallocate() で作成した色 ID です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagefilledellipse() の例

```
<?php

// 空の画像を作成します
$image  = imagecreatetruecolor(400, 300);

// 背景色を塗ります
$bg = imagecolorallocate($image, 0, 0, 0);

// 楕円の色を選択します
$col_ellipse  = imagecolorallocate($image, 255, 255, 255);

// 白い楕円を描画します
imagefilledellipse($image, 200, 150, 300, 200, $col_ellipse);

// 画像を出力します
header("Content-type:  image/png");
imagepng($image);

?>
```

上の例の出力は、たとえば以下のようになります。

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

参考

- imageellipse()
- imagefilledarc()

---

imagefilledpolygon

(PHP 4, PHP 5)

imagefilledpolygon — 塗りつぶした多角形を描画する

説明

bool **imagefilledpolygon** (resource $image , array $points , int $num_points , int $color)

imagefilledpolygon() は画像 image 上に塗りつぶした多角形を生成します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

points

多角形の頂点の座標 x および y を含む配列。

num_points

頂点の総数。 3 以上である必要があります。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagefilledpolygon() の例

```
<?php
// 多角形の点の配列を準備します
$values = array(
              40,  50,  // Point 1 (x, y)
              20,  240, // Point 2 (x, y)
              60,  60,  // Point 3 (x, y)
              240, 20,  // Point 4 (x, y)
              50,  40,  // Point 5 (x, y)
              10,  10   // Point 6 (x, y)
              );

// 画像を生成します
$image = imagecreatetruecolor(250, 250);

// いくつかの色
$bg    = imagecolorallocate($image, 200, 200, 200);
$blue = imagecolorallocate($image, 0, 0, 255);

// 多角形を描画します
imagefilledpolygon($image, $values, 6, $blue);

// 画像を出力します
header('Content-type:  image/png');
imagepng($image);
imagedestroy($image);
?>
```

上の例の出力は、たとえば以下のようになります。

---

imagefilledrectangle

(PHP 4, PHP 5)

imagefilledrectangle — 塗りつぶした矩形を描画する

説明

bool **imagefilledrectangle** (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color)

色 color で塗りつぶした矩形を、 指定した image 上に作成します。 開始位置と終了位置を指定します。0, 0 が画像の左上角を表します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

x1

開始位置の x 座標。

y1

開始位置の y 座標。

x2

終了位置の x 座標。

y2

終了位置の y 座標。

color

塗りつぶし色。 imagecolorallocate() で作成した色 ID です。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

---

imagefilltoborder

(PHP 4, PHP 5)

imagefilltoborder — 特定色で塗りつぶす

説明

bool **imagefilltoborder** (resource $image , int $x , int $y , int $border , int $color)

imagefilltoborder()は、 border で指定した色を境界色として塗りつぶし を行います。(x ,y)が 塗りつぶしの始点(左上が0, 0)で、領域内を color 色で塗りつぶします。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

x

開始位置の x 座標。

y

開始位置の y 座標。

border

境界色。 imagecolorallocate() で作成した色 ID です。

color

塗りつぶし色。 imagecolorallocate() で作成した色 ID です。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

---

imagefilter

(PHP 5)

imagefilter — 画像にフィルタを適用する

説明

bool **imagefilter** (resource $image , int $filtertype [, int $arg1 [, int $arg2 [, int $arg3 [, int $arg4]]]])

imagefilter() は、指定したフィルタ filtertype を image に適用します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filtertype

filtertype は、以下のいずれかです。

- **IMG_FILTER_NEGATE**: 画像の色を反転させます。
- **IMG_FILTER_GRAYSCALE**: 画像をグレースケールに変換します。
- **IMG_FILTER_BRIGHTNESS**: 画像の輝度を変更します。 輝度レベルを *arg1* で設定します。
- **IMG_FILTER_CONTRAST**: 画像のコントラストを 変更します。コントラストのレベルを *arg1* で設定します。
- **IMG_FILTER_COLORIZE**: **IMG_FILTER_GRAYSCALE** と似ていますが、 色を指定することが可能です。 *arg1* 、*arg2* および *arg3* を使用して *red* 、*blue* 、 *green* の値を指定します。また *arg4* を使用して *alpha* チャネルの値を指定します。 各値の範囲は *0* から *255* までです。
- **IMG_FILTER_EDGEDETECT**: エッジを検出し、 画像のエッジを強調します。
- **IMG_FILTER_EMBOSS**: 画像にエンボス処理を行います。
- **IMG_FILTER_GAUSSIAN_BLUR**: ガウス分布を 使用して画像をぼかします。
- **IMG_FILTER_SELECTIVE_BLUR**: 画像をぼかします。
- **IMG_FILTER_MEAN_REMOVAL**: 平均を除去し、 「スケッチ風の」効果を得ます。
- **IMG_FILTER_SMOOTH**: 画像を滑らかにします。 滑らかさのレベルを *arg1* で指定します。

arg1

arg2

arg3

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.2.5	**IMG_FILTER_COLORIZE** でアルファチャネルをサポートするようになりました。

例

Example#1 imagefilter() グレースケールの例

```
<?php
$im = imagecreatefrompng('dave.png');
if ($im && imagefilter($im, IMG_FILTER_GRAYSCALE)) {
    echo 'Image converted to grayscale.';
    imagepng($im, 'dave.png');
} else {
    echo 'Conversion to grayscale failed.';
}

imagedestroy($im);
?>
```

Example#2 imagefilter() 輝度の例

```
<?php
$im = imagecreatefrompng('sean.png');
if ($im && imagefilter($im, IMG_FILTER_BRIGHTNESS, 20)) {
    echo 'Image brightness changed.';
    imagepng($im, 'sean.png');
} else {
    echo 'Image brightness change failed.';
}

imagedestroy($im);
?>
```

Example#3 imagefilter() 単色化の例

```
<?php
$im = imagecreatefrompng('philip.png');

/* R, G, B, so 0, 255, 0 is green */
if ($im && imagefilter($im, IMG_FILTER_COLORIZE, 0, 255, 0)) {
    echo 'Image successfully shaded green.';
    imagepng($im, 'philip.png');
} else {
    echo 'Green shading failed.';
}

imagedestroy($im);
?>
```

注意

注意: この関数は、*PHP* がバンドル版の *GD* ライブラリでコンパイルされている場合のみ使用可能です。

参考

- *imageconvolution()*

---

imagefontheight

(PHP 4, PHP 5)

imagefontheight — フォントの高さを取得する

説明

int **imagefontheight** (int $font)

指定されたフォントの文字の高さをピクセル単位で返します。

パラメータ

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか (数字が大きなほうが、より大きいフォントに対応します)、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

返り値

高さをピクセルで返します。

参考

- imagefontwidth()
- imageloadfont()

---

imagefontwidth

(PHP 4, PHP 5)

imagefontwidth — フォントの幅を取得する

説明

int **imagefontwidth** (int $font)

指定されたフォントの文字の幅をピクセル単位で返します。

パラメータ

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか (数字が大きなほうが、より大きいフォントに対応します)、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

返り値

幅をピクセルで返します。

参考

- imagefontheight()
- imageloadfont()

---

imageftbbox

(PHP 4 >= 4.0.7, PHP 5)

imageftbbox — freetype2 によるフォントを用いたテキストを囲む箱を取得する

説明

array **imageftbbox** (float $size , float $angle , string $font_file , string $text [, array $extrainfo])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

size

angle

font_file

text

extrainfo

返り値

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

注意: この関数は、PHP が FreeType サポート (--with-freetype-dir=DIR) を有効にしてコンパイルされている場合のみ使用可能です。

変更履歴

バージョン	説明
4.3.5	extrainfo がオプションになりました。

---

imagefttext

(PHP 4 >= 4.0.7, PHP 5)

imagefttext — FreeType 2 によるフォントを用いてイメージにテキストを描画する

説明

array **imagefttext** (resource $image , float $size , float $angle , int $x , int $y , int $color , string $font_file , string $text [, array $extrainfo])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

size

使用するフォントのサイズ (ポイント数)。

angle

角度 (度数)。0 度は、左から右に読むテキストを表します。 度数を上げていくと、反時計回りに回転します。たとえば、 90 度の場合は下から上に読むテキストとなります。

x

x と y で表す座標が、最初の文字のベースポイント (その文字の左下の角とほぼ等しい点) となります。 imagestring() の場合は x と y で最初の文字の左上の角を指定しており、 たとえば "左上" は 0, 0 となりますが、この関数では異なります。

y

y 座標。これはフォントのベースラインを指定するものであり、 文字の最下端を指定するものではありません。

color

テキストに使用する色のインデックス。 **iamgecolorexact()** を参照ください。

font_file

使用するフォントへのフルパス。

text

画像に挿入するテキスト。

extrainfo

返り値

この関数は、長方形の 4 つの角を表す点の配列を返します。 最初が左下の位置で、そこから反時計回りに回ります。

0 左下の x 座標
1 左下の y 座標
2 右下の x 座標
3 右下の y 座標
4 右上の x 座標
5 右上の y 座標
6 左上の x 座標
7 左上の y 座標

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

注意: この関数は、PHP が FreeType サポート (--with-freetype-dir=DIR) を有効にしてコンパイルされている場合のみ使用可能です。

変更履歴

バージョン	説明

バージョン	説明
4.3.5	*extrainfo* がオプションになりました。

---

imagegammacorrect

(PHP 4, PHP 5)

imagegammacorrect — GD イメージにガンマ補正を適用する

説明

bool ***imagegammacorrect*** *(resource $image , float $inputgamma , float $outputgamma)*

指定した GD 画像 image に 入力ガンマ値、出力ガンマ値を指定してガンマ補正を適用します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

inputgamma

入力ガンマ値。

outputgamma

出力ガンマ値。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

---

imagegd2

(PHP 4 >= 4.0.7, PHP 5)

imagegd2 — GD2 イメージをブラウザまたはファイルに出力する

説明

bool ***imagegd2*** *(resource $image [, string $filename [, int $chunk_size [, int $type]]])*

GD2 イメージを、指定した filename に出力します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存するファイルへのパス。省略したり ***NULL*** *を指定したりした場合は、生の画像ストリームが直接出力されます。*

chunk_size

type

IMG_GD2_RAW *あるいは* ***IMG_GD2_COMPRESSED*** *のいずれかです。デフォルトは* ***IMG_GD2_RAW*** *です。*

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

注意

> ***注意***: *この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。*
>
> ***注意***: *通常 GD2 フォーマットはイメージの一部を高速にロードするために 使用されます。GD2 フォーマットは GD2 互換アプリケーションでのみ使用可能であることに注意してください。*

変更履歴

バージョン	説明
4.3.2	*chunk_size* および *type* が追加されました。

参考

- *imagegd()*

---

imagegd

(PHP 4 >= 4.0.7, PHP 5)

imagegd — GD イメージをブラウザまたはファイルに出力する

説明

bool **imagegd** (resource $image [, string $filename])

GD イメージを、指定した filename に出力します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存するファイルへのパス。省略したり **NULL** を指定したりすると、生の画像ストリームが直接出力されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: 通常 GD フォーマットはイメージの一部を高速にロードするために 使用されます。GD フォーマットは GD 互換アプリケーションでのみ使用可能であることに注意してください。

参考

- imagegd2()

---

imagegif

(PHP 4, PHP 5)

imagegif — ブラウザまたはファイルへ画像を出力する

説明

bool **imagegif** (resource $image [, string $filename])

imagegif() は画像 image から GIF ファイル filename を作成します。 引数 image は imagecreate() あるいは imagecreatefrom* 関数から返されたものです。

画像フォーマットは、imagecolortransparent() を 用いて画像を透明化しない限り GIF87a となります。 透明化した場合の画像フォーマットは GIF89a となります。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存するファイルへのパス。省略したり **NULL** を指定したりした場合は、生の画像ストリームが直接出力されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: GD ライブラリバージョン 1.6 以降で GIF サポートが完全に削除されたので、該当する 版の GD ライブラリではこの関数を使用することは できません。GD ライブラリが GIF を サポートするバージョンを出す 2004 年なかごろ以降にサポートが再開される と期待されています。詳しくは » GD Project のサイトを見てください。
> 以下の短いコードにより、利用可能な GD サポートの型を自動検出して 汎用性のある PHP アプリケーションを書くことが可能です。
> Header("Content-type: image/gif");ImageGIF($im); の部分を、より柔軟性のある このコードに置換してください。

```
<?php
if (function_exists("imagegif")) {
    header("Content-type: image/gif");
    imagegif($im);
} elseif (function_exists("imagejpeg")) {
    header("Content-type: image/jpeg");
    imagejpeg($im, "", 0.5);
} elseif (function_exists("imagepng")) {
    header("Content-type: image/png");
    imagepng($im);
} elseif (function_exists("imagewbmp")) {
    header("Content-type: image/vnd.wap.wbmp");
    imagewbmp($im);
} else {
    die("No image support in this PHP server");
```

```
}
?>
```

注意: バージョン 4.0.2 以降および 3.0.18 以降では、種々のイメージ関数の サポートを調べるために function_exists() の 代わりに関数 imagetypes() を使用することが可能です。

```
<?php
if (imagetypes() & IMG_GIF) {
    header ("Content-type: image/gif");
    imagegif ($im);
} elseif (imagetypes() & IMG_JPG) {
    /* ... etc. */
}
?>
```

参考

- imagepng()
- imagewbmp()
- imagejpeg()
- imagetypes()

---

imagegrabscreen

(PHP 5 >= 5.2.2)

imagegrabscreen — 画面全体をキャプチャする

説明

resource **imagegrabscreen** (void)

画面全体のスクリーンショットを取得します。

返り値

成功した場合に画像リソースの ID、失敗した場合に **FALSE** を返します。

例

Example#1 imagegrabscreen() の例

この例は、現在の画面の状態のスクリーンショットを取得してそれを png 画像として保存するものです。

```
<?php
$im = imagegrabscreen();
imagepng($im, "myscreenshot.png");
?>
```

注意

注意: この関数は Windows 上でしか使用できません。

参考

- imagegrabwindow()

---

imagegrabwindow

(PHP 5 >= 5.2.2)

imagegrabwindow — ウィンドウをキャプチャする

説明

resource **imagegrabwindow** (int $window [, int $client_area])

ウィンドウあるいはそのクライアント領域のキャプチャを、 ウィンドウハンドル（COM インスタンスの HWND プロパティ）を指定して取得します。

パラメータ

window

HWND ウィンドウ ID。

client_area

アプリケーションのクライアント領域を含めるかどうか。

返り値

成功した場合に画像リソースの ID、失敗した場合に **FALSE** を返します。

エラー / 例外

window_handle が無効なウィンドウハンドルである場合に E_NOTICE、 Windows API のバージョンが古すぎる場合に E_WARNING が発生します。

例

Example#1 imagegrabwindow() の例

ウィンドウ (ここでは IE) のキャプチャを行います。

```
<?php
$browser = new COM("InternetExplorer.Application");
$handle = $browser->HWND;
$browser->Visible = true;
$im = imagegrabwindow($handle);
$browser->Quit();
imagepng($im, "iesnap.png");
?>
```

ウィンドウ (ここでは IE) の中身のキャプチャを行います。

```
<?php
$browser = new COM("InternetExplorer.Application");
$handle = $browser->HWND;
$browser->Visible = true;
$browser->Navigate("http://www.libgd.org");

/* Still working? */
while ($browser->Busy) {
    com_message_pump(4000);
}
$im = imagegrabwindow($handle, 0);
$browser->Quit();
imagepng($im, "iesnap.png");
?>
```

注意

> **注意**: この関数は Windows 上でしか使用できません。

参考

- imagegrabscreen()

---

imageinterlace

(PHP 4, PHP 5)

imageinterlace — インターレースを有効もしくは無効にする

説明

int **imageinterlace** (resource $image [, int $interlace])

imageinterlace() は、インターレースビットをon またはoffに切り替えます。

インターレースビットがonでかつその画像がJPEGの場合、 その画像はプログレッシブJPEGとして生成されています。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

interlace

ゼロ以外の場合はインターレース画像となり、 ゼロの場合はインターレースビットを off にします。

返り値

画像のインターレースビットが設定されている場合に 1、 それ以外の場合に 0 を返します。

---

imageistruecolor

(PHP 4 >= 4.3.2, PHP 5)

imageistruecolor — 画像が truecolor かどうか調べる

説明

bool **imageistruecolor** (resource $image)

imageistruecolor() は、 image が truecolor 画像かどうか調べます。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

返り値

image が *truecolor* の場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

注意

注意: この関数は、*GD 2.0.1* 以降を必要とします *(2.0.28* 以降を推奨します*)*。

参考

- *imagecreatetruecolor()*

---

imagejpeg

(PHP 4, PHP 5)

imagejpeg — 画像をブラウザまたはファイルに出力する

説明

bool ***imagejpeg*** *(resource $image [, string $filename [, int $quality]])*

imagejpeg() は、画像 *image* から *JPEG* ファイルを作成します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存するファイルへのパス。省略したり **NULL** を指定したりした場合は、 生の画像ストリームを直接出力します。

quality パラメータを指定するためにこの引数をスキップするには、 **NULL** を指定します。

quality

quality はオプションであり、*0*(品質は最低 ですが、ファイルはより小さい)から*100*(品質は最高ですが、ファイルは 最大)の範囲で指定します。デフォルトは *IJG* 品質値(*75*)です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: *JPEG* サポートは、 *PHP* が *GD-1.8* か、それ以降に対して、コンパイルされている場合のみ使用可能です。

注意: プログレッシブ *JPEG* を出力したい場合には、 *imageinterlace()* でインターレースをセットする必要があります。

参考

- *imagepng()*
- *imagegif()*
- *imagewbmp()*
- *imageinterlace()*
- *imagetypes()*

---

imagelayereffect

(PHP 4 >= 4.3.0, PHP 5)

imagelayereffect — アルファブレンディングフラグを設定し、 *libgd* にバンドルされているレイヤ効果を使用する

説明

bool ***imagelayereffect*** *(resource $image , int $effect)*

アルファブレンディングフラグを設定し、 *libgd* にバンドルされているレイヤ効果を使用します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

effect

以下の定数のいずれか。

IMG_EFFECT_REPLACE
ピクセルの置換を使用します (imagealphablending() に **TRUE** を渡すのと同じです)。
IMG_EFFECT_ALPHABLEND
通常のピクセルブレンディングを使用します (imagealphablending() に **FALSE** を渡すのと同じです)。
IMG_EFFECT_NORMAL
IMG_EFFECT_ALPHABLEND と同じです。
IMG_EFFECT_OVERLAY
オーバーレイを使用すると、背景の黒い部分は黒のまま。 一方背景の白い部分は白のままとなります。 背景のグレーな部分は、前景のピクセルの色となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数は、PHP がバンドル版の GD ライブラリでコンパイルされている場合のみ使用可能です。
>
> **注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

---

imageline

(PHP 4, PHP 5)

imageline — 直線を描画する

説明

bool **imageline** (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color)

imageline() は、指定したふたつの点を結ぶ直線を描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

x1

最初の点の x 座標。

y1

最初の点の y 座標。

x2

二番目の点の x 座標。

y2

二番目の点の y 座標。

color

直線の色。 imagecolorallocate() で作成した色 ID です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 太い線を描画する

```php
<?php

function imagelinethick($image, $x1, $y1, $x2, $y2, $color, $thick = 1)
{
    /* この方法は直行する線の場合のみうまく動作します
      imagesetthickness($image, $thick);
      return imageline($image, $x1, $y1, $x2, $y2, $color);
     */
    if ($thick == 1) {
        return imageline($image, $x1, $y1, $x2, $y2, $color);
    }
    $t = $thick / 2 - 0.5;
    if ($x1 == $x2 || $y1 == $y2) {
        return imagefilledrectangle($image, round(min($x1, $x2) - $t), round(min($y1, $y2) - $t), round(max($x1, $x2) + $
    }
    $k = ($y2 - $y1) / ($x2 - $x1); //y = kx + q
    $a = $t / sqrt(1 + pow($k, 2));
    $points = array(
        round($x1 - (1+$k)*$a), round($y1 + (1-$k)*$a),
        round($x1 - (1-$k)*$a), round($y1 - (1+$k)*$a),
```

```
        round($x2 + (1+$k)*$a), round($y2 - (1-$k)*$a),
        round($x2 + (1-$k)*$a), round($y2 + (1+$k)*$a),
    );
    imagefilledpolygon($image, $points, 4, $color);
    return imagepolygon($image, $points, 4, $color);
}

?>
```

参考

- imagecreatetruecolor()
- imagecolorallocate()

---

imageloadfont

(PHP 4, PHP 5)

imageloadfont — 新しいフォントを読み込む

説明

int **imageloadfont** (string $file)

imageloadfont() はユーザが定義したビットマップを読み込み、 その ID を返します。

パラメータ

file

フォントファイル形式は現在はバイナリで、アーキクチャに依存します。 このため、PHP を実行するマシーンと同一の型の CPU 上でフォントファイルを 生成する必要があります。

フォントファイルのフォーマット

バイト位置	C 言語のデータ型	説明
0-3 バイト目	int	フォント中の文字の数
4-7 バイト目	int	フォント中の最初の文字の値(しばしば 空白を表す 32 となります)
8-11 バイト目	int	各文字のピクセル幅
12-15 バイト目	int	各文字のピクセル高さ
16 バイト目から	char	文字データの配列、各文字のピクセルにつき1バイトで、 総数は(文字数*幅*高さ)バイトです。

返り値

フォント ID を返します。これは常に 5 より大きくなり、 組み込みのフォントと衝突することはありません。 エラー時には **FALSE** を返します。

例

Example#1 imageloadfont の使用法

```
<?php
header("Content-type: image/png");
$im = imagecreatetruecolor(50, 20);
$black = imagecolorallocate($im, 0, 0, 0);
$white = imagecolorallocate($im, 255, 255, 255);
imagefilledrectangle($im, 0, 0, 49, 19, $white);
$font = imageloadfont("04b.gdf");
imagestring($im, $font, 0, 0, "Hello", $black);
imagepng($im);
?>
```

参考

- imagefontwidth()
- imagefontheight()

---

imagepalettecopy

(PHP 4 >= 4.0.1, PHP 5)

imagepalettecopy — あるイメージから他のイメージにパレットをコピーする

説明

void **imagepalettecopy** (resource $destination , resource $source)

imagepalettecopy() は、パレットを source から destination にコピーします。

パラメータ

destination

コピー先の画像リソース。

source

コピー元の画像リソース。

返り値

値を返しません。

---

imagepng

(PHP 4, PHP 5)

imagepng — PNG イメージをブラウザまたはファイルに出力する

説明

bool **imagepng** (resource $image [, string $filename [, int $quality [, int $filters]]])

指定した image から、PNG 画像を出力あるいは保存します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存するファイルへのパス。省略したり **NULL** を設定したりした場合は、 生の画像ストリームを直接出力します。

注意: quality および filters を使用しない場合は、 **NULL** を指定することはできません。

quality

圧縮レベル。0 (圧縮しない) から 9 までの値です。

filters

PNG ファイルの大きさを小さくします。 これはビットマスクフィールドで、定数 PNG_FILTER_XXX の組み合わせを指定します。 **PNG_NO_FILTER** や **PNG_ALL_FILTERS** を使用すると、 全フィルタを一括で無効にしたり有効にしたりできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.3	filters パラメータが追加されました。
5.1.2	quality パラメータが追加されました。

例

```
<?php
$im = imagecreatefrompng("test.png");
imagepng($im);
?>
```

参考

- imagegif()
- imagewbmp()
- imagejpeg()
- imagetypes()

---

imagepolygon

(PHP 4, PHP 5)

imagepolygon — 多角形を描画する

説明

bool **imagepolygon** (resource $image , array $points , int $num_points , int $color)

imagepolygon() は、指定した image に多角形を作成します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

points

多角形の頂点からなる PHP の配列で、 points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1 というようになります。

num_points

頂点の総数。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 imagepolygon() の例

```
<?php
// 空の画像を生成します
$image = imagecreatetruecolor(400, 300);

// 背景色を塗ります
$bg = imagecolorallocate($image, 0, 0, 0);

// 多角形の色を選択します
$col_poly = imagecolorallocate($image, 255, 255, 255);

// 多角形を描画します
imagepolygon($image, array (
        0,   0,
        100, 200,
        300, 200
    ),
    3,
    $col_poly);

// 画像を出力します
header("Content-type: image/png");
imagepng($image);

?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagecreate()
- imagecreatetruecolor()

---

imagepsbbox

(PHP 4, PHP 5)

imagepsbbox — PostScript Type1 フォントを用いてテキスト矩形のバウンディングボックス を指定する

説明

*array **imagepsbbox** (string $text , int $font , int $size [, int $space], int $tightness , float $angle)*

PostScript Type1 フォントを用いてテキスト矩形のバウンディングボックス を指定します。

バウンディングボックスは文字メトリックスから得られる情報を用いて 計算されますが、残念なことに実際に描画される文字列の描画結果とは わずかに異なる傾向があります。角度が0度の場合、全ての方向に1ピク セル分多く必要であると予想することができます。

パラメータ

text

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか (数字が大きなほうが、より大きいフォントに対応します)、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

size

size はピクセルで表します。

space

フォントが占める空間のデフォルト値を変更することが可能です。 この値が元の値に付加されます。また、負の値とすることも可能です。 文字間隔の単位で表されます。 1 単位は文字の矩形の 1/1000 です。

tightness

tightness により文字間の空白の量を制御できます。 この量は元の文字幅に追加され、負の値とすることも可能です。 文字間隔の単位で表されます。 1 単位は文字の矩形の 1/1000 です。

angle

angle は、度で指定します。

返り値

以下の要素を持つ配列を返します。

0 左側の x 座標

1 上側の y 座標

2 右側の x 座標

3 下側の y 座標

注意

> ***注意**: この関数は、PHP が --with-t1lib を指定してコンパイルされている場合のみ使用可能です。*

参考

- *imagepstext()*

---

imagepsencodefont

(PHP 4, PHP 5)

imagepsencodefont — フォントの文字エンコードベクトルを変更する

説明

*bool **imagepsencodefont** (resource $font_index , string $encodingfile)*

ファイルから文字エンコードベクトルをロードし、変更します。 PostScript フォントのデフォルトベクトルは、127以上の位置には文字 がほとんどないので、英語以外の言語を使用する場合には 恐らくこの部分を変更したいと思うことでしょう。

この関数を頻繁に用いている場合には、ずっと優れた方法として 設定ファイル で ps.default_encoding が正しいエンコードファイルを指すようにして エンコード法を定義する方法があります。この場合、 自動的にロードされる全てのフォントは、正しいエンコードとなります。

パラメータ

font_index

encodingfile

このファイルの正しいフォーマットは、T1libs のドキュメントに記述されています。 T1libs には、すぐに使用できるファイルとして IsoLatin1.enc および IsoLatin2.enc が含まれています。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

注意

> ***注意**: この関数は、PHP が --with-t1lib を指定してコンパイルされている場合のみ使用可能です。*

---

imagepsextendfont

(PHP 4, PHP 5)

imagepsextendfont — フォントを展開または圧縮する

説明

*bool **imagepsextendfont** (int $font_index , float $extend)*

フォント(font_index)を展開または圧縮します。 パラメータextend の値が1より小さい場合、 フォントの圧縮が行われます。

パラメータ

font_index

extend

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

注意

注意: この関数は、PHP が --with-t1lib を指定してコンパイルされている場合のみ使用可能です。

---

imagepsfreefont

(PHP 4, PHP 5)

imagepsfreefont — PostScript Type 1 フォント用メモリを解放する

説明

bool **imagepsfreefont** (resource $fontindex)

imagepsfreefont() は PostScript Type 1 フォントが使用したメモリを解放します。

パラメータ

fontindex

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数は、PHP が --with-t1lib を指定してコンパイルされている場合のみ使用可能です。

参考

- imagepsloadfont()

---

imagepsloadfont

(PHP 4, PHP 5)

imagepsloadfont — ファイルから PostScript Type 1 フォントをロードする

説明

resource **imagepsloadfont** (string $filename)

指定した filename から PostScript Type 1 フォントを読み込みます。

パラメータ

filename

返り値

全て正常に処理された場合、有効なフォント ID が返され、後で使用することができます。 それ以外の場合、この関数は **FALSE** を返します。

例

Example#1 imagepsloadfont() の例

```
<?php
header("Content-type: image/png");
$im = imagecreatetruecolor(350, 45);
$black = imagecolorallocate($im, 0, 0, 0);
$white = imagecolorallocate($im, 255, 255, 255);
imagefilledrectangle($im, 0, 0, 349, 44, $white);
$font = imagepsloadfont("bchbi.pfb"); // もしくはあなたのマシンにある独自の .pfb ファイルを指定する
imagepstext($im, "Testing... It worked!", $font, 32, $white, $black, 32, 32);
imagepsfreefont($font);
imagepng($im);
imagedestroy($im);
?>
```

注意

注意: この関数は、PHP が --with-t1lib を指定してコンパイルされている場合のみ使用可能です。

参考

- imagepsfreefont()

---

imagepsslantfont

(PHP 4, PHP 5)

imagepsslantfont — フォントを傾ける

説明

bool **imagepsslantfont** (resource $font_index , float $slant)

指定したフォントを傾けます。

パラメータ

font_index

slant

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数は、PHP が --with-t1lib を指定してコンパイルされている場合のみ使用可能です。

---

imagepstext

(PHP 4, PHP 5)

imagepstext — PostScript Type1 フォントを用いて画像の上に文字列を描く

説明

array **imagepstext** (resource $image , string $text , resource $font , int $size , int $foreground , int $background , int $x , int $y [, int $space [, int $tightness [, float $angle [, int $antialias_steps]]]])

PostScript Type1 フォントを使用して、画像の上にテキストを描画します。

動作に関して不明な場合は、フォント及びその測り方に関する PostScipt ドキュメントを参照ください。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

text

書き出すテキスト。

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか (数字が大きなほうが、より大きいフォントに対応します)、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

size

size はピクセルで表します。

foreground

テキストの色。

background

アンチエイリアス時にフェードアウトする色。 background 色のピクセルは描画されないので、 背景画像が無地である必要はありません。

x

最初の文字の左下隅の x 座標。

y

最初の文字の左下隅の y 座標。

space

フォントが占める空間のデフォルト値を変更することが可能です。 この値が元の値に付加されます。また、負の値とすることも可能です。 文字間隔の単位で表され、1 単位が文字矩形の 1/1000 となります。

tightness

tightness により文字間の空白の量を制御できます。 この量は元の文字幅に追加され、負の値とすることも可能です。 文字間隔の単位で表され、1 単位が文字矩形の 1/1000 となります。

angle

angle は、度で表します。

antialias_steps

antialias_steps によりアンチエイリアスを行 うテキストの色数を制御することが可能です。指定できるのは 4 および 16 です。 20 より小さな大きさのテキストには、見易くするために大きい方を推奨します。 より大きなフォントでは、計算負荷がより少ない 4 を使用してください。

返り値

この関数は、以下の要素からなる配列を返します。
0 左下の x 座標
1 左下の y 座標
2 右上の x 座標
3 右上の y 座標

注意

注意: この関数は、PHP が --with-t1lib を指定してコンパイルされている場合のみ使用可能です。

参考

- imagepsbbox()

---

imagerectangle

(PHP 4, PHP 5)

imagerectangle — 矩形を描画する

説明

bool **imagerectangle** (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color)

imagerectangle() は、指定した座標から始まる矩形を作成します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

x1

左上の x 座標。

y1

左上の y 座標。 0, 0 が画像の左上隅を表します。

x2

右下の x 座標。

y2

右下の y 座標。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

imagerotate

(PHP 4 >= 4.3.0, PHP 5)

imagerotate — 指定された角度で画像を回転する

説明

resource **imagerotate** (resource $source_image , float $angle , int $bgd_color [, int $ignore_transparent])

画像 source_image を、 angle で指定された角度だけ回転します。

回転の中心は画像の中心です。回転された画像はスケールダウンされるので、 回転された画像全体は対象画像にフィットします。縁はクリップされません。

パラメータ

source_image

元画像のリンク。

angle

回転角度。

bgd_color

回転後、カバーされない部分の色。

ignore_transparent

ゼロ以外を指定すると、透過色は無視されます（その他の場合は保持されます）。

返り値

変更履歴

バージョン	説明
5.1.0	*ignore_transparent* が追加されました。

例

Example#1 画像を 180 度回転する

この例は、画像を 180 度 - 上下逆さまに回転します

```
<?php
// ファイルと回転角
$filename = 'test.jpg';
$degrees = 180;

// コンテントタイプ
header('Content-type: image/jpeg');

// 読み込み
$source = imagecreatefromjpeg($filename);

// 回転
$rotate = imagerotate($source, $degrees, 0);

// 出力
imagejpeg($rotate);
?>
```

上の例の出力は、たとえば以下のようになります。

注意

注意: この関数は、PHP がバンドル版の GD ライブラリでコンパイルされている場合のみ使用可能です。

---

imagesavealpha

(PHP 4 >= 4.3.2, PHP 5)

imagesavealpha — PNG 画像を保存する際に（単一色の透過設定ではない）完全な アルファチャネル情報を保存するフラグを設定する

説明

bool **imagesavealpha** (resource $image , bool $saveflag)

imagesavealpha() は PNG 画像を保存する際に（単一色の透過設定ではない）完全な アルファチャネル情報を保存するフラグを設定します。

これを使用するためには、アルファブレンディングを解除する必要があります (imagealphablending($im, false))。

アルファチャネルはすべてのブラウザでサポートされているわけではありません。 ブラウザ上での表示で問題が発生した場合は、アルファチャネルに対応している ブラウザ（例: 最新の Mozilla）でスクリプトを実行しなおしてみましょう。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

saveflag

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

参考

- imagealphablending()

---

imagesetbrush

(PHP 4 >= 4.0.6, PHP 5)

imagesetbrush — 線の描画用にブラシイメージを設定する

説明

bool **imagesetbrush** (resource $image , resource $brush)

imagesetbrush() は、特別な色 **IMG_COLOR_BRUSHED** または **IMG_COLOR_STYLEDBRUSHED** で描画される際に (imageline() や imagepolygon() のような) 全ての線描画関数で使用されるブラシイメージを設定します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

brush

画像リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: ブラシの使用が終った際には特別な処理は不要ですが、 ブラシイメージを破棄する場合には、新たにブラシイメージを設定するまでは、 色 **IMG_COLOR_BRUSHED** または **IMG_COLOR_STYLEDBRUSHED** を使用するべきではありません。

---

imagesetpixel

(PHP 4, PHP 5)

imagesetpixel — 点を生成する

説明

bool **imagesetpixel** (resource $image , int $x , int $y , int $color)

imagesetpixel() は、指定した座標にピクセルを描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

x

x 座標。

y

y 座標。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imagesetpixel() の例

A random drawing that ends with a regular picture.

```
<?php

$x = 200;
$y = 200;

$gd = imagecreatetruecolor($x, $y);

$corners[0] = array('x' => 100, 'y' =>  10);
$corners[1] = array('x' =>   0, 'y' => 190);
$corners[2] = array('x' => 200, 'y' => 190);

$red = imagecolorallocate($gd, 255, 0, 0);

for ($i = 0; $i < 100000; $i++) {
  imagesetpixel($gd, round($x),round($y), $red);
  $a = rand(0, 2);
  $x = ($x + $corners[$a]['x']) / 2;
  $y = ($y + $corners[$a]['y']) / 2;
}
```

```
header('Content-Type:  image/png');
imagepng($gd);

?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagecreatetruecolor()
- imagecolorallocate()

---

imagesetstyle

(PHP 4 >= 4.0.6, PHP 5)

imagesetstyle — 線描画用のスタイルを設定する

説明

bool **imagesetstyle** (resource $image , array $style)

imagesetstyle() は、特別な IMG_COLOR_STYLED または色を有するイメージの線 IMG_COLOR_STYLEDBRUSHED を描画する際に (imageline() と imagepolygon() のような) 全ての線描画関数で使用されるスタイルを設定します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

style

はピクセルの配列です。定数 **IMG_COLOR_TRANSPARENT** を使用すると、透明なピクセルを追加できます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

以下の例は、キャンバスの左上から右下隅に破線を描画するスクリプトです。

Example#1 imagesetstyle() の例

```
<?php
header("Content-type:  image/jpeg");
$im   = imagecreatetruecolor(100, 100);
$w    = imagecolorallocate($im, 255, 255, 255);
$red = imagecolorallocate($im, 255, 0, 0);

/* 5 ピクセルの赤線と 5 ピクセルの白線の破線を描画します */
$style = array($red, $red, $red, $red, $w, $w, $w, $w, $w);
imagesetstyle($im, $style);
imageline($im, 0, 0, 100, 100, IMG_COLOR_STYLED);

/* imagesetstyle を併用した imagesetbrush() を使用して happy face の線を描画します */
$style = array($w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $w, $red);
imagesetstyle($im, $style);

$brush = imagecreatefrompng("http://www.libpng.org/pub/png/images/smile.happy.png");
$w2 = imagecolorallocate($brush, 255, 255, 255);
imagecolortransparent($brush, $w2);
imagesetbrush($im, $brush);
imageline($im, 100, 0, 0, 100, IMG_COLOR_STYLEDBRUSHED);

imagejpeg($im);
imagedestroy($im);
?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imagesetbrush()
- imageline()

---

imagesetthickness

(PHP 4 >= 4.0.6, PHP 5)

imagesetthickness — 線描画用の線幅を設定する

説明

bool **imagesetthickness** (resource $image , int $thickness)

imagesetthickness() は、長方形、多角形、楕円等を描画する際の線幅を thickness ピクセルに設定します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

thickness

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

---

imagesettile

(PHP 4 >= 4.0.6, PHP 5)

imagesettile — 塗りつぶし用のイメージを設定する

説明

bool **imagesettile** (resource $image , resource $tile)

imagesettile() は、特別な色 **IMG_COLOR_TILED** を指定して塗りつぶされた場合に、 (imagefill() や imagefilledpolygon() のような) 領域塗りつぶし関数で使用されるタイルイメージを設定します。

タイルは、領域を塗りつぶすために繰り返し使用されるイメージです。 全ての GD イメージをタイルとして使用可能で、 imagecolortransparent() でタイルの透過色 ID を設定することにより、 その一部から下の部分が透けて見えるようなタイルを作成することが可能です。

> **注意**: タイルの使用が終った際には、特別な処理は不要ですが、 タイルイメージを破棄する場合には、新たにタイルイメージを設定するまでは、色 **IMG_COLOR_TILED** を使用してはいけません。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

tile

タイルとして使用する画像リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

imagestring

(PHP 4, PHP 5)

imagestring — 文字列を水平に描画する

説明

bool **imagestring** (resource $image , int $font , int $x , int $y , string $string , int $color)

指定した座標に文字列 string を描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか (数字が大きなほうが、より大きいフォントに対応します)、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

x

左上隅の x 座標。

y

左上隅の y 座標。

string

書き出す文字列。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 imagestring() の例

```
<?php
// 100*30 の画像を生成します
$im = imagecreate(100, 30);

// 白色の背景と青色のテキスト
$bg = imagecolorallocate($im, 255, 255, 255);
$textcolor = imagecolorallocate($im, 0, 0, 255);

// 左上に文字列を描画します
imagestring($im, 5, 0, 0, "Hello world!", $textcolor);

// 画像を出力します
header("Content-type: image/png");
imagepng($im);
?>
```

上の例の出力は、たとえば以下のようになります。

参考

- imageloadfont()
- imagettftext()

---

imagestringup

(PHP 4, PHP 5)

imagestringup — 文字列を垂直に描画する

説明

bool **imagestringup** (resource $image , int $font , int $x , int $y , string $string , int $color)

文字列 string を、指定した座標で垂直に描画します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

font

latin2 エンコーディングの組み込みのフォントの場合は 1, 2, 3, 4, 5 のいずれか (数字が大きなほうが、より大きいフォントに対応します)、あるいは imageloadfont() で登録したフォントの識別子のいずれか。

x

左上隅の x 座標。

y

左上隅の y 座標。

sring

書き出す文字列。

color

imagecolorallocate() で作成した色 ID。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- imageloadfont()

---

imagesx

(PHP 4, PHP 5)

imagesx — 画像の幅を取得する

説明

int **imagesx** (resource $image)

指定した画像リソース image の幅を返します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

返り値

image の幅、あるいはエラー時に **FALSE** を返します。

例

Example#1 imagesx() の使用法

```
<?php

// 300*200 の画像を作成します
$img = imagecreatetruecolor(300, 200);

echo imagesx($img); // 300

?>
```

参考

- imagecreatetruecolor()
- getimagesize()
- imagesy()

---

imagesy

(PHP 4, PHP 5)

imagesy — 画像の高さを取得する

説明

int **imagesy** (resource $image)

指定した画像リソース image の高さを返します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

返り値

image の高さ、あるいはエラー時に **FALSE** を返します。

例

Example#1 imagesy() の使用法

```
<?php

// 300*200 の画像を作成します
$img = imagecreatetruecolor(300, 200);

echo imagesy($img); // 200

?>
```

参考

- imagecreatetruecolor()
- getimagesize()

- imagesx()

---

imagetruecolortopalette

(PHP 4 >= 4.0.6, PHP 5)

imagetruecolortopalette — TrueColor イメージをパレットイメージに変換する

説明

bool **imagetruecolortopalette** (resource $image , bool $dither , int $ncolors)

imagetruecolortopalette() は、TrueColorイメー ジをパレットイメージに変換します。この関数のコードは、元々 Independent JPEG Group ライブラリ用に書かれたもので、素晴らしいも のです。このコードは、色をできる限り維持することに加えて、アルファ チャネルに関する情報を出力されるパレットにおいてできるだけ維持す るように修正されています。これは、期待通りにうまくいきません。通 常は、最高の出力品質が保障される TrueColor出力イメージを単に出力す るのが最良の方法です。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

dither

イメージにディザーをかけることを指定します。 **TRUE** の場合はディザーが行われます。 出力はぼやけますが、色の近似はより良くなります。

ncolors

パレットに保持される最大の色数を設定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数は、GD 2.0.1 以降を必要とします (2.0.28 以降を推奨します)。

---

imagettfbbox

(PHP 4, PHP 5)

imagettfbbox — TypeType フォントを使用したテキストの bounding box を生成する

説明

array **imagettfbbox** (float $size , float $angle , string $fontfile , string $text)

この関数は TrueType テキストの bounding box をピクセル単位で計算して 返します。

パラメータ

size

ピクセル単位のフォントの大きさ。

angle

測定する text の角度(度単位)。

fontfile

TrueType フォントファイルのファイル名 (URL)。PHP が使用している GD ライブラリのバージョンによっては、'/' から始まり '.ttf' で終わるような ファイル名で探し、またライブラリによって定義されているフォントパスで 探そうと試みます。

text

測定する文字列。

返り値

imagettfbbox() は、テキストの bounding box を 作成するための 4 点を表現する 8 個の要素からなる配列を返します。
0 左下角の X 座標
1 左下角の Y 座標
2 右下角の X 座標
3 右下角の Y 座標
4 右上角の X 座標
5 右上角の Y 座標
6 左上角の X 座標

7 左上角の Y 座標

各点の位置は、 angle にかかわらず text からの相対位置で表されます。 つまり、"左上"はテキストを水平に見た場合の左上の角を意味します。

参考

注意: 本関数は GD ライブラリと » FreeType ライブラリの両方が必要です。

参考

- imagettftext()

---

imagettftext

(PHP 4, PHP 5)

imagettftext — TrueType フォントを使用してテキストを画像に書き込む

説明

array **imagettftext** (resource $image , float $size , float $angle , int $x , int $y , int $color , string $fontfile , string $text)

指定した text を、 TrueType フォントを使用して画像に書き込みます。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

size

フォントサイズ。GD のバージョンに依存し、ピクセルサイズ (GD1) もしくはポイントサイズ (GD2) で指定する必要があります。

angle

度で表される角度。0 度は左から右にテキストを読む方向になります。 0 より大きな値は、時計廻りの回転を表現します。例えば、 90 という値は下から上にテキストを読む方向になります。

x

x と y で与えられた座標は、最初の文字のベースポイント (ほぼ文字の左下角) を定義します。 この仕様は、x と y で最初の文字の右上角を定義する imagestring() と異なっています。 例えば、左上は 0, 0 となります。

y

y 座標。これは文字の最下位置ではなく、 フォントベースラインの位置を指定します。

color

カラーインデックス。負の数を使用した場合、 アンチエイリアス機能がオフになります。 imagecolorallocate() を参照ください。

fontfile

使用する TrueType フォントのパス。

PHP が使用している GD ライブラリのバージョンに依存しており、 fontfile が / で始まらない場合、ファイル名に .ttf が追加され、 ライブラリはライブラリが定義したフォントパスから ファイル名を検索しようとします。

2.0.18 以前のバージョンの GD ライブラリを使用している場合、 異なるフォントファイルに対する 'パスの区切り文字' として セミコロンよりも space 文字が使用されます。 この特徴が意図せずに使用されることで Warning: Could not find/open font という結果になります。 これらの影響があるバージョンに対する唯一の解決方法は、 スペースを含まないパスにフォントを移動させることです。

フォントがスクリプトと同一ディレクトリに存在している場合のほとんどで、 以下のトリックが問題の軽減に繋がります。

```
<?php
// GD の環境変数を設定します
putenv('GDFONTPATH='  . realpath('.'));
        The y-ordinate. This sets the position of the fonts baseline, not
          the very bottom of the character.
// 使用するフォント名を指定します (拡張子 .ttf がないことに注意)
$font = 'SomeFont';
?>
```

text

テキスト文字列を UTF-8 エンコーディングで表したもの。

フォント内で 127 文字目以降の文字にアクセスするために、 (€ のような) 十進数文字参照を含めることができます。 (© のような) 十六進形式は PHP 5.2.0 以降でサポートされます。 UTF-8 エンコーディングされた文字列を直接渡すことができます。

© のような文字エンティティはサポートされません。 html_entity_decode() を使用して、 文字エンティティを UTF-8 文字列にすることを検討してください (html_entity_decode() は、PHP 5.0.0 以降でこの機能をサポートします)。

フォントでサポートされていない文字が文字列で使用されている場合、 その文字は白抜きの矩形に置き換えられます。

返り値

テキストの境界を 構成する 4 点を表す 8 個の要素を有する配列を返します。 返される点は左下、右下、右上、左上の順番となります。 点の座標は、角度によらず text に関する相対座標として表されます。 つまり、"左上"は、text を水平に見た場合の左上の隅を表します。

例

Example#1 imagettftext() の例

この例は、400x30 ピクセルの白地に Arial フォントを用いて、黒字（グレーの影付き）で "Testing..." と書かれた PNG を作成します。

```
<?php
// コンテントタイプを設定します
header("Content-type:  image/png");

// 画像を生成します
$im = imagecreatetruecolor(400, 30);

// いくつかの色を生成します
$white = imagecolorallocate($im, 255, 255, 255);
$grey = imagecolorallocate($im, 128, 128, 128);
$black = imagecolorallocate($im, 0, 0, 0);
imagefilledrectangle($im, 0, 0, 399, 29, $white);

// 描画する文字列
$text = 'Testing...';
// フォント自身のパスでパスを置き換えます
$font = 'arial.ttf';

// テキストに影を付けます
imagettftext($im, 20, 0, 11, 21, $grey, $font, $text);

// テキストを追加します
imagettftext($im, 20, 0, 10, 20, $black, $font, $text);

// imagepng() を使用して imagejpeg() よりもクリアなテキストにします
imagepng($im);
imagedestroy($im);
?>
```

上の例の出力は、たとえば以下のようになります。

注意

注意: この関数は GD ライブラリと » FreeType ライブラリの両方が必要です。

参考

- imagettfbbox()

---

imagetypes

(PHP 4 >= 4.0.2, PHP 5)

imagetypes — この PHP がサポートしている画像形式を返す

説明

int **imagetypes** (void)

現在使用している PHP がサポートする画像形式を返します。

返り値

画像形式に対応するビットフィールドで、 PHP に組み込まれている GD がサポートする画像形式を返します。 返されるビットは次のとおりです。
IMG_GIF | **IMG_JPG** | **IMG_PNG** | **IMG_WBMP** | **IMG_XPM**。

例

Example#1 PNG をサポートしているかどうかの確認

```
<?php
if (imagetypes() & IMG_PNG) {
    echo "PNG をサポートしています";
}
?>
```

---

imagewbmp

(PHP 4 >= 4.0.1, PHP 5)

imagewbmp — ブラウザまたはファイルにイメージを出力する

説明

bool **imagewbmp** (resource $image [, string $filename [, int $foreground]])

imagewbmp() は、指定した image を WBMP にして出力あるいは保存します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存するファイルへのパス。省略したり **NULL** を設定したりした場合は、 生の画像ストリームを直接出力します。

foreground

このパラメータで、 前景の色を指定できます。*imagecolorallocate()* で 取得した *ID* を使用してください。デフォルトの前景色は黒です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: *WBMP* サポートは、*PHP* を *GD-1.8* 以降と組み合わせてコンパイルした 場合にのみ利用可能です。

参考

- *image2wbmp()*
- *imagepng()*
- *imagegif()*
- *imagejpeg()*
- *imagetypes()*

---

imagexbm

(PHP 5)

imagexbm — XBM 画像をブラウザあるいはファイルに出力する

説明

bool ***imagexbm*** *(resource $image , string $filename [, int $foreground])*

imagewbmp() は、指定した *image* を *XBM* にして出力あるいは保存します。

パラメータ

image

imagecreatetruecolor() のような画像作成関数が返す画像リソース。

filename

保存するファイルへのパス。省略したり **NULL** を設定したりした場合は、 生の画像ストリームを直接出力します。

foreground

このパラメータで、 前景の色を指定できます。*imagecolorallocate()* で 取得した *ID* を使用してください。デフォルトの前景色は黒です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数は、*PHP* がバンドル版の *GD* ライブラリでコンパイルされている場合のみ使用可能です。

---

iptcembed

(PHP 4, PHP 5)

iptcembed — バイナリ IPTC データを JPEG イメージに埋めこむ

説明

mixed ***iptcembed*** *(string $iptcdata , string $jpeg_file_name [, int $spool])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

iptcdata

jpeg_file_name

spool

---

iptcparse

(PHP 4, PHP 5)

iptcparse — バイナリの IPTC ブロックのタグをパースする

説明

array **iptcparse** (string $iptcblock)

» IPTC ブロックをパースします。

パラメータ

iptcblock

バイナリ IPTC ブロック。

返り値

タグマーカーをインデックスとし、その値を保持する配列を返します。 エラー時や IPTC データが見つからなかったときは **FALSE** を返します。

参考

- getimagesize() の例

---

jpeg2wbmp

(PHP 4 >= 4.0.5, PHP 5)

jpeg2wbmp — JPEG イメージファイルから WBMP イメージファイルに変換する

説明

bool **jpeg2wbmp** (string $jpegname , string $wbmpname , int $dest_height , int $dest_width , int $threshold)

JPEG ファイルを WBMP ファイルに変換します。

パラメータ

jpegname

JPEG ファイルへのパス。

wbmpname

変換後の WBMP ファイルへのパス。

dest_height

変換後の画像の高さ。

dest_width

変換後の画像の幅。

threshold

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: WBMP サポートは、 PHP が GD-1.8 か、それ以降に対して、コンパイルされている場合のみ使用可能です。

注意: JPEG サポートは、 PHP が GD-1.8 か、それ以降に対して、コンパイルされている場合のみ使用可能です。

参考

- png2wbmp()

---

png2wbmp

(PHP 4 >= 4.0.5, PHP 5)

png2wbmp — PNG イメージファイルから WBMP イメージファイルに変換する

説明

bool ***png2wbmp*** *(string $pngname , string $wbmpname , int $dest_height , int $dest_width , int $threshold)*

PNG ファイルを WBMP ファイルに変換します。

パラメータ

pngname

PNG ファイルへのパス。

wbmpname

変換後の WBMP ファイルへのパス。

dest_height

変換後の画像の高さ。

dest_width

変換後の画像の幅。

threshold

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

注意

> ***注意***: *WBMP サポートは、 PHP が GD-1.8 か、それ以降に対して、コンパイルされている場合のみ使用可能です。*

参考

- *jpeg2wbmp()*

目次

- imagecreatefromgd2 — GD2 ファイルまたは URL から新規イメージを生成する
- imagecreatefromgd2part — GD2 ファイルまたは URL の指定した部分から新規イメージを生成する
- imagecreatefromgd — GD ファイルまたは URL から新規イメージを生成する
- imagecreatefromgif — ファイルまたは URL から新規画像を作成する
- imagecreatefromjpeg — ファイル又は URL から新規 JPEG 画像を作成する
- imagecreatefrompng — ファイルまたは URL から新規 PNG 画像を作成する
- imagecreatefromstring — 文字列の中のイメージストリームから新規イメージを作成する
- imagecreatefromwbmp — ファイルまたは URL から新規イメージを作成する
- imagecreatefromxbm — ファイル又は URL から新規イメージを生成する
- imagecreatefromxpm — ファイルまたは URL から新規イメージを生成する
- imagecreatetruecolor — TrueColor イメージを新規に作成する
- imagedashedline — 破線を描画する
- imagedestroy — 画像を破棄する
- imageellipse — 楕円を描画する
- imagefill — 塗り潰す
- imagefilledarc — 部分楕円を描画し、塗りつぶす
- imagefilledellipse — 塗りつぶされた楕円を描画する
- imagefilledpolygon — 塗りつぶした多角形を描画する
- imagefilledrectangle — 塗りつぶした矩形を描画する
- imagefilltoborder — 特定色で塗りつぶす
- imagefilter — 画像にフィルタを適用する
- imagefontheight — フォントの高さを取得する
- imagefontwidth — フォントの幅を取得する
- imageftbbox — freetype2 によるフォントを用いたテキストを囲む箱を取得する
- imagefttext — FreeType 2 によるフォントを用いてイメージにテキストを描画する
- imagegammacorrect — GD イメージにガンマ補正を適用する
- imagegd2 — GD2 イメージをブラウザまたはファイルに出力する
- imagegd — GD イメージをブラウザまたはファイルに出力する
- imagegif — ブラウザまたはファイルへ画像を出力する
- imagegrabscreen — 画面全体をキャプチャする
- imagegrabwindow — ウィンドウをキャプチャする
- imageinterlace — インターレースを有効もしくは無効にする
- imageistruecolor — 画像が truecolor かどうか調べる
- imagejpeg — 画像をブラウザまたはファイルに出力する
- imagelayereffect — アルファブレンディングフラグを設定し、 libgd にバンドルされているレイヤ効果を使用する
- imageline — 直線を描画する
- imageloadfont — 新しいフォントを読み込む
- imagepalettecopy — あるイメージから他のイメージにパレットをコピーする
- imagepng — PNG イメージをブラウザまたはファイルに出力する
- imagepolygon — 多角形を描画する
- imagepsbbox — PostScript Type1 フォントを用いてテキスト矩形のバウンディングボックス を指定する
- imagepsencodefont — フォントの文字エンコードベクトルを変更する
- imagepsextendfont — フォントを展開または圧縮する
- imagepsfreefont — PostScript Type 1 フォント用メモリを解放する
- imagepsloadfont — ファイルから PostScript Type 1 フォントをロードする
- imagepsslantfont — フォントを傾ける
- imagepstext — PostScript Type1 フォントを用いて画像の上に文字列を描く
- imagerectangle — 矩形を描画する
- imagerotate — 指定された角度で画像を回転する
- imagesavealpha — PNG 画像を保存する際に（単一色の透過設定ではない）完全な アルファチャネル情報を保存するフラグを設定する
- imagesetbrush — 線の描画用にブラシイメージを設定する
- imagesetpixel — 点を生成する
- imagesetstyle — 線描画用のスタイルを設定する
- imagesetthickness — 線描画用の線幅を設定する
- imagesettile — 塗りつぶし用のイメージを設定する
- imagestring — 文字列を水平に描画する
- imagestringup — 文字列を垂直に描画する
- imagesx — 画像の幅を取得する
- imagesy — 画像の高さを取得する
- imagetruecolortopalette — TrueColor イメージをパレットイメージに変換する
- imagettfbbox — TypeType フォントを使用したテキストの bounding box を生成する
- imagettftext — TrueType フォントを使用してテキストを画像に書き込む
- imagetypes — この PHP がサポートしている画像形式を返す
- imagewbmp — ブラウザまたはファイルにイメージを出力する

- imagexbm — XBM 画像をブラウザあるいはファイルに出力する
- iptcembed — バイナリ IPTC データを JPEG イメージに埋めこむ
- iptcparse — バイナリの IPTC ブロックのタグをパースする
- jpeg2wbmp — JPEG イメージファイルから WBMP イメージファイルに変換する
- png2wbmp — PNG イメージファイルから WBMP イメージファイルに変換する

Imagick 画像ライブラリ

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

Imagick は、ImageMagick API を使用して画像の作成や修正を行う ネイティブ PHP 拡張モジュールです。

ImageMagick は、ビットマップ画像の作成や修正を行うソフトウェア群です。 さまざまな (100 をこえる) 形式の画像の読み書きや変換に対応しており、 DPX や EXR、GIF、JPEG、JPEG-2000、PDF、PhotoCD、PNG、Postscript、SVG そして TIFF といった形式を扱うことができます。

Copyright 1999-2007 ImageMagick Studio LLC. ImageMagick Studio LLC は、 フリーソフトウェアによる画像処理ソリューションを提供することを目指す非営利組織です。

例

Imagick は、PHP による画像の操作を、 とても簡単なオブジェクト指向インターフェイスで行います。 画像のサムネイルを作成する簡単な例をごらんください。

Example#1 Imagick によるサムネイルの作成

```
<?php

header('Content-type:  image/jpeg');

$image  = new Imagick('image.jpg');

// 幅あるいは高さに 0 を指定すると、
// 元の画像のアスペクト比を維持します
$image->thumbnailImage(100, 0);

echo $image;

?>
```

SPL および Imagick がサポートするその他のオブジェクト指向機能を使用すると、 ディレクトリ内のすべてのファイルのサイズを変更するのも簡単です (デジタルカメラで撮影した巨大な画像ファイル群をウェブ用に変換するような バッチ処理で有用です)。この例では、リサイズ機能を使用しています。 これは、画像のメタデータを残しておきたいからです。

Example#2 ディレクトリ内のすべての JPG ファイルのサムネイルの作成

```
<?php

$images  = new Imagick(glob('images/*.JPG'));

foreach($images as $image) {

    // 0 を指定することで、thumbnailImage にアスペクト比を維持させています
    $image->thumbnailImage(1024,0);

}

$images->writeImages();

?>
```

定数

(No version information available, might be only in CVS)

定数 — Imagick クラスの定数

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

色形式定数
imagick::COLOR_BLACK (integer)
imagick::COLOR_BLUE (integer)
imagick::COLOR_CYAN (integer)
シアン
imagick::COLOR_GREEN (integer)
緑
imagick::COLOR_RED (integer)
赤
imagick::COLOR_YELLOW (integer)
黄色
imagick::COLOR_MAGENTA (integer)

マゼンタ
imagick::COLOR_OPACITY (integer)
不透明
imagick::COLOR_ALPHA (integer)
アルファ
imagick::COLOR_FUZZ (integer)
ファズ

配置方式の定数
imagick::DISPOSE_UNRECOGNIZED (integer)
認識不可能
imagick::DISPOSE_UNRECOGNIZED (integer)
認識不可能
imagick::DISPOSE_UNDEFINED (integer)
未定義
imagick::DISPOSE_NONE (integer)
なし
imagick::DISPOSE_BACKGROUND (integer)
背景
imagick::DISPOSE_PREVIOUS (integer)
前

合成演算子用定数
imagick::COMPOSITE_OVER (integer)
ある画像を次の画像にかぶせる
imagick::COMPOSITE_IN (integer)
あるレイヤーの内側を別のもので置き換えます。
imagick::COMPOSITE_OUT (integer)
あるレイヤーの外側を別のもので置き換えます。
imagick::COMPOSITE_ATOP (integer)
あるレイヤーの内側を別のものと合成します。
imagick::COMPOSITE_XOR (integer)
元画像のうち対象画像の外側にある部分と、 対象画像のうち元画像の外側にある部分を合成します。
imagick::COMPOSITE_PLUS (integer)
元画像を対象画像に加え、対象画像を置き換えます。
imagick::COMPOSITE_MINUS (integer)
元画像を対象画像から引き、対象画像を置き換えます。
imagick::COMPOSITE_ADD (integer)
非推奨。
imagick::COMPOSITE_SUBTRACT (integer)
非推奨。
imagick::COMPOSITE_DIFFERENCE (integer)
色の値の差。画像を比較する際に有用です。
imagick::COMPOSITE_BUMPMAP (integer)
COMPOSITE_MULTIPLY と同じですが、まず最初に元画像をグレースケールに変換します。
imagick::COMPOSITE_COPY (integer)
単純に、元画像を対象画像の上に置きます。
imagick::COMPOSITE_DISPLACE (integer)
imagick::COMPOSITE_DEFAULT (integer)

モンタージュモード定数
imagick::MONTAGEMODE_FRAME (integer)
imagick::MONTAGEMODE_UNFRAME (integer)
imagick::MONTAGEMODE_CONCATENATE (integer)

スタイル定数
imagick::STYLE_NORMAL (integer)
imagick::STYLE_ITALIC (integer)
imagick::STYLE_OBLIQUE (integer)
imagick::STYLE_ANY (integer)

フィルタ定数
imagick::FILTER_UNDEFINED (integer)
imagick::FILTER_POINT (integer)
imagick::FILTER_BOX (integer)
imagick::FILTER_TRIANGLE (integer)
imagick::FILTER_HERMITE (integer)
imagick::FILTER_HANNING (integer)
imagick::FILTER_HAMMING (integer)
imagick::FILTER_BLACKMAN (integer)
imagick::FILTER_GAUSSIAN (integer)
imagick::FILTER_QUADRATIC (integer)
imagick::FILTER_CUBIC (integer)
imagick::FILTER_CATROM (integer)
imagick::FILTER_MITCHELL (integer)
imagick::FILTER_LANCZOS (integer)
imagick::FILTER_BESSEL (integer)
imagick::FILTER_SINC (integer)

画像形式定数
imagick::IMGTYPE_UNDEFINED (integer)
imagick::IMGTYPE_BILEVEL (integer)
imagick::IMGTYPE_GRAYSCALE (integer)
imagick::IMGTYPE_GRAYSCALEMATTE (integer)
imagick::IMGTYPE_PALETTE (integer)
imagick::IMGTYPE_PALETTEMATTE (integer)
imagick::IMGTYPE_TRUECOLOR (integer)
imagick::IMGTYPE_TRUECOLORMATTE (integer)
imagick::IMGTYPE_COLORSEPARATION (integer)
imagick::IMGTYPE_COLORSEPARATIONMATTE (integer)
imagick::IMGTYPE_OPTIMIZE (integer)

解像度定数
imagick::RESOLUTION_UNDEFINED (integer)
imagick::RESOLUTION_PIXELSPERINCH (integer)
imagick::RESOLUTION_PIXELSPERCENTIMETER (integer)

圧縮定数
imagick::COMPRESSION_UNDEFINED (integer)
imagick::COMPRESSION_NO (integer)
imagick::COMPRESSION_BZIP (integer)

imagick::COMPRESSION_FAX (integer)
imagick::COMPRESSION_GROUP4 (integer)
imagick::COMPRESSION_JPEG (integer)
imagick::COMPRESSION_JPEG2000 (integer)
imagick::COMPRESSION_LOSSLESSJPEG (integer)
imagick::COMPRESSION_LZW (integer)
imagick::COMPRESSION_RLE (integer)
imagick::COMPRESSION_ZIP (integer)

描画定数
imagick::PAINT_POINT (integer)
imagick::PAINT_REPLACE (integer)
imagick::PAINT_FLOODFILL (integer)
imagick::PAINT_FILLTOBORDER (integer)
imagick::PAINT_RESET (integer)

グラビティ定数
imagick::GRAVITY_NORTHWEST (integer)
imagick::GRAVITY_NORTH (integer)
imagick::GRAVITY_NORTHEAST (integer)
imagick::GRAVITY_WEST (integer)
imagick::GRAVITY_CENTER (integer)
imagick::GRAVITY_EAST (integer)
imagick::GRAVITY_SOUTHWEST (integer)
imagick::GRAVITY_SOUTH (integer)
imagick::GRAVITY_SOUTHEAST (integer)

ストレッチ定数
imagick::STRETCH_NORMAL (integer)
imagick::STRETCH_ULTRACONDENSED (integer)
imagick::STRETCH_CONDENSED (integer)
imagick::STRETCH_SEMICONDENSED (integer)
imagick::STRETCH_SEMIEXPANDED (integer)
imagick::STRETCH_EXPANDED (integer)
imagick::STRETCH_EXTRAEXPANDED (integer)
imagick::STRETCH_ULTRAEXPANDED (integer)
imagick::STRETCH_ANY (integer)

配置定数
imagick::ALIGN_UNDEFINED (integer)
imagick::ALIGN_LEFT (integer)
imagick::ALIGN_CENTER (integer)
imagick::ALIGN_RIGHT (integer)

デコレーション定数
imagick::DECORATION_NO (integer)
imagick::DECORATION_UNDERLINE (integer)
imagick::DECORATION_OVERLINE (integer)
imagick::DECORATION_LINETROUGH (integer)

ノイズ定数
imagick::NOISE_UNIFORM (integer)
imagick::NOISE_GAUSSIAN (integer)
imagick::NOISE_MULTIPLICATIVEGAUSSIAN (integer)
imagick::NOISE_IMPULSE (integer)
imagick::NOISE_LAPLACIAN (integer)
imagick::NOISE_POISSON (integer)

チャネル定数
imagick::CHANNEL_UNDEFINED (integer)
imagick::CHANNEL_RED (integer)
imagick::CHANNEL_GRAY (integer)
imagick::CHANNEL_CYAN (integer)
imagick::CHANNEL_GREEN (integer)
imagick::CHANNEL_MAGENTA (integer)
imagick::CHANNEL_BLUE (integer)
imagick::CHANNEL_YELLOW (integer)
imagick::CHANNEL_ALPHA (integer)
imagick::CHANNEL_OPACITY (integer)
imagick::CHANNEL_MATTE (integer)
imagick::CHANNEL_BLACK (integer)
imagick::CHANNEL_INDEX (integer)
imagick::CHANNEL_ALL (integer)

メトリック定数
imagick::METRIC_UNDEFINED (integer)
imagick::METRIC_MEANABSOLUTEERROR (integer)
imagick::METRIC_MEANSQUAREERROR (integer)
imagick::METRIC_PEAKABSOLUTEERROR (integer)
imagick::METRIC_PEAKSIGNALTONOISERATIO (integer)
imagick::METRIC_ROOTMEANSQUAREDERROR (integer)

ピクセル定数
imagick::PIXEL_CHAR (integer)
imagick::PIXEL_DOUBLE (integer)
imagick::PIXEL_FLOAT (integer)
imagick::PIXEL_INTEGER (integer)
imagick::PIXEL_LONG (integer)
imagick::PIXEL_QUANTUM (integer)
imagick::PIXEL_SHORT (integer)

評価演算子定数
imagick::EVALUATE_UNDEFINED (integer)
imagick::EVALUATE_ADD (integer)
imagick::EVALUATE_AND (integer)
imagick::EVALUATE_DIVIDE (integer)
imagick::EVALUATE_LEFTSHIFT (integer)
imagick::EVALUATE_MAX (integer)
imagick::EVALUATE_MIN (integer)
imagick::EVALUATE_MULTIPLY (integer)
imagick::EVALUATE_OR (integer)

imagick::EVALUATE_RIGHTSHIFT (integer)
imagick::EVALUATE_SET (integer)
imagick::EVALUATE_SUBTRACT (integer)
imagick::EVALUATE_XOR (integer)

色空間定数
imagick::COLORSPACE_UNDEFINED (integer)
imagick::COLORSPACE_RGB (integer)
imagick::COLORSPACE_GRAY (integer)
imagick::COLORSPACE_TRANSPARENT (integer)
imagick::COLORSPACE_OHTA (integer)
imagick::COLORSPACE_LAB (integer)
imagick::COLORSPACE_XYZ (integer)
imagick::COLORSPACE_YCBCR (integer)
imagick::COLORSPACE_YCC (integer)
imagick::COLORSPACE_YIQ (integer)
imagick::COLORSPACE_YPBPR (integer)
imagick::COLORSPACE_YUV (integer)
imagick::COLORSPACE_CMYK (integer)
imagick::COLORSPACE_SRGB (integer)
imagick::COLORSPACE_HSB (integer)
imagick::COLORSPACE_HSL (integer)
imagick::COLORSPACE_HWB (integer)
imagick::COLORSPACE_REC601LUMA (integer)
imagick::COLORSPACE_REC709LUMA (integer)
imagick::COLORSPACE_LOG (integer)

仮想ピクセルメソッド定数
imagick::VIRTUALPIXELMETHOD_UNDEFINED (integer)
imagick::VIRTUALPIXELMETHOD_BACKGROUND (integer)
imagick::VIRTUALPIXELMETHOD_CONSTANT (integer)
imagick::VIRTUALPIXELMETHOD_EDGE (integer)
imagick::VIRTUALPIXELMETHOD_MIRROR (integer)
imagick::VIRTUALPIXELMETHOD_TILE (integer)
imagick::VIRTUALPIXELMETHOD_TRANSPARENT (integer)

プレビュー定数
imagick::PREVIEW_UNDEFINED (integer)
imagick::PREVIEW_ROTATE (integer)
imagick::PREVIEW_SHEAR (integer)
imagick::PREVIEW_ROLL (integer)
imagick::PREVIEW_HUE (integer)
imagick::PREVIEW_SATURATION (integer)
imagick::PREVIEW_BRIGHTNESS (integer)
imagick::PREVIEW_GAMMA (integer)
imagick::PREVIEW_SPIFF (integer)
imagick::PREVIEW_DULL (integer)
imagick::PREVIEW_GRAYSCALE (integer)
imagick::PREVIEW_QUANTIZE (integer)
imagick::PREVIEW_DESPECKLE (integer)
imagick::PREVIEW_REDUCENOISE (integer)
imagick::PREVIEW_ADDNOISE (integer)
imagick::PREVIEW_SHARPEN (integer)
imagick::PREVIEW_BLUR (integer)
imagick::PREVIEW_THRESHOLD (integer)
imagick::PREVIEW_EDGEDETECT (integer)
imagick::PREVIEW_SPREAD (integer)
imagick::PREVIEW_SOLARIZE (integer)
imagick::PREVIEW_SHADE (integer)
imagick::PREVIEW_RAISE (integer)
imagick::PREVIEW_SEGMENT (integer)
imagick::PREVIEW_SWIRL (integer)
imagick::PREVIEW_IMPLODE (integer)
imagick::PREVIEW_WAVE (integer)
imagick::PREVIEW_OILPAINT (integer)
imagick::PREVIEW_CHARCOALDRAWING (integer)
imagick::PREVIEW_JPEG (integer)

レンダリング方法定数
imagick::RENDERINGINTENT_UNDEFINED (integer)
imagick::RENDERINGINTENT_SATURATION (integer)
imagick::RENDERINGINTENT_PERCEPTUAL (integer)
imagick::RENDERINGINTENT_ABSOLUTE (integer)
imagick::RENDERINGINTENT_RELATIVE (integer)

インターレース定数 (imagick::INTERLACE_GIF、imagick::INTERLACE_JPEG および imagick::INTERLACE_PNG は Imagemagick 6.3.5 以降を使用して Imagick をコンパイルした場合にのみ使用可能)
imagick::INTERLACE_UNDEFINED (integer)
imagick::INTERLACE_NO (integer)
imagick::INTERLACE_LINE (integer)
imagick::INTERLACE_PLANE (integer)
imagick::INTERLACE_PARTITION (integer)
imagick::INTERLACE_JPEG (integer)
imagick::INTERLACE_GIF (integer)
imagick::INTERLACE_PNG (integer)

塗りつぶし定数
imagick::FILLRULE_UNDEFINED (integer)
imagick::FILLRULE_EVENODD (integer)
imagick::FILLRULE_NONZERO (integer)

パスユニット定数
imagick::PATHUNITS_UNDEFINED (integer)
imagick::PATHUNITS_USERSPACE (integer)
imagick::PATHUNITS_USERSPACEONUSE (integer)
imagick::PATHUNITS_OBJECTBOUNDINGBOX (integer)

ラインキャップ定数
imagick::LINECAP_UNDEFINED (integer)
imagick::LINECAP_BUTT (integer)
imagick::LINECAP_ROUND (integer)

imagick::LINECAP_SQUARE (integer)

ラインジョイン定数
imagick::LINEJOIN_UNDEFINED (integer)
imagick::LINEJOIN_MITER (integer)
imagick::LINEJOIN_ROUND (integer)
imagick::LINEJOIN_BEVEL (integer)

リソース型定数
imagick::RESOURCETYPE_UNDEFINED (integer)
imagick::RESOURCETYPE_AREA (integer)
imagick::RESOURCETYPE_DISK (integer)
imagick::RESOURCETYPE_FILE (integer)
imagick::RESOURCETYPE_MAP (integer)
imagick::RESOURCETYPE_MEMORY (integer)

レイヤーメソッド定数 (ImageMagick 6.3.3 以降でコンパイルした場合に使用可能)
imagick::LAYERMETHOD_UNDEFINED (integer)
imagick::LAYERMETHOD_COALESCE (integer)
imagick::LAYERMETHOD_COMPAREANY (integer)
imagick::LAYERMETHOD_COMPARECLEAR (integer)
imagick::LAYERMETHOD_COMPAREOVERLAY (integer)
imagick::LAYERMETHOD_DISPOSE (integer)
imagick::LAYERMETHOD_OPTIMIZE (integer)
imagick::LAYERMETHOD_OPTIMIZEIMAGE (integer)
imagick::LAYERMETHOD_OPTIMIZEPLUS (integer)
imagick::LAYERMETHOD_OPTIMIZETRANS (integer)
imagick::LAYERMETHOD_REMOVEDUPS (integer)
imagick::LAYERMETHOD_REMOVEZERO (integer)
imagick::LAYERMETHOD_COMPOSITE (integer)

方向定数 (ImageMagick 6.3.4 以降でコンパイルした場合に使用可能)
imagick::ORIENTATION_UNDEFINED (integer)
imagick::ORIENTATION_TOPLEFT (integer)
imagick::ORIENTATION_TOPRIGHT (integer)
imagick::ORIENTATION_BOTTOMRIGHT (integer)
imagick::ORIENTATION_BOTTOMLEFT (integer)
imagick::ORIENTATION_LEFTTOP (integer)
imagick::ORIENTATION_RIGHTTOP (integer)
imagick::ORIENTATION_RIGHTBOTTOM (integer)
imagick::ORIENTATION_LEFTBOTTOM (integer)

歪め定数 (ImageMagick 6.3.6 以降でコンパイルした場合に使用可能)
imagick::DISTORTION_UNDEFINED (integer)
imagick::DISTORTION_AFFINE (integer)
imagick::DISTORTION_AFFINEPROJECTION (integer)
imagick::DISTORTION_ARC (integer)
imagick::DISTORTION_BILINEAR (integer)
imagick::DISTORTION_PERSPECTIVE (integer)
imagick::DISTORTION_PERSPECTIVEPROJECTION (integer)
imagick::DISTORTION_SCALEROTATETRANSLATE (integer)

インストール

(No version information available, might be only in CVS)

インストール — Imagick 拡張モジュールのインストール

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/imagick.

> **注意**: この拡張モジュールの正式な名前は imagick です。

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

その他のプラットフォームでのインストール要件

PHP >= 5.2.1 および ImageMagick >= 6.2.4 が必要です。 Imagick 拡張モジュールが処理できるフォーマットの数は、 インストールされている ImageMagick でサポートしているフォーマットに依存します。 たとえば、Imagemagick で PDF を操作するには ghostscript が必要となります。

Windows でのインストール要件

バージョンに関する条件は上と同じです。ImageMagick のバイナリは http://imagemagick.org/ で取得できるので、Windows でこの拡張モジュールを使用するにはコンパイラは不要です。

Imagick

(No version information available, might be only in CVS)

Imagick — Imagick クラス

```
class Imagick implements Iterator, Traversable
```

画像メソッドおよびグローバルメソッド

Imagick クラスには、複数の画像を同時に保持して操作する機能があります。 内部的には、この機能をスタックで実装しています。 現在の画像を保持する内部ポインタが存在します。 Imagick クラスのすべての画像に対する操作を行う関数もありますが、 大半の関数は内部ポインタが指す現在の画像に対してのみ操作を行います。 スタック内の現在の画像に対してのみ影響を及ぼす関数については、 そのメソッド名に Image という単語を含める決まりになっています。

クラスメソッド

あまりにもたくさんのメソッドがあるので、 一般的な用法に絞った簡単な一覧を示します。

目的別のクラスメソッド

画像に対する効果	Get メソッド	Set メソッド	画像の読み書き	その他
adaptiveBlurImage	getCompression	setBackgroundColor	__construct	clear
adaptiveResizeImage	getFilename	setCompressionQuality	addImage	clone
adaptiveSharpenImage	getFormat	setCompression	appendImages	current
adaptiveTresholdImage	getImageBackgroundColor	setFilename	getFilename	destroy
addNoiseImage	getImageBlob	setFormat	getFormat	getCopyright
affinetransformimage	getImageBluePrimary	setImageBackgroundColor	getImageFilename	getHomeURL
annotateImage	getImageBorderColor	setFirstIterator	getImageFormat	commentImage
averageImages	getImageChannelDepth	setImageBias	getImage	getNumberImages
blackThresholdImage	getImageChannelDistortion	setImageBluePrimary	setImageFilename	getReleaseDate
blurImage	getImageChannelExtrema	setImageBorderColor	setImageFormat	getVersion
borderImage	getImageChannelMean	setImageChannelDepth	readImageFile	hasNextImage
charcoalImage	getImageChannelStatistics	setImageColormapColor	readImage	hasPreviousImage
chopImage	getImageColormapColor	setImageColorSpace	writeImages	labelImage
clipImage	getImageColorspace	setImageCompose	writeImage	newImage
clipPathImage	getImageColors	setImageCompression		newPseudoImage
coalesceImages	getImageCompose	setImageDelay		nextImage
colorFloodFillImage	getImageDelay	setImageDepth		pingImageBlob
colorizeImage	getImageDepth	setImageDispose		pingImageFile
combineImages	getImageDispose	setImageDispose		pingImage
compareImageChannels	getImageDistortion	setImageExtent		previousImage
compareImageLayers	getImageExtrema	setImageFilename		profileImage
compositeImage	getImageFilename	setImageFormat		queryFormats
contrastImage	getImageFormat	setImageGamma		removeImageProfile
constrastStretchImage	getImageGamma	setImageGreenPrimary		removeImage
convolveImage	getImageGeometry	setImageIndex		setFirstIterator
cropImage	getImageGreenPrimary	setImageInterpolateMethod		setImageIndex
cycleColormapImage	getImageHeight	setImageIterations		valid
deconstructImages	getImageHistogram	setImageMatteColor		
drawImage	getImageIndex	setImageMatte		
edgeImage	getImageInterlaceScheme	setImagePage		
embossImage	getImageInterpolateMethod	setImageProfile		
enhanceImage	getImageIterations	setImageProperty		
equalizeImage	getImageMatteColor	setImageRedPrimary		
evaluateImage	getImageMatte	setImageRenderingIntent		
flattenImages	getImagePage	setImageResolution		
flipImage	getImagePixelColor	setImageScene		
flopImage	getImageProfile	setImageTicksPerSecond		
imageImage	getImageProperty	setImageType		
fxImage	getImageRedPrimary	setImageUnits		
gammaImage	getImageRegion	setImageVirtualPixelMethod		
gaussianBlurImage	getImageRenderingIntent	setImageWhitepoint		
implodeImage	getImageResolution	setInterlaceScheme		
levelImage	getImageScene	setOption		
linearStretchImage	getImageSignature	setPage		
magnifyImage	getImageTicksPerSecond	setResolution		
matteFloodFillImage	getImageTotalInkDensity	setResourceLimit		
medianFilterImage	getImageType	setSamplingFactors		
minifyImage	getImageUnits	setSizeOffset		
modulateImage	getImageVirtualPixelMethod	setSize		

Imagick::adaptiveBlurImage

(No version information available, might be only in CVS)

Imagick::adaptiveBlurImage — adaptive blur (順応性にじみ) フィルタを画像に追加する

説明

bool **Imagick::adaptiveBlurImage** (float $radius , float $sigma [, int $channel])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

adaptive blur (順応性にじみ) フィルタを画像に追加します。 adaptive blur の効果は、画像の端に近づくほど弱くなります。 一方、標準の blur の場合は画像全体に均一に働きます。

パラメータ

radius

ガウス分布の半径。中心を含まないピクセル数。 0 を指定すると、半径を自動的に選択します。

sigma

ガウス分布の標準偏差 (ピクセル単位)。

channel

そのチャネルモードで使用可能なチャネル定数を指定します。 複数のチャネルを適用するには、定数をビット演算子で連結します。 デフォルトは Imagick::CHANNEL_ALL です。 チャネル定数 の一覧を参照ください。

返り値

成功した場合に **TRUE** を返します。

エラー / 例外

エラー時に ImagickException をスローします。

例

Example#1 Imagick::adaptiveBlurImage() の使用法

画像に順応性にじみを適用し、ブラウザに表示します。

```
<?php

header('Content-type: image/jpeg');

$image = new Imagick('test.jpg');

$image->adaptiveBlurImage(5,3);
echo $image;

?>
```

参考

- Imagick::blurImage()
- Imagick::motionBlurImage()
- Imagick::radialBlurImage()

Imagick::adaptiveResizeImage

(No version information available, might be only in CVS)

Imagick::adaptiveResizeImage — データに依存する三角測量にもとづいて画像のサイズを変更する

説明

bool **Imagick::adaptiveResizeImage** (int $columns , int $rows)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

データに依存する三角測量にもとづいて画像のサイズを変更します。 色が突然変わる箇所でのにじみを避けます。 たとえば、画像をほんの少しだけ縮小して "ウェブサイズ" にする場合などに便利です。フルサイズの画像のサムネイルを作成する場合などは、 あまりうまくいかないかもしれません。

パラメータ

columns

変更後の画像のカラム数。

rows

変更後の画像の行数。

返り値

成功した場合に ***TRUE*** *を返します。*

エラー / 例外

エラー時に *ImagickException* *をスローします。*

例

Example#1 *Imagick::adaptiveResizeImage()* の使用法

画像のサイズをウェブの標準サイズに変更します。 このメソッドは、元のサイズより少しだけ小さくする場合に最適です。 *size.*

```
<?php
header('Content-type:  image/jpeg');

$image = new Imagick('image.jpg');
$image->adaptiveResizeImage(1024,768);

echo $image;
?>
```

参考

- *Imagick::chopImage()*
- *Imagick::cropImage()*
- *Imagick::magnifyImage()*
- *Imagick::minifyImage()*
- *Imagick::resizeImage()*
- *Imagick::scaleImage()*
- *Imagick::shaveImage()*
- *Imagick::thumbnailImage()*
- *Imagick::trimImage()*

Imagick::adaptiveSharpenImage

(No version information available, might be only in CVS)

Imagick::adaptiveSharpenImage — 順応して画像をシャープにする

説明

bool ***Imagick::adaptiveSharpenImage*** *(float $radius , float $sigma [, int $channel])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な *PHP* *のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。*

画像をシャープにします。 画像の端の方ほど強くシャープ処理を行い、 端から離れるにつれてシャープ処理の程度を弱くします。

パラメータ

radius

ガウス分布の半径。中心を含まないピクセル数。 *0* *を使用すると自動的に選択します。*

sigma

ガウス分布の標準偏差（ピクセル単位）。

channel

そのチャネルモードで使用可能なチャネル定数を指定します。 複数のチャネルを適用するには、定数をビット演算子で連結します。 デフォルトは *Imagick::CHANNEL_ALL* *です。 チャネル定数 の一覧を参照ください。*

返り値

成功した場合に ***TRUE*** *を返します。*

例

Example#1 *Imagick::adaptiveSharpenImage()* の例

半径 2、シグマ 1 で順応性のシャープ処理を施します。

```
<?php
try {
    $image = new Imagick('image.png');
    $image->adaptiveSharpenImage(2,1);
} catch(ImagickException $e) {
    echo 'Error: ' , $e->getMessage();
    die();
}
header('Content-type: image/png');
echo $image;
?>
```

参考

- Imagick::sharpenImage()

Imagick::adaptiveThresholdImage

(No version information available, might be only in CVS)

Imagick::adaptiveThresholdImage — 輝度の範囲にもとづいて各ピクセルの閾値を選択する

説明

bool **Imagick::adaptiveThresholdImage** (int $width , int $height , int $offset)

周辺のピクセルの輝度の範囲にもとづいて、 各ピクセルの閾値を選択します。 画像全体の輝度のヒストグラムが特定の頂点を持っていない場合の閾値の設定が可能となります。

パラメータ

width

周辺の幅。

height

周辺の高さ。

offset

平均オフセット。

返り値

成功した場合に **TRUE** を返します。

Imagick::addImage

(No version information available, might be only in CVS)

Imagick::addImage — 新しい画像を Imagick オブジェクトの画像リストに追加する

説明

bool **Imagick::addImage** (Imagick $source)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

元オブジェクトの現在の位置にある新しい画像を、 Imagick オブジェクトの画像リストに追加します。 この操作を行うと、イテレータがリストの最後尾に移動します。

パラメータ

source

元の Imagick オブジェクト。

返り値

成功した場合に **TRUE** を返します。

Imagick::addNoiseImage

(No version information available, might be only in CVS)

Imagick::addNoiseImage — ランダムなノイズを画像に追加する

説明

bool **Imagick::addNoiseImage** (int $noise_type [, int $channel])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ランダムなノイズを画像に追加します。

パラメータ

noise_type

ノイズの形式。 ノイズ定数 の一覧を参照ください。

channel

そのチャネルモードで使用可能なチャネル定数を指定します。 複数のチャネルを適用するには、定数をビット演算子で連結します。 デフォルトは Imagick::CHANNEL_ALL です。 チャネル定数 の一覧を参照ください。

返り値

成功した場合に **TRUE** を返します。

Imagick::affineTransformImage

(No version information available, might be only in CVS)

Imagick::affineTransformImage — 画像を変換する

説明

bool **Imagick::affineTransformImage** (ImagickDraw $matrix)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

アフィン行列の指定にしたがって画像を変換します。

パラメータ

matrix

アフィン行列。

返り値

成功した場合に **TRUE** を返します。

Imagick::annotateImage

(No version information available, might be only in CVS)

Imagick::annotateImage — 画像にテキストによる注記を加える

説明

bool **Imagick::annotateImage** (ImagickDraw $draw_settings , float $x , float $y , float $angle , string $text)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を画像に加えます。

パラメータ

draw_settings

テキスト描画設定を含む ImagickDraw オブジェクト。

x

テキストの左端の水平オフセットをあらわすピクセル数。

y

テキストのベースラインの垂直オフセットをあらわすピクセル数。

angle

テキストを書き出す角度。

text

描画するテキスト。

返り値

成功した場合に **TRUE** を返します。

例

Example#1 Imagick::annotateImage() の使用法

空の画像にテキスト注記を加えます。

```
<?php
/* オブジェクトを作成します */
$image = new Imagick();
$draw = new ImagickDraw();
$pixel = new ImagickPixel( 'gray' );

/* 画像を作成します */
$image->newImage(800, 75, $pixel);

/* 黒いテキスト */
$pixel->setColor('black');

/* フォントのプロパティ */
$draw->setFont('Bookman-DemiItalic');
$draw->setFontSize( 30 );

/* テキストの作成 */
$image->annotateImage($draw, 10, 45, 0, 'The quick brown fox jumps over the lazy dog');

/* 画像形式の設定 */
$image->setImageFormat('png');

/* ヘッダをつけて画像の出力 */
header('Content-type: image/png');
echo $image;

?>
```

参考

- ImagickDraw::annotation()
- ImagickDraw::setFont()

Imagick::appendImages

(No version information available, might be only in CVS)

Imagick::appendImages — 画像群を追加する

説明

Imagick **Imagick::appendImages** (bool $stack)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像群を追加します。

パラメータ

stack

積み込む方向（上から下、あるいは下から上）。

返り値

成功した場合に Imagick インスタンスを返します。 失敗した場合に ImagickException をスローします。

Imagick::averageImages

(No version information available, might be only in CVS)

Imagick::averageImages — 画像群を平均化する

説明

Imagick **Imagick::averageImages** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像群を平均化します。

返り値

成功した場合に Imagick インスタンスを返します。 失敗した場合に ImagickException をスローします。

Imagick::blackThresholdImage

(No version information available, might be only in CVS)

Imagick::blackThresholdImage — 閾値に満たないすべてのピクセルを黒にする

説明

bool **Imagick::blackThresholdImage** (ImagickPixel $threshold)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick::thresholdImage() と似ていますが、 これは閾値に満たないピクセルをすべて黒にし、その他のピクセルはそのままにします。

パラメータ

threshold

すべて黒にする閾値。

返り値

成功した場合に **TRUE** を返します。

Imagick::blurImage

(No version information available, might be only in CVS)

Imagick::blurImage — blur (にじみ) フィルタを画像に追加する

説明

bool **Imagick::blurImage** (float $radius , float $sigma [, int $channel])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

blur (にじみ) フィルタを画像に追加します。 オプションの 3 番目のパラメータを指定すると、 特定のチャネルにのみ適用します。

パラメータ

radius

にじみの半径。

sigma

標準偏差。

channel

チャネル 定数。 省略した場合は全チャネルが対象となります。

返り値

成功した場合に TRUE を返します。

例

Example#1 Imagick::blurImage() の使用法

画像をにじませ、ブラウザに表示します。

```
<?php

header('Content-type:  image/jpeg');

$image = new Imagick('test.jpg');

$image->blurImage(5,3);
echo $image;

?>
```

参考

- Imagick::adaptiveBlurImage()
- Imagick::motionBlurImage()
- Imagick::radialBlurImage()

Imagick::borderImage

(No version information available, might be only in CVS)

Imagick::borderImage — 画像の周りを枠線で囲む

説明

bool **Imagick::borderImage** (ImagickPixel $bordercolor , int $width , int $height)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の周りを枠線で囲みます。枠線の色は ImagickPixel オブジェクトで指定します。

パラメータ

bordercolor

枠線の色を含む ImagickPixel オブジェクト。

width

枠線の幅。

height

枠線の高さ。

返り値

成功した場合に TRUE を返します。

Imagick::charcoalImage

(No version information available, might be only in CVS)

Imagick::charcoalImage — 木炭画をシミュレートする

説明

bool **Imagick::charcoalImage** (float $radius , float $sigma)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

木炭画をシミュレートします。

パラメータ

radius

ガウス分布の半径。中心を含まないピクセル数。

sigma

ガウス分布の標準偏差（ピクセル単位）。

返り値

成功した場合に **TRUE** を返します。

Imagick::chopImage

(No version information available, might be only in CVS)

Imagick::chopImage — 画像の一部を取り除き、切り詰める

説明

```
bool Imagick::chopImage ( int $width , int $height , int $x , int $y )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の一部を削除し、そこにあった画像をなくします。

パラメータ

width

切り取る範囲の幅。

height

切り取る範囲の高さ。

x

切り取る範囲の X 座標。

y

切り取る範囲の Y 座標。

返り値

成功した場合に **TRUE** を返します。

例

Example#1 Imagick::chopImage() の使用法

Imagick::chopImage の使用例です。

```
<?php
/* オブジェクトの作成 */
$image = new Imagick();
$pixel = new ImagickPixel( 'gray' );

/* 新しい画像 */
$image->newImage(400, 200, $pixel);

/* 画像の切り取り */
$image->chopImage(200, 200, 0, 0);

/* 画図おフォーマットの指定 */
$image->setImageFormat('png');

/* ヘッダをつけて画像の出力 */
header('Content-type: image/png');
echo $image;

?>
```

参考

- Imagick::cropImage()

Imagick::clear

(No version information available, might be only in CVS)

Imagick::clear — *Imagick オブジェクトに関連付けられたすべてのリソースをクリアする*

説明

bool ***Imagick::clear*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトに関連付けられたすべてのリソースをクリアします。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::clipImage

(No version information available, might be only in CVS)

Imagick::clipImage — *8BIM プロファイルの最初のパスにそって切り取る*

説明

bool ***Imagick::clipImage*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

8BIM プロファイルの最初のパスが存在する場合に、それにそって切り取ります。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::clipPathImage

(No version information available, might be only in CVS)

Imagick::clipPathImage — *8BIM プロファイルの指定した名前のパスにそって切り取る*

説明

bool ***Imagick::clipPathImage*** *(string $pathname , bool $inside)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

8BIM プロファイルの指定した名前のパスが存在する場合に、それにそって切り取ります。 その後の操作はパスの内部に対してのみ有効となります。番号で指定する場合は、番号の前に # を付加します。つまり "#1" とすると最初のパスを使用します。

パラメータ

pathname

パスの名前。

inside

true の場合は、その後の操作はクリップパスの内側に対して適用されます。 それ以外の場合は、その後の操作はクリップパスの外側に対して適用されます。

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::clone

(No version information available, might be only in CVS)

Imagick::clone — Imagick オブジェクトの完全なコピーを作成する

説明

Imagick **Imagick::clone** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトの完全なコピーを作成します。

パラメータ

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

Imagick::coalesceImages

(No version information available, might be only in CVS)

Imagick::coalesceImages — 複数の画像を合成する

説明

Imagick **Imagick::coalesceImages** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ページオフセットや処理方法を指定して、複数の画像を合成します。 典型的な使用例としては、たとえば GIF や MIFF、MNG といった画像シーケンスを最初に背景として指定し、 それ以降にさまざまなサイズやオフセットの画像を合成するというものがあります。 返り値は、新規 Imagick オブジェクトとなります。 シーケンスの各画像のサイズは、最初の画像のサイズと同じになります。

パラメータ

返り値

成功した場合に新規 Imagick オブジェクトを返します。 失敗した場合に ImagickException をスローします。

Imagick::colorFloodfillImage

(No version information available, might be only in CVS)

Imagick::colorFloodfillImage — 対象にマッチする任意の点の色の値を変更する

説明

bool **Imagick::colorFloodfillImage** (ImagickPixel $fill , float $fuzz , ImagickPixel $bordercolor , int $x , int $y)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

対象にマッチする任意の点の色の値を変更します。

パラメータ

fill

塗りつぶし色を表す ImagickPixel オブジェクト。

fuzz

fuzz の量。たとえば fuzz を 10 に設定すると、 赤の輝度が 100 および 102 の点は同じ色とみなされます。

bordercolor

枠線の色を表す ImagickPixel オブジェクト。

x

開始位置の X 座標。

y

開始位置の Y 座標。

返り値

*成功した場合に **TRUE** を返します。*

Imagick::colorizeImage

(No version information available, might be only in CVS)

Imagick::colorizeImage — 塗りつぶし色と画像を混合する

説明

*bool **Imagick::colorizeImage** (ImagickPixel $colorize , ImagickPixel $opacity)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の各ピクセルに、塗りつぶし色を混合します。

パラメータ

colorize

色を表す ImagickPixel オブジェクト。

opacity

不透明度を表す ImagickPixel オブジェクト。

返り値

*成功した場合に **TRUE** を返します。*

Imagick::clutImage

(No version information available, might be only in CVS)

Imagick::clutImage — ルックアップテーブルをもとに画像の色を置き換える

説明

*bool **Imagick::clutImage** (Imagick $lookup_table [, int $channel])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の色を、ルックアップテーブルをもとに置き換えます。 オプションの 2 番目のパラメータを指定すると、 特定のチャネルの色を置き換えます。このメソッドは、ImageMagick 6.3.5-7 以降で Imagick をコンパイルした場合に使用可能です。

パラメータ

lookup_table

色ルックアップテーブルを含む Imagick オブジェクト。

channel

チャネルタイプ 定数。 省略した場合は全チャネルが対象となります。

返り値

*成功した場合に **TRUE** を返します。*

例

Example#1 Imagick::clutImage() の使用法

画像の色を、ルックアップテーブルをもとに置き換えます。

```
<?php
$image = new Imagick('test.jpg');
$clut = new Imagick();
$clut->newImage(1, 1, new ImagickPixel('black'));
$image->clutImage( $clut );
$image->writeImage( 'test_out.jpg' );
?>
```

参考

- Imagick::adaptiveBlurImage()
- Imagick::motionBlurImage()
- Imagick::radialBlurImage()

Imagick::combineImages

(No version information available, might be only in CVS)

Imagick::combineImages — ひとつあるいは複数の画像をひとつにまとめる

説明

Imagick **Imagick::combineImages** (int $channelType)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ひとつあるいは複数の画像をひとつにまとめます。 一連の画像の各ピクセルのグレースケール値が、 合成した画像の特定のチャネルの値となります。典型的なパターンでは 画像 1 => Red、画像 2 => Green、画像 3 => Blue などとなります。

パラメータ

channelType

そのチャネルモードで使用可能なチャネル定数のいずれか。 複数のチャネルを適用するには、channeltype 定数をビット演算子でまとめます。 デフォルトは Imagick::CHANNEL_ALL です。 チャネル定数 の一覧を参照ください。

返り値

成功した場合に **TRUE** を返します。

Imagick::commentImage

(No version information available, might be only in CVS)

Imagick::commentImage — コメントを画像に追加する

説明

bool **Imagick::commentImage** (string $comment)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

コメントを画像に追加します。

パラメータ

comment

追加するコメント。

返り値

成功した場合に **TRUE** を返します。

例

Example#1 Imagick::commentImage() の使用法

画像にコメントを追加し、そのコメントを取得します。

```php
<?php

/* 新しい Imagick オブジェクトを作成します */
$im = new imagick();

/* 空の画像を作成します */
$im->newImage(100, 100, new ImagickPixel("red"));

/* コメントをその画像に追加します */
$im->commentImage("Hello World!");

/* コメントを表示します */
echo $im->getImageProperty("comment");

?>
```

Imagick::compareImageChannels

(No version information available, might be only in CVS)

Imagick::compareImageChannels — ひとつあるいは複数の画像の差を返す

説明

Imagick **Imagick::compareImageChannels** (Imagick $image , int $channelType , int $metricType)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ひとつあるいは複数の画像を比較し、差分の画像を返します。

パラメータ

image

比較する画像を含む Imagick オブジェクト。

channelType

そのチャネルモードで使用可能なチャネル定数を指定します。 複数のチャネルを指定するには、チャネル型定数をビット演算子で結合します。 デフォルトは Imagick::CHANNEL_ALL です。 チャネル定数 の一覧を参照ください。

metricType

メトリック型定数 のいずれか。

返り値

成功した場合に **TRUE** を返します。

Imagick::compareImageLayers

(No version information available, might be only in CVS)

Imagick::compareImageLayers — 複数の画像の中で最大の境界範囲を返す

説明

Imagick **Imagick::compareImageLayers** (int $method)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

各画像をシーケンス内の次の画像と比較し、 最大の境界範囲を返します。

パラメータ

method

レイヤメソッド定数 のいずれか。

返り値

成功した場合に **TRUE** を返します。

Imagick::compositeImage

(No version information available, might be only in CVS)

Imagick::compositeImage — ある画像を別の画像に合成する

説明

bool **Imagick::compositeImage** (Imagick $composite_object , int $composite , int $x , int $y [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ある画像を、別の画像の指定した位置に合成します。

パラメータ

composite_object

合成する画像を保持する Imagick オブジェクト。

compose

合成演算子。合成演算定数 を参照ください。

x

合成する位置の列オフセット。

y

合成する位置の行オフセット。

channel

そのチャネルモードで使用可能なチャネル定数を指定します。 複数のチャネルを指定するには、チャネル型定数をビット演算子で結合します。 デフォルトは Imagick::CHANNEL_ALL です。 チャネル定数 の一覧を参照ください。

返り値

成功した場合に **TRUE** を返します。

Imagick::__construct

(No version information available, might be only in CVS)

Imagick::__construct — Imagick のコンストラクタ

説明

Imagick **Imagick::__construct** ([mixed $files])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick のコンストラクタです。

パラメータ

files

読み込みたい画像へのパス、あるいは複数のパスの配列。

返り値

成功した場合に新しい Imagick オブジェクトを返します。 失敗した場合に ImagickException をスローします。

Imagick::contrastImage

(No version information available, might be only in CVS)

Imagick::contrastImage — 画像のコントラストを変更する

説明

bool **Imagick::contrastImage** (bool $sharpen)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の中の暗めの部分と明るめの部分の輝度の差を強調します。 sharpen に 0 以外を指定すると画像のコントラストを上げ、 0 を指定するとコントラストを下げます。

パラメータ

sharpen

返り値

成功した場合に TRUE を返します。

Imagick::contrastStretchImage

(No version information available, might be only in CVS)

Imagick::contrastStretchImage — カラー画像のコントラストを強調する

説明

bool **Imagick::contrastStretchImage** *(float $black_point , float $white_point [, int $channel])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

とりうる色の範囲全体に色を調整することによって、 カラー画像のコントラストを強調します。

パラメータ

black_point

white_point

channel

返り値

成功した場合に TRUE を返します。

Imagick::convolveImage

(No version information available, might be only in CVS)

Imagick::convolveImage — 独自の畳み込み関数を画像に適用する

説明

bool **Imagick::convolveImage** *(array $kernel [, int $channel])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

独自の畳み込み関数を画像に適用します。

パラメータ

kernel

畳み込みの中心。

channel

そのチャネルモードで使用可能なチャネル定数を指定します。 複数のチャネルを適用するには、定数をビット演算子で連結します。 デフォルトは Imagick::CHANNEL_ALL です。 チャネル定数 の一覧を参照ください。

返り値

成功した場合に TRUE を返します。

Imagick::cropImage

(No version information available, might be only in CVS)

Imagick::cropImage — 画像の一部を抽出する

説明

bool **Imagick::cropImage** (int $width , int $height , int $x , int $y)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の一部を抽出します。

パラメータ

width

height

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::current

(No version information available, might be only in CVS)

Imagick::current — 画像のポインタを正しいシーケンスに設定する

説明

Imagick **Imagick::current** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の imagick オブジェクトへの参照を返し、 画像ポインタを正しいシーケンスに設定します。

パラメータ

返り値

成功した場合に自分自身を返します。 失敗した場合に ImagickException をスローします。

Imagick::cycleColormapImage

(No version information available, might be only in CVS)

Imagick::cycleColormapImage — 画像のカラーマップを移動する

説明

bool **Imagick::cycleColormapImage** (int $displace)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

位置番号を指定して、画像のカラーマップを移動します。 カラーマップを何度か回転させると、サイケデリックな効果が得られます。

パラメータ

displace

返り値

成功した場合に **TRUE** を返します。

Imagick::deconstructImages

(No version information available, might be only in CVS)

Imagick::deconstructImages — 画像間の特定のピクセルの差を返す

説明

bool **Imagick::deconstructImages** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

各画像をシーケンス内の次の画像と比較し、 最大の境界範囲を返します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::despeckleImage

(No version information available, might be only in CVS)

Imagick::despeckleImage — 画像内のスペックルノイズを軽減する

説明

bool **Imagick::despeckleImage** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像内のスペックルノイズを軽減します。ただし元画像の輪郭は保持します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::destroy

(No version information available, might be only in CVS)

Imagick::destroy — Imagick オブジェクトを破棄する

説明

bool **Imagick::destroy** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトを破棄します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::drawImage

(No version information available, might be only in CVS)

Imagick::drawImage — 現在の画像上の ImagickDrawing オブジェクトをレンダリングする

説明

bool **Imagick::drawImage** (ImagickDraw $drawing_wand)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の画像上の ImagickDrawing オブジェクトをレンダリングします。

パラメータ

drawing_wand

返り値

成功した場合に **TRUE** を返します。

Imagick::edgeImage

(No version information available, might be only in CVS)

Imagick::edgeImage — 画像の輪郭を強調する

説明

bool **Imagick::edgeImage** (float $radius)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した半径の畳み込みフィルタを使用して、画像の輪郭を強調します。 radius に 0 を設定すると、Edge() は最適な半径を自動的に取得します。

パラメータ

radius

返り値

成功した場合に **TRUE** を返します。

Imagick::embossImage

(No version information available, might be only in CVS)

Imagick::embossImage — グレースケール画像に三次元効果を施して返す

説明

bool **Imagick::embossImage** (float $radius , float $sigma)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

グレースケール画像に三次元効果を施したものを返します。 指定した半径と標準偏差（シグマ）によるガウス演算によって画像を畳み込みます。 意味のある結果を得るには、半径がシグマより大きくなければなりません。 radius に 0 を指定すると、適切な半径を自動的に設定します。

パラメータ

radius

sigma

返り値

成功した場合に **TRUE** を返します。

Imagick::enhanceImage

(No version information available, might be only in CVS)

Imagick::enhanceImage — ノイジーな画像の品質を向上させる

説明

bool **Imagick::enhanceImage** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

デジタルフィルタを適用して、ノイジーな画像の品質を向上させます。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::equalizeImage

(No version information available, might be only in CVS)

Imagick::equalizeImage — 画像ヒストグラムを均等化する

説明

bool **Imagick::equalizeImage** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像ヒストグラムを均等化します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::evaluateImage

(No version information available, might be only in CVS)

Imagick::evaluateImage — 式を画像に適用する

説明

bool **Imagick::evaluateImage** (int $op , float $constant [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

算術演算、関係演算、論理演算などを画像に適用します。 これらの演算子を使用すると、画像の明度を変えたりコントラストを変えたり、 あるいは画像の "ネガ" を作成したりすることができます。

パラメータ

op

constant

channel

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::flattenImages

(No version information available, might be only in CVS)

Imagick::flattenImages — *画像シーケンスをマージする*

説明

bool ***Imagick::flattenImages*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像シーケンスをマージします。 これは、たとえば Photoshop のレイヤを単一の画像にまとめる場合などに便利です。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::flipImage

(No version information available, might be only in CVS)

Imagick::flipImage — *垂直方向に反転した画像を作成する*

説明

bool ***Imagick::flipImage*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

中心の x 軸を基準してピクセルを反転させ、垂直方向のミラー画像を作成します。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::flopImage

(No version information available, might be only in CVS)

Imagick::flopImage — *水平方向に反転した画像を作成する*

説明

bool ***Imagick::flopImage*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

中心の y 軸を基準してピクセルを反転させ、水平方向のミラー画像を作成します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::frameImage

(No version information available, might be only in CVS)

Imagick::frameImage — 三次元の枠線をシミュレートする

説明

bool **Imagick::frameImage** (ImagickPixel $matte_color , int $width , int $height , int $inner_bevel , int $outer_bevel)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の周りに三次元の枠線を追加します。 width と height で、それぞれ縦の枠線と横の枠線の幅を指定します。 inner_bevel および outer_bevel は、 それぞれ枠の内側と外側の影の幅を指定します。

パラメータ

matte_color

width

height

inner_bevel

outer_bevel

返り値

成功した場合に **TRUE** を返します。

Imagick::fxImage

(No version information available, might be only in CVS)

Imagick::fxImage — 式を画像の各ピクセルに適用する

説明

bool **Imagick::fxImage** (string $expression [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

式を画像の各ピクセルに適用します。

パラメータ

expression

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::gammaImage

(No version information available, might be only in CVS)

Imagick::gammaImage — 画像をガンマ補正する

説明

bool **Imagick::gammaImage** (float $gamma [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像をガンマ補正します。同じ画像でも、 異なるデバイス上で見ると見た目が変わることがあります。 これは、画面上に画像の輝度を表現する方法がことなるためです。 赤、緑、青の各チャネルに個別のガンマレベルを指定するか、 あるいはすべてのチャネルに共通のガンマレベルを指定します。値の典型的な範囲は、0.8 から 2.3 の間となります。

パラメータ

gamma

channel

返り値

成功した場合に TRUE を返します。

Imagick::gaussianBlurImage

(No version information available, might be only in CVS)

Imagick::gaussianBlurImage — 画像をぼかす

説明

bool **Imagick::gaussianBlurImage** (float $radius , float $sigma [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像をぼかします。 指定した半径と標準偏差（シグマ）によるガウス演算によって画像を畳み込みます。 意味のある結果を得るには、半径がシグマより大きくなければなりません。 radius に 0 を指定すると、適切な半径を自動的に設定します。

パラメータ

radius

sigma

channel

返り値

成功した場合に TRUE を返します。

Imagick::getCompressionQuality

(No version information available, might be only in CVS)

Imagick::getCompressionQuality — オブジェクトの圧縮品質を取得する

説明

int **Imagick::getCompressionQuality** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの圧縮品質を取得します。

パラメータ

返り値

成功した場合に TRUE を返します。

Imagick::getCompression

(No version information available, might be only in CVS)

Imagick::getCompression — オブジェクトの圧縮形式を取得する

説明

int ***Imagick::getCompression*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの圧縮形式を取得します。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::getCopyright

(No version information available, might be only in CVS)

Imagick::getCopyright — ImageMagick API の著作権情報を文字列定数で返す

説明

string ***Imagick::getCopyright*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImageMagick API の著作権情報を文字列定数で返します。

パラメータ

返り値

Imagemagick および Magickwand C API の著作権情報を含む文字列を返します。

Imagick::getFilename

(No version information available, might be only in CVS)

Imagick::getFilename — 画像シーケンスに関連付けられたファイル名を取得する

説明

string ***Imagick::getFilename*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像シーケンスに関連付けられたファイル名を取得します。

パラメータ

返り値

成功した場合に文字列を返します。 失敗した場合に ImagickException をスローします。

Imagick::getFormat

(No version information available, might be only in CVS)

Imagick::getFormat — Imagick オブジェクトのフォーマットを取得する

説明

string ***Imagick::getFormat*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトのフォーマットを返します。

パラメータ

返り値

画像フォーマットを返します。 失敗した場合に ImagickException をスローします。

Imagick::getHomeURL

(No version information available, might be only in CVS)

Imagick::getHomeURL — ImageMagick のホーム URL を返す

説明

string ***Imagick::getHomeURL*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImageMagick のホーム URL を返します。

パラメータ

返り値

imagemagick のホームページへのリンクを返します。

Imagick::getImageBackgroundColor

(No version information available, might be only in CVS)

Imagick::getImageBackgroundColor — 画像の背景色を返す

説明

ImagickPixel ***Imagick::getImageBackgroundColor*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の背景色を返します。

パラメータ

返り値

画像の背景色を設定した ImagickPixel を返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageBlob

(No version information available, might be only in CVS)

Imagick::getImageBlob — 画像シーケンスを blob で返す

説明

string ***Imagick::getImageBlob*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像フォーマットを直接記憶します。 これは、画像シーケンスを文字列で返します。 画像のフォーマットによって、返される blob の形式 (GIF, JPEG, PNG, など) が決まります。 別の画像フォーマットを返すには Imagick::SetImageFormat() を使用します。

パラメータ

返り値

画像を含む文字列を返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageBluePrimary

(No version information available, might be only in CVS)

Imagick::getImageBluePrimary — 青が一番強い点を返す

説明

ImagickPixel **Imagick::getImageBluePrimary** (float $x , float $y)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

青が一番強い点を返します。

パラメータ

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageBorderColor

(No version information available, might be only in CVS)

Imagick::getImageBorderColor — 画像の前景色を返す

説明

ImagickPixel **Imagick::getImageBorderColor** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の前景色を返します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageChannelDepth

(No version information available, might be only in CVS)

Imagick::getImageChannelDepth — 特定の画像チャネルの深度を返す

説明

int **Imagick::getImageChannelDepth** (int $channelType)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

特定の画像チャネルの深度を返します。

パラメータ

expression

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageChannelDistortion

(No version information available, might be only in CVS)

Imagick::getImageChannelDistortion — 画像のチャネルを再構築した画像と比較する

説明

float **Imagick::getImageChannelDistortion** (*Imagick $reference* , *int $channel* , *int $metric*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ひとつあるいは複数のチャネルの内容を再構築した画像と比較し、 指定した歪み係数を返します。

パラメータ

reference

channel

metric

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageChannelExtrema

(No version information available, might be only in CVS)

Imagick::getImageChannelExtrema — ひとつあるいは複数の画像チャネルの極値を取得する

説明

array **Imagick::getImageChannelExtrema** (*int $channel*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ひとつあるいは複数の画像チャネルの極値を取得します。 返り値は連想配列で、*"minima"* および *"maxima"* というキーを含みます。

パラメータ

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageChannelMean

(No version information available, might be only in CVS)

Imagick::getImageChannelMean — 平均値と標準偏差を取得する

説明

array **Imagick::getImageChannelMean** (*int $channel*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ひとつあるいは複数の画像チャネルの平均値および標準偏差を取得します。 返り値は連想配列で、"mean" および "standardDeviation" というキーを含みます。

パラメータ

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageChannelStatistics

(No version information available, might be only in CVS)

Imagick::getImageChannelStatistics — 画像の各チャネルの統計情報を返す

説明

array **Imagick::getImageChannelStatistics** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の各チャネルの統計情報を返します。 この情報に含まれるのは、チャネルの深度、最小値と最大値、 平均値、そして標準偏差です。 たとえば赤チャネルの平均値を取得するにはこのようにします。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageColormapColor

(No version information available, might be only in CVS)

Imagick::getImageColormapColor — 指定したインデックスに対応する色マップ上の色を返す

説明

ImagickPixel **Imagick::getImageColormapColor** (int $index)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したインデックスに対応する色マップ上の色を返します。

パラメータ

index

返り値

成功した場合に **TRUE** を返します。

Imagick::getImageColorspace

(No version information available, might be only in CVS)

Imagick::getImageColorspace — 画像の色空間を取得する

説明

int **Imagick::getImageColorspace** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の色空間を取得します。

パラメータ

返り値

*成功した場合に **TRUE** を返します。*

Imagick::getImageColors

(No version information available, might be only in CVS)

Imagick::getImageColors — 画像で使われている色の数を取得する

説明

*int **Imagick::getImageColors** (void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像で使われている色の数を取得します。

パラメータ

返り値

*成功した場合に **TRUE** を返します。*

Imagick::getImageCompose

(No version information available, might be only in CVS)

Imagick::getImageCompose — 画像の合成演算子を返す

説明

*int **Imagick::getImageCompose** (void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像に関連付けられている合成演算子を返します。

パラメータ

返り値

*成功した場合に **TRUE** を返します。*

Imagick::getImageDelay

(No version information available, might be only in CVS)

Imagick::getImageDelay — 画像の遅延を取得する

説明

*int **Imagick::getImageDelay** (void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の遅延を取得します。

パラメータ

返り値

画像の遅延を返します。失敗した場合は ImagickException をスローします。

Imagick::getImageDepth

(PECL imagick:0.9.1-0.9.9)

Imagick::getImageDepth — 画像の深度を取得する

説明

```
int Imagick::getImageDepth ( void )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の深度を取得します。

パラメータ

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

Imagick::getImageDispose

(No version information available, might be only in CVS)

Imagick::getImageDispose — 画像の配置方法を取得する

説明

```
int Imagick::getImageDispose ( void )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の配置方法を取得します。

パラメータ

返り値

成功した場合に配置方法を返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageDistortion

(No version information available, might be only in CVS)

Imagick::getImageDistortion — ある画像と再構築した画像を比較する

説明

```
float Imagick::getImageDistortion ( MagickWand $reference , int $metric )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ある画像と再構築した画像を比較し、 指定した歪み係数を返します。

パラメータ

reference

metric

返り値

その画像で使用する歪み係数（あるいはその推測値）を返します。 エラーが発生した場合は例外をスローします。

Imagick::getImageExtrema

(No version information available, might be only in CVS)

Imagick::getImageExtrema — 画像の極値を取得する

説明

array ***Imagick::getImageExtrema*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の極値を取得します。 "min" および "max" というキーを持つ連想配列を返します。

パラメータ

返り値

"min" および "max" というキーを持つ連想配列を返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageFilename

(No version information available, might be only in CVS)

Imagick::getImageFilename — シーケンス内の特定の画像のファイル名を返す

説明

string ***Imagick::getImageFilename*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

シーケンス内の特定の画像のファイル名を返します。

パラメータ

返り値

画像のファイル名を文字列で返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageSize

(No version information available, might be only in CVS)

Imagick::getImageSize — 画像の長さをバイト数で返す

説明

int ***Imagick::getImageSize*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の長さをバイト数で返します。

パラメータ

返り値

現在の画像のサイズをバイト数で返します。

Imagick::getImageLength

(No version information available, might be only in CVS)

Imagick::getImageLength — 画像の長さをバイト数で取得する

説明

int **Imagick::getImageLength** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の長さをバイト数で返します。

パラメータ

返り値

現在の画像のサイズを整数値で返します。

Imagick::getImageFormat

(No version information available, might be only in CVS)

Imagick::getImageFormat — シーケンス内の特定の画像のフォーマットを返す

説明

string **Imagick::getImageFormat** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

シーケンス内の特定の画像のフォーマットを返します。

パラメータ

返り値

成功した場合は画像のフォーマットを文字列で返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageGamma

(No version information available, might be only in CVS)

Imagick::getImageGamma — 画像のガンマを取得する

説明

float **Imagick::getImageGamma** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のガンマを取得します。

パラメータ

返り値

成功した場合にガンマを返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageGeometry

(No version information available, might be only in CVS)

Imagick::getImageGeometry — 幅と高さを連想配列で取得する

説明

array **Imagick::getImageGeometry** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

幅と高さを連想配列で返します。

パラメータ

返り値

画像の幅と高さを表す配列を返します。 エラー時に ImagickException をスローします。

Imagick::getImageGreenPrimary

(No version information available, might be only in CVS)

Imagick::getImageGreenPrimary — 緑が一番強い点を返す

説明

array **Imagick::getImageGreenPrimary** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

緑が一番強い点を返します。"x" および "y" というキーを持つ配列を返します。

パラメータ

返り値

成功した場合に "x" および "y" というキーを持つ配列を返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageHeight

(No version information available, might be only in CVS)

Imagick::getImageHeight — 画像の高さを返す

説明

int **Imagick::getImageHeight** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の高さを返します。

パラメータ

返り値

画像の高さをピクセル数で返します。 エラー時に ImagickException をスローします。

Imagick::getImageHistogram

(No version information available, might be only in CVS)

Imagick::getImageHistogram — 画像のヒストグラムを取得する

説明

array **Imagick::getImageHistogram** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のヒストグラムを ImagickPixel オブジェクトの配列で返します。

パラメータ

返り値

画像のヒストグラムを ImagickPixel オブジェクトの配列で返します。 エラー時に ImagickException をスローします。

Imagick::getImageIndex

(No version information available, might be only in CVS)

Imagick::getImageIndex — 現在アクティブな画像のインデックスを取得する

説明

int **Imagick::getImageIndex** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクト内で現在アクティブな画像のインデックスを取得します。 このメソッドは非推奨です。 Imagick::getIteratorIndex を参照ください。

パラメータ

返り値

スタック内の画像のインデックスを表す整数値を返します。 エラー時に ImagickException をスローします。

Imagick::getIteratorIndex

(No version information available, might be only in CVS)

Imagick::getIteratorIndex — 現在アクティブな画像のインデックスを取得する

説明

int **Imagick::getIteratorIndex** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクト内で現在アクティブな画像のインデックスを取得します。

パラメータ

返り値

スタック内の画像のインデックスを含む整数値を返します。 エラー時に ImagickException をスローします。

Imagick::getImageInterlaceScheme

(No version information available, might be only in CVS)

Imagick::getImageInterlaceScheme — 画像のインターレース手法を取得する

説明

int **Imagick::getImageInterlaceScheme** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のインターレース手法を取得します。

パラメータ

返り値

成功した場合に画像のインターレース手法を返します。 失敗した場合に ImagickException をスローします。

Imagick::getImageInterpolateMethod

(No version information available, might be only in CVS)

Imagick::getImageInterpolateMethod — 画像の補間方式を返す

説明

int **Imagick::getImageInterpolateMethod** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した画像の補間方式を返します。

パラメータ

返り値

成功した場合に補間方式を返します。 エラー時に ImagickException をスローします。

Imagick::getImageIterations

(No version information available, might be only in CVS)

Imagick::getImageIterations — 画像の反復を取得する

説明

int **Imagick::getImageIterations** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の反復を取得します。

パラメータ

返り値

画像の反復を表す整数値を返します。 失敗した場合に ImagickException をスローします。

Imagick::getImageMatteColor

(No version information available, might be only in CVS)

Imagick::getImageMatteColor — 画像のマット色を返す

説明

ImagickPixel ***Imagick::getImageMatteColor*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のマット色を返します。

パラメータ

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

Imagick::getImageMatte

(No version information available, might be only in CVS)

Imagick::getImageMatte — *画像がマットチャネルを持っているかどうかを返す*

説明

int ***Imagick::getImageMatte*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像がマットチャネルを持っている場合に true、 そうでない場合に false を返します。

パラメータ

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

Imagick::getImagePage

(No version information available, might be only in CVS)

Imagick::getImagePage — *ページのジオメトリを返す*

説明

array ***Imagick::getImagePage*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のページジオメトリを配列で返します。配列のキーは "width"、"height"、"x" および "y" です。

パラメータ

返り値

画像のページジオメトリを配列で返します。配列のキーは "width"、"height"、"x" および "y" です。

Imagick::getImagePixelColor

(No version information available, might be only in CVS)

Imagick::getImagePixelColor — *指定したピクセルの色を返す*

説明

ImagickPixel ***Imagick::getImagePixelColor*** *(int $x , int $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したピクセルの色を返します。

パラメータ

x

y

返り値

指定した座標の色を表す ImagickPixel のインスタンスを返します。 エラー時に ImagickException をスローします。

Imagick::getImageProfile

(No version information available, might be only in CVS)

Imagick::getImageProfile — 指定した名前の画像プロファイルを返す

説明

string ***Imagick::getImageProfile*** *(string $name)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した名前の画像プロファイルを返します。

パラメータ

name

返り値

画像プロファイルを含む文字列を返します。 エラー時に ImagickException をスローします。

Imagick::getImageProfiles

(No version information available, might be only in CVS)

Imagick::getImageProfiles — 画像プロファイルを返す

説明

array ***Imagick::getImageProfiles*** *([string $pattern [, bool $only_names]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

関連付けられているプロファイルのうち、パターンにマッチするものをすべて返します。 二番目の引数を true にすると、プロファイルの名前だけを返します。 このメソッドは、ImageMagick 6.3.5-9 以降を使用して Imagick をコンパイルした場合にのみ存在します。

パラメータ

pattern

プロファイル名のパターン。デフォルトは ""。*

only_names

プロファイル名のみを返すかどうか。

返り値

画像のプロファイルあるいはプロファイル名を含む連想配列を返します。

Imagick::getImageProperty

(No version information available, might be only in CVS)

Imagick::getImageProperty — 指定した名前の画像のプロパティを返す

説明

string **Imagick::getImageProperty** (string $name)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した名前の画像プロパティを返します。

パラメータ

name

プロパティの名前 (たとえば Exif:DateTime)。

返り値

画像のプロパティを含む文字列を返します。 指定した名前のプロパティが存在しない場合に false を返します。

Imagick::getImageProperties

(No version information available, might be only in CVS)

Imagick::getImageProperties — 画像のプロパティを返す

説明

array **Imagick::getImageProperties** ([string $pattern [, bool $only_names]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

関連付けられているプロパティのうち、パターンにマッチするものをすべて返します。 二番目の引数を true にすると、プロパティの名前だけを返します。 このメソッドは、ImageMagick 6.3.5-9 以降を使用して Imagick をコンパイルした場合にのみ存在します。

パラメータ

pattern

プロパティ名のパターン。デフォルトは "*"。

only_names

プロパティ名のみを返すかどうか。

返り値

画像のプロパティあるいはプロパティ名を含む連想配列を返します。

例

Example#1 Imagick::getImageProperties() の使用法

EXIF 情報を抽出します。

```
<?php

/* オブジェクトを作成します */
$im = new imagick("/path/to/example.jpg");

/* EXIF 情報を取得します */
$exifArray = $im->getImageProperties("exif:*");

/* exif のプロパティをループします */
foreach ($exifArray as $name => $property)
{
    echo "{$name} => {$property}<br />¥n";
}

?>
```

Imagick::getImageRedPrimary

(No version information available, might be only in CVS)

Imagick::getImageRedPrimary — 赤が一番強い点を返す

説明

array **Imagick::getImageRedPrimary** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

赤が一番強い点を返します。"x" および "y" というキーを持つ配列を返します。

パラメータ

返り値

赤が一番強い点を "x" および "y" というキーを持つ配列で返します。 失敗した場合は ImagickException をスローします。

Imagick::getImageRegion

(No version information available, might be only in CVS)

Imagick::getImageRegion — 画像の一部を抽出する

説明

bool **Imagick::getImageRegion** (int $width , int $height , int $x , int $y)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の一部を抽出し、新しい wand として返します。

パラメータ

width

height

x

y

返り値

画像の一部を抽出し、新しい wand として返します。 エラー時に ImagickException をスローします。

Imagick::getImageRenderingIntent

(No version information available, might be only in CVS)

Imagick::getImageRenderingIntent — 画像のレンダリング方向を取得する

説明

int **Imagick::getImageRenderingIntent** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のレンダリング方向を取得します。

パラメータ

返り値

画像の レンダリング方向 を返します。 エラー時に ImagickException をスローします。

Imagick::getImageResolution

(No version information available, might be only in CVS)

Imagick::getImageResolution — 画像の X 方向と Y 方向の解像度を取得する

説明

array **Imagick::getImageResolution** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

X 方向および Y 方向の解像度を取得します。

パラメータ

返り値

解像度を配列で返します。 エラー時に ImagickException をスローします。

Imagick::getImageScene

(No version information available, might be only in CVS)

Imagick::getImageScene — 画像のシーンを取得する

説明

int **Imagick::getImageScene** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のシーンを取得します。

パラメータ

返り値

画像のシーンを返します。 エラー時に ImagickException をスローします。

Imagick::getImageSignature

(No version information available, might be only in CVS)

Imagick::getImageSignature — SHA-256 メッセージダイジェストを生成する

説明

string **Imagick::getImageSignature** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像ピクセルストリームの SHA-256 メッセージダイジェストを生成します。

パラメータ

返り値

そのファイルの SHA-256 ハッシュを文字列で返します。 エラー時に ImagickException をスローします。

Imagick::getImageTicksPerSecond

(No version information available, might be only in CVS)

Imagick::getImageTicksPerSecond — *画像の ticks-per-second を取得する*

説明

int ***Imagick::getImageTicksPerSecond*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の ticks-per-second を取得します。

パラメータ

返り値

画像の ticks-per-second を返します。 エラー時に ImagickException をスローします。

Imagick::getImageTotalInkDensity

(No version information available, might be only in CVS)

Imagick::getImageTotalInkDensity — *画像の総インク密度を取得する*

説明

float ***Imagick::getImageTotalInkDensity*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の総インク密度を取得します。

パラメータ

返り値

画像の総インク密度を返します。 エラー時に ImagickException をスローします。

Imagick::getImageType

(PECL imagick:0.9.10-0.9.9)

Imagick::getImageType — *画像の型を取得する*

説明

int ***Imagick::getImageType*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の型を取得します。

パラメータ

返り値

画像の型を返します。 エラー時に ImagickException をスローします。

Imagick::getImageUnits

(No version information available, might be only in CVS)

Imagick::getImageUnits — *画像の解像度の単位を取得する*

説明

int ***Imagick::getImageUnits*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の解像度の単位を取得します。

パラメータ

返り値

画像の解像度の単位を返します。 失敗した場合に ImagickException をスローします。

Imagick::getImageVirtualPixelMethod

(No version information available, might be only in CVS)

Imagick::getImageVirtualPixelMethod — 仮想ピクセルメソッドを取得する

説明

int ***Imagick::getImageVirtualPixelMethod*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した画像の仮想ピクセルメソッドを返します。

パラメータ

返り値

成功した場合に仮想ピクセルメソッドを返します。 失敗した場合に ImagickException をスローします。

Imagick::getImageWhitePoint

(No version information available, might be only in CVS)

Imagick::getImageWhitePoint — 色度が白い点を返す

説明

array ***Imagick::getImageWhitePoint*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

色度が白い点を返します。 "x" および "y" というキーを持つ連装配列となります。

パラメータ

返り値

色度が白い点を返します。 "x" および "y" というキーを持つ連装配列となります。 エラー時に ImagickException をスローします。

Imagick::getImageWidth

(No version information available, might be only in CVS)

Imagick::getImageWidth — 画像の幅を返す

説明

int ***Imagick::getImageWidth*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の幅を返します。

パラメータ

返り値

画像の幅を返します。 エラー時に ImagickException をスローします。

Imagick::getImage

(No version information available, might be only in CVS)

Imagick::getImage — 新しい Imagick オブジェクトを返す

説明

```
Imagick Imagick::getImage ( void )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の画像シーケンスを使用した新しい Imagick オブジェクトを返します。

パラメータ

返り値

現在の画像シーケンスを使用した新しい Imagick オブジェクトを返します。 エラー時には ImagickException をスローします。

Imagick::getInterlaceScheme

(No version information available, might be only in CVS)

Imagick::getInterlaceScheme — オブジェクトのインターレース方式を取得する

説明

```
int Imagick::getInterlaceScheme ( void )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトのインターレース方式を取得します。

パラメータ

返り値

インターレース方式 を返します。エラー時に ImagickException をスローします。

Imagick::getImageOrientation

(No version information available, might be only in CVS)

Imagick::getImageOrientation — 画像の方向を取得する

説明

```
int Imagick::getImageOrientation ( void )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の方向を取得します。返り値は 方向定数 のいずれかとなります。

パラメータ

返り値

成功した場合に整数値を返します。 失敗した場合に ImagickException をスローします。

Imagick::getNumberImages

(No version information available, might be only in CVS)

Imagick::getNumberImages — オブジェクト内の画像の数を返す

説明

int **Imagick::getNumberImages** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトに関連付けられている画像の数を返します。

パラメータ

返り値

Imagick オブジェクトに関連付けられている画像の数を返します。 失敗した場合に ImagickException をスローします。

Imagick::getOption

(No version information available, might be only in CVS)

Imagick::getOption — 指定したキーに対応する値を返す

説明

string **Imagick::getOption** (string $key)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したキーに対応する値を返します。

パラメータ

key

返り値

指定したキーに対応する値を返します。 エラー時に ImagickException をスローします。

Imagick::getPackageName

(No version information available, might be only in CVS)

Imagick::getPackageName — ImageMagick パッケージ名を返す

説明

string **Imagick::getPackageName** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImageMagick パッケージ名を返します。

パラメータ

返り値

ImageMagick パッケージ名を文字列で返します。 エラー時に ImagickException をスローします。

Imagick::getPage

(No version information available, might be only in CVS)

Imagick::getPage — ページのジオメトリを返す

説明

array **Imagick::getPage** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトに関連付けられたページのジオメトリを返します。 "width"、"height"、"x" そして "y" というキーを持つ連装配列となります。

パラメータ

返り値

Imagick オブジェクトに関連付けられたページのジオメトリを返します。 "width"、"height"、"x" そして "y" というキーを持つ連装配列となります。 エラー時に ImagickException をスローします。

Imagick::getPixelIterator

(No version information available, might be only in CVS)

Imagick::getPixelIterator — MagickPixelIterator を返す

説明

ImagickPixelIterator **Imagick::getPixelIterator** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

MagickPixelIterator を返します。

パラメータ

返り値

成功した場合に ImagickPixelIterator を返します。 失敗した場合に ImagickException をスローします。

Imagick::getPixelRegionIterator

(No version information available, might be only in CVS)

Imagick::getPixelRegionIterator — 画像セクションの ImagickPixelIterator を取得する

説明

ImagickPixelIterator **Imagick::getPixelRegionIterator** (int $x , int $y , int $columns , int $rows)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像セクションの ImagickPixelIterator を取得します。

パラメータ

x

y

columns

rows

返り値

画像セクションの ImagickPixelIterator を返します。 エラー時に ImagickException をスローします。

Imagick::getQuantumDepth

(No version information available, might be only in CVS)

Imagick::getQuantumDepth — quantum depth を取得する

説明

array ***Imagick::getQuantumDepth*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick quantum depth を文字列で返します。

パラメータ

返り値

Imagick quantum depth を文字列で返します。 エラー時に ImagickException をスローします。

Imagick::getQuantumRange

(No version information available, might be only in CVS)

Imagick::getQuantumRange — Imagick quantum range を返す

説明

array ***Imagick::getQuantumRange*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick quantum range を文字列で返します。

パラメータ

返り値

Imagick quantum range を文字列で返します。 エラー時に ImagickException をスローします。

Imagick::getReleaseDate

(No version information available, might be only in CVS)

Imagick::getReleaseDate — ImageMagick のリリース日を返す

説明

string ***Imagick::getReleaseDate*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImageMagick のリリース日を文字列で返します。

パラメータ

返り値

ImageMagick のリリース日を文字列で返します。 エラー時に ImagickException をスローします。

Imagick::getResourceLimit

(No version information available, might be only in CVS)

Imagick::getResourceLimit — 指定したリソースの制限を返す

説明

int **Imagick::getResourceLimit** (int $type)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したリソースの制限をメガバイト単位で返します。

パラメータ

type

返り値

指定したリソースの制限をメガバイト単位で返します。 エラー時に ImagickException をスローします。

Imagick::getResource

(No version information available, might be only in CVS)

Imagick::getResource — 指定したリソースのメモリ使用状況を返す

説明

int **Imagick::getResource** (int $type)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したリソースのメモリ使用状況をメガバイト単位で返します。

パラメータ

type

返り値

指定したリソースのメモリ使用状況をメガバイト単位で返します。 エラー時に ImagickException をスローします。

Imagick::getSamplingFactors

(No version information available, might be only in CVS)

Imagick::getSamplingFactors — 水平方向および垂直方向のサンプリング係数を取得する

説明

array **Imagick::getSamplingFactors** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

水平方向および垂直方向のサンプリング係数を取得します。

パラメータ

返り値

画像の水平方向および垂直方向のサンプリング係数を含む連装配列を返します。 エラー時に ImagickException をスローします。

Imagick::getSizeOffset

(No version information available, might be only in CVS)

Imagick::getSizeOffset — サイズのオフセットを返す

説明

int **Imagick::getSizeOffset** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトのサイズのオフセットを返します。

パラメータ

返り値

Imagick オブジェクトのサイズのオフセットを返します。 エラー時に ImagickException をスローします。

Imagick::getSize

(No version information available, might be only in CVS)

Imagick::getSize — Imagick オブジェクトのサイズを取得する

説明

array **Imagick::getSize** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトのサイズを返します。 "columns" および "rows" というキーを持つ連装配列となります。

パラメータ

返り値

Imagick オブジェクトのサイズを返します。 "columns" および "rows" というキーを持つ連装配列となります。

Imagick::getVersion

(No version information available, might be only in CVS)

Imagick::getVersion — ImageMagick API のバージョンを返す

説明

array **Imagick::getVersion** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImageMagick API のバージョンを文字列と数値で返します。

パラメータ

返り値

ImageMagick API のバージョンを文字列と数値で返します。 エラー時に ImagickException をスローします。

Imagick::hasNextImage

(No version information available, might be only in CVS)

Imagick::hasNextImage — オブジェクトが次の画像を保持しているかどうかを調べる

説明

bool **Imagick::hasNextImage** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

リスト内でひとつ先に進んだときにまだオブジェクトが画像を保持していれば true を返します。

パラメータ

返り値

リスト内でひとつ先に進んだときにまだオブジェクトが画像を保持していれば true を返します。 エラー時に ImagickException をスローします。

Imagick::hasPreviousImage

(No version information available, might be only in CVS)

Imagick::hasPreviousImage — オブジェクトが前の画像を保持しているかどうかを調べる

説明

bool **Imagick::hasPreviousImage** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

リスト内で逆方向にひとつ進んだときにまだオブジェクトが画像を保持していれば true を返します。

パラメータ

返り値

リスト内で逆方向にひとつ進んだときにまだオブジェクトが画像を保持していれば true を返します。 失敗した場合に ImagickException をスローします。

Imagick::identifyImage

(No version information available, might be only in CVS)

Imagick::identifyImage — 画像を識別し、属性を取得する

説明

string **Imagick::identifyImage** ([bool $appendRawOutput])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像を識別し、属性を返します。属性には画像の幅や高さ、サイズなどが含まれます。

パラメータ

appendRawOutput

返り値

画像を識別し、属性を返します。属性には画像の幅や高さ、サイズなどが含まれます。 エラー時に ImagickException をスローします。

Imagick::implodeImage

(No version information available, might be only in CVS)

Imagick::implodeImage — 新しい画像をコピーとして作成する

説明

bool **Imagick::implodeImage** (float $radius)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

既存の画像をコピーして新しい画像を作成します。 既存の画像のピクセルを、指定したパーセンテージで "内破" します。

パラメータ

radius

返り値

成功した場合に **TRUE** を返します。

Imagick::labelImage

(No version information available, might be only in CVS)

Imagick::labelImage — ラベルを画像に追加する

説明

bool **Imagick::labelImage** (string $label)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ラベルを画像に追加します。

パラメータ

label

返り値

成功した場合に **TRUE** を返します。

Imagick::levelImage

(No version information available, might be only in CVS)

Imagick::levelImage — 画像のレベルを調節する

説明

bool **Imagick::levelImage** (float $blackPoint , float $gamma , float $whitePoint [, int $channel])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のレベルを調整します。 指定した白と黒の範囲にある色を、利用可能な色の範囲全体に拡大します。 黒の点、中間点、白の点をパラメータで指定します。 黒の点は、画像の中で一番暗い色を表します。 これより暗い色の点はゼロに設定されます。 中間点は、画像に適用するガンマ補正を指定します。 黒の点は、画像の中で一番明るい色を表します。 これより明るい色の点は最大値に設定されます。

パラメータ

blackPoint

gamma

whitePoint

channel

返り値

*成功した場合に **TRUE** を返します。*

Imagick::linearStretchImage

(No version information available, might be only in CVS)

Imagick::linearStretchImage — 画像の輝度を引き伸ばして飽和させる

説明

*bool **Imagick::linearStretchImage** (float $blackPoint , float $whitePoint)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の輝度を引き伸ばして飽和させます。

パラメータ

blackPoint

whitePoint

返り値

*成功した場合に **TRUE** を返します。*

Imagick::magnifyImage

(No version information available, might be only in CVS)

Imagick::magnifyImage — 画像を 2 倍に比例拡大する

説明

*bool **Imagick::magnifyImage** (void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像を元のサイズの 2 倍に比例拡大します。

パラメータ

返り値

*成功した場合に **TRUE** を返します。*

Imagick::matteFloodfillImage

(No version information available, might be only in CVS)

Imagick::matteFloodfillImage — 色の透明度を変更する

説明

*bool **Imagick::matteFloodfillImage** (float $alpha , float $fuzz , ImagickPixel $bordercolor , int $x , int $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

対象となるピクセルとその近傍の透明度を変更します。 FillToBorderMethod を指定した場合は、 画像の描画色に該当しない近傍ピクセルについて

も透明度を変更します。

パラメータ

alpha

fuzz

bordercolor

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::medianFilterImage

(No version information available, might be only in CVS)

Imagick::medianFilterImage — デジタルフィルタを適用する

説明

bool **Imagick::medianFilterImage** (float $radius)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

デジタルフィルタを適用してノイジーな画像の品質を向上させます。 各ピクセルの内容を、radius で定義した近傍ピクセルの中央値で置き換えます。

パラメータ

radius

返り値

成功した場合に **TRUE** を返します。

Imagick::minifyImage

(No version information available, might be only in CVS)

Imagick::minifyImage — 画像をその半分のサイズに比例縮小する

説明

bool **Imagick::minifyImage** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のサイズを元の半分に比例縮小します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::modulateImage

(No version information available, might be only in CVS)

Imagick::modulateImage — 明度、飽和度、色相を制御する

説明

bool **Imagick::modulateImage** (float $brightness , float $saturation , float $hue)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の明度、飽和度、色相を制御します。 色相は、現在位置からの絶対回転角のパーセンテージで表します。 たとえば 50 の場合は反時計回りに 90 度の回転を表します。 また 150 の場合は時計回りの 90 度の回転、0 および 200 はともに 180 度の回転を表します。

パラメータ

brightness

saturation

hue

返り値

*成功した場合に **TRUE** を返します。*

Imagick::montageImage

(No version information available, might be only in CVS)

Imagick::montageImage — 合成画像を作成する

説明

*Imagick **Imagick::montageImage** (ImagickDraw $drawing_wand , string $tile_geometry , string $thumbnail_geometry , int $mode , string $frame)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

複数の画像を合成して画像を作成します。 合成画像上で、画像の名前でタイル状に並べられます。 またオプションで各タイルの直下に画像を置くこともできます。

パラメータ

drawing_wand

tile_geometry

thumbnail_geometry

mode

frame

返り値

*成功した場合に **TRUE** を返します。*

Imagick::morphImages

(No version information available, might be only in CVS)

Imagick::morphImages — 複数の画像をモーフィングする

説明

*Imagick **Imagick::morphImages** (int $number_frames)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

複数の画像をモーフィングします。 それぞれの画像のピクセルとサイズが線形補間され、 ある画像から次の画像へと変形していきます。

パラメータ

number_frames

返り値

このメソッドは、成功した場合に新しい Imagick オブジェクトを返します。 エラー時は ImagickException をスローします。

Imagick::compareImages

(No version information available, might be only in CVS)

Imagick::compareImages — ある画像を再構築された画像と比較する

説明

array **Imagick::compareImages** (Imagick $compare , int $metric)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

再構築された画像と画像の差異を含む配列を返します。

パラメータ

compare

比較したい画像。

metric

メトリック定数を指定します。この メトリック定数 の一覧を参照ください。

返り値

成功した場合に **TRUE** を返します。

例

Example#1 Imagick::compareImages() の使用法

画像を比較し、再構築された画像を表示します。

```
<?php

$image1 = new imagick("image1.png");
$image2 = new imagick("image2.png");

$result = $image1->compareImage($image2, Imagick::METRIC_MEANSQUAREERROR);
$result[0]->setImageFormat("png");

header("Content-Type: image/png");
echo $result[0];

?>
```

Imagick::mosaicImages

(No version information available, might be only in CVS)

Imagick::mosaicImages — 画像からモザイクを作成する

説明

Imagick **Imagick::mosaicImages** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像シーケンスをはめ込み合成して単一の画像にします。 シーケンス内の各画像を、画像のページオフセットで指定した位置に合成したものを返します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::motionBlurImage

(No version information available, might be only in CVS)

Imagick::motionBlurImage — モーションブラーをシミュレートする

説明

bool **Imagick::motionBlurImage** (float $radius , float $sigma , float $angle)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

モーションブラーをシミュレートします。 指定した半径と標準偏差（シグマ）によるガウス演算によって画像を畳み込みます。 意味のある結果を得るには、半径がシグマより大きくなければなりません。 radius に 0 を指定すると、MotionBlurImage() が適切な半径を自動的に設定します。 angle にはぼかしをかける角度を指定します。

パラメータ

radius

sigma

angle

返り値

成功した場合に **TRUE** を返します。

Imagick::negateImage

(No version information available, might be only in CVS)

Imagick::negateImage — 画像の色を打ち消す

説明

bool **Imagick::negateImage** (bool $gray [, int $channel])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の色を打ち消します。グレースケールオプションを指定すると、 画像内のグレースケール値のみを打ち消します。

パラメータ

gray

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::distortImage

(No version information available, might be only in CVS)

Imagick::distortImage — さまざまな方式で画像を歪める

説明

bool **Imagick::distortImage** (int $method , array $arguments , bool $bestfit)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

さまざまな方式で画像を歪めます。 これは、元画像の色を新たな画像の色に対応させることで行います。 新たな画像は、'bestfit' を true に設定しない限りは元画像と同じ大きさになります。 'bestfit' を有効にし、使用する歪め方式がサイズ変更を許可していた場合は、 変換後の画像に合わせて画像のサイズやオフセットが調整されます。 多くの場合、元画像の仮想オフセットも考慮したマッピングが行われます。 この機能は、Imagick を ImageMagick 6.3.6 以降でコンパイルした場合にのみ存在します。

パラメータ

method

画像の歪め方式。 歪め定数 を参照ください。

arguments

歪め方式の引数。

bestfit

元画像のサイズを変更する。

返り値

成功した場合に ***TRUE*** *を返します。*

エラー / 例外

エラー時に *ImagickException* *をスローします。*

例

Example#1 Imagick::distortImage() の使用法

画像を歪めてディスクに書き込みます。

```
<?php
$im = new imagick( "example.jpg" );
$im->distortImage( Imagick::DISTORTION_PERSPECTIVE, array( 7,40, 4,30, 4,124, 4,123, 85,122, 100,123, 85,2, 100,30 ), tru
$im->writeImage( "example_out.jpg" );
?>
```

参考

- *Imagick::blurImage()*
- *Imagick::motionBlurImage()*
- *Imagick::radialBlurImage()*

Imagick::newImage

(No version information available, might be only in CVS)

Imagick::newImage *— 新しい画像を作成する*

説明

bool ***Imagick::newImage*** *(int $cols , int $rows , ImagickPixel $background [, string $format])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい画像を作成し、ImagickPixel の値を背景色として関連付けます。

パラメータ

cols

新しい画像のカラム数。

rows

新しい画像の行数。

background

この画像で使用する背景色。

format

画像フォーマット。このパラメータは Imagick バージョン 2.0.1 で追加されました。

返り値

成功した場合に ***TRUE*** *を返します。*

例

Example#1 Imagick::newImage() の使用法

新しい画像を作成し、それを表示します。

```
<?php
$image = new Imagick();
```

```
$image->newImage(100, 100, new ImagickPixel('red'));
$image->setImageFormat('png');

header('Content-type:  image/png');
echo $image;

?>
```

Imagick::setImage

(No version information available, might be only in CVS)

Imagick::setImage — オブジェクト内の画像を置き換える

説明

bool **Imagick::setImage** (Imagick $replace)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の画像シーケンスを、replace で指定したオブジェクトの画像で置き換えます。

パラメータ

replace

置き換える Imagick オブジェクト。

返り値

成功した場合に **TRUE** を返します。

例

Example#1 Imagick::setImage() の例

Imagick::setImage() の使用例です。

```
<?php
/* オブジェクトを作成します */
$image = new Imagick('source.jpg');
$replace = new Imagick('replace.jpg');

/* source.jpg を replace.jpg で置き換えます */
$image->setImage($replace);

/* 画像を出力します */
header('Content-type:  image/jpeg');
echo $image;

?>
```

Imagick::setImageOpacity

(No version information available, might be only in CVS)

Imagick::setImageOpacity — 画像の不透明度を設定する

説明

bool **Imagick::setImageOpacity** (float $opacity)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の不透明度を設定します。このメソッドは、ImageMagick 6.3.1 以降を使用して Imagick をコンパイルした場合にのみ存在します。

パラメータ

opacity

透明度。1.0 が完全な不透明で、0.0 が完全な透明です。

返り値

成功した場合に **TRUE** を返します。

例

Example#1 Imagick::setImageOpacity() の例

Imagick::setImageOpacity() を使用する例です。

```
<?php
/* オブジェクトを作成します */
$image = new Imagick('source.png');

/* 透明度を設定します */
$image->setImageOpacity(0.7);

/* 画像を出力します */
header('Content-type: image/png');
echo $image;

?>
```

Imagick::newPseudoImage

(No version information available, might be only in CVS)

Imagick::newPseudoImage — 新しい画像を作成する

説明

bool **Imagick::newPseudoImage** (int $columns , int $rows , string $pseudoString)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい画像を、ImageMagick 疑似フォーマットを使用して作成します。

パラメータ

columns

新しい画像のカラム数。

rows

新しい画像の行数。

pseudoString

疑似画像定義を含む文字列。

返り値

成功した場合に **TRUE** を返します。

Imagick::nextImage

(No version information available, might be only in CVS)

Imagick::nextImage — 次の画像に移動する

説明

bool **Imagick::nextImage** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像リスト内の次の画像を Imagick オブジェクトに関連付けます。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::normalizeImage

(No version information available, might be only in CVS)

Imagick::normalizeImage — カラー画像のコントラストを強調する

説明

bool **Imagick::normalizeImage** ([int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

カラー画像のコントラストを強調します。 ピクセルの色を、とりうる値の範囲全体に拡張します。

パラメータ

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::oilPaintImage

(No version information available, might be only in CVS)

Imagick::oilPaintImage — 油絵をシミュレートする

説明

bool **Imagick::oilPaintImage** (float $radius)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

油絵風の効果を施すフィルタを適用します。 各ピクセルが、指定した半径内の周囲の中で最もよくあらわれる色に置き換えられます。

パラメータ

radius

返り値

成功した場合に **TRUE** を返します。

Imagick::optimizeImageLayers

(No version information available, might be only in CVS)

Imagick::optimizeImageLayers — 画像の繰り返し部分を削除して最適化する

説明

bool **Imagick::optimizeImageLayers** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

GIF 形式の画像を、シーケンス内のひとつ前の画像と比較します。 必要最小限の画像で各フレームを置き換え、結果のアニメーションはそのままとなるようにします。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::paintOpaqueImage

(No version information available, might be only in CVS)

Imagick::paintOpaqueImage — 色にマッチするピクセルを変更する

説明

bool **Imagick::paintOpaqueImage** (ImagickPixel $target , ImagickPixel $fill , float $fuzz [, int $channel])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

色にマッチするピクセルを、塗りつぶし色に変更します。

パラメータ

target

fill

fuzz

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::paintTransparentImage

(No version information available, might be only in CVS)

Imagick::paintTransparentImage — 色にマッチするピクセルを塗りつぶし色に変更する

説明

bool **Imagick::paintTransparentImage** (Imagick $targetObj , float $alpha , float $fuzz)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

targetObj

alpha

fuzz

返り値

成功した場合に **TRUE** を返します。

Imagick::pingImageBlob

(No version information available, might be only in CVS)

Imagick::pingImageBlob — 手早く属性を取得する

説明

bool **Imagick::pingImageBlob** (string $image)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドを使用すると、画像全体を読み込まなくても 画像の幅や高さ、サイズそしてフォーマットを取得できます。

パラメータ

image

返り値

成功した場合に **TRUE** を返します。

Imagick::pingImageFile

(No version information available, might be only in CVS)

Imagick::pingImageFile — 画像の基本属性を手軽に取得する

説明

bool **Imagick::pingImageFile** (resource $filehandle [, string $fileName])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドを使用すると、画像全体を読み込まなくても 画像の幅や高さ、サイズそしてフォーマットを取得できます。

パラメータ

filehandle

fileName

返り値

成功した場合に **TRUE** を返します。

Imagick::pingImage

(No version information available, might be only in CVS)

Imagick::pingImage — 画像の基本属性を取得する

説明

bool **Imagick::pingImage** (string $filename)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドを使用すると、画像全体を読み込まなくても 画像の幅や高さ、サイズそしてフォーマットを取得できます。

パラメータ

filename

返り値

成功した場合に **TRUE** を返します。

Imagick::posterizeImage

(No version information available, might be only in CVS)

Imagick::posterizeImage — 指定した色数まで画像を減色する

説明

bool **Imagick::posterizeImage** (int $levels , bool $dither)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した色数まで画像を減色します。

パラメータ

levels

dither

返り値

*成功した場合に **TRUE** を返します。*

Imagick::previousImage

(No version information available, might be only in CVS)

Imagick::previousImage — オブジェクト内の前の画像に移動する

説明

*bool **Imagick::previousImage** (void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像リスト内の前の画像を Imagick オブジェクトに関連付けます。

パラメータ

返り値

*成功した場合に **TRUE** を返します。*

Imagick::profileImage

(No version information available, might be only in CVS)

Imagick::profileImage — 画像のプロファイルを追加あるいは削除する

説明

*bool **Imagick::profileImage** (string $name , string $profile)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ICC、IPTC あるいは汎用のプロファイルを画像に追加あるいは削除します。 profile が NULL の場合は画像からプロファイルを削除し、 それ以外の場合は追加します。 name に ''、profile に NULL を指定すると、画像からすべてのプロパティを削除します。*

パラメータ

name

profile

返り値

*成功した場合に **TRUE** を返します。*

Imagick::queryFormats

(No version information available, might be only in CVS)

Imagick::queryFormats — Imagick がサポートするフォーマットを返す

説明

*array **Imagick::queryFormats** ([string $pattern])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick がサポートするフォーマットを返します。

パラメータ

pattern

返り値

Imagick がサポートするフォーマットを含む配列を返します。 エラー時に ImagickException をスローします。

Imagick::radialBlurImage

(No version information available, might be only in CVS)

Imagick::radialBlurImage — 画像にラジアルブラーを施す

説明

bool **Imagick::radialBlurImage** (float $angle [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像にラジアルブラーを施します。

パラメータ

angle

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::raiseImage

(No version information available, might be only in CVS)

Imagick::raiseImage — 三次元のボタン風の効果をシミュレートする

説明

bool **Imagick::raiseImage** (int $width , int $height , int $x , int $y , bool $raise)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の輪郭を明るくしたり暗くしたりすることによって、 三次元のボタン風の効果をシミュレートします。 raise_info のメンバ width および height が、 垂直方向および水平方向の効果の輪郭を定義します。

パラメータ

width

height

x

y

raise

返り値

成功した場合に **TRUE** を返します。

Imagick::randomThresholdImage

(No version information available, might be only in CVS)

Imagick::randomThresholdImage — コントラストの高い 2 色の画像を作成する

説明

bool **Imagick::randomThresholdImage** (float $low , float $high [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

個々のピクセルの輝度を閾値と比較し、その色を変更します。 変換結果は、コントラストの高い 2 色の画像となります。

パラメータ

low

high

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::readImageBlob

(No version information available, might be only in CVS)

Imagick::readImageBlob — バイナリ文字列から画像を読み込む

説明

bool **Imagick::readImageBlob** (string $image [, string $filename])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

バイナリ文字列から画像を読み込みます。

パラメータ

image

返り値

成功した場合に **TRUE** を返します。

Imagick::readImageFile

(No version information available, might be only in CVS)

Imagick::readImageFile — オープンしているファイルハンドルから画像を読み込む

説明

bool **Imagick::readImageFile** (resource $filehandle [, string $fileName])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オープンしているファイルハンドルから画像を読み込みます。

パラメータ

filehandle

fileName

返り値

成功した場合に **TRUE** を返します。

Imagick::readImage

(PECL imagick:0.9-0.9.9)

Imagick::readImage — ファイルから画像を読み込む

説明

bool **Imagick::readImage** (string $filename)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ファイルから画像を読み込みます。

パラメータ

filename

返り値

成功した場合に **TRUE** を返します。

Imagick::reduceNoiseImage

(No version information available, might be only in CVS)

Imagick::reduceNoiseImage — 画像の輪郭をなめらかにする

説明

bool **Imagick::reduceNoiseImage** (float $radius)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の輪郭をなめらかにします。縁の情報は保持します。 各ピクセルの内容を近傍の一番近い値で置き換えるというアルゴリズムです。 近傍は半径で定義します。radius に 0 を指定すると、 Imagick::reduceNoiseImage() が適切な半径を自動的に設定します。

パラメータ

radius

返り値

成功した場合に **TRUE** を返します。

Imagick::removeImageProfile

(No version information available, might be only in CVS)

Imagick::removeImageProfile — 指定した名前の画像プロファイルを削除してそれを返す

説明

string **Imagick::removeImageProfile** (string $name)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した名前の画像プロファイルを削除してそれを返します。

パラメータ

name

返り値

画像のプロファイル名を返します。 エラー時に ImagickException をスローします。

Imagick::removeImage

(No version information available, might be only in CVS)

Imagick::removeImage — 画像リストから画像を削除する

説明

bool **Imagick::removeImage** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像リストから画像を削除します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::render

(No version information available, might be only in CVS)

Imagick::render — それまでのすべての描画コマンドをレンダリングする

説明

bool **Imagick::render** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

それまでのすべての描画コマンドをレンダリングします。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::resampleImage

(No version information available, might be only in CVS)

Imagick::resampleImage — 画像を指定した解像度にリサンプリングする

説明

bool **Imagick::resampleImage** (float $x_resolution , float $y_resolution , int $filter , float $blur)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像を指定した解像度にリサンプリングします。

パラメータ

x_resolution

y_resolution

filter

blur

返り値

成功した場合に **TRUE** を返します。

Imagick::resizeImage

(No version information available, might be only in CVS)

Imagick::resizeImage — 画像のサイズを変更する

説明

bool **Imagick::resizeImage** (float $columns , float $rows , int $filter , float $blur)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した大きさと フィルタ で、画像のサイズを変更します。

パラメータ

columns

rows

filter

blur

返り値

成功した場合に **TRUE** を返します。

Imagick::rollImage

(No version information available, might be only in CVS)

Imagick::rollImage — 画像を補正する

説明

bool **Imagick::rollImage** (int $x , int $y)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

x および y を指定して画像を補正します。

パラメータ

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::rotateImage

(No version information available, might be only in CVS)

Imagick::rotateImage — 画像を回転する

説明

bool **Imagick::rotateImage** (ImagickPixel $background , float $degrees)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した角度だけ画像を回転させます。回転によって生じる空き領域は、 背景色で埋められます。

パラメータ

background

degrees

返り値

成功した場合に **TRUE** を返します。

Imagick::sampleImage

(No version information available, might be only in CVS)

Imagick::sampleImage — ピクセルのサンプリングによって画像の倍率を変更する

説明

bool **Imagick::sampleImage** (int $columns , int $rows)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ピクセルのサンプリングによって、指定した大きさに画像の倍率を変更します。 他の拡大・縮小メソッドとは異なり、 このメソッドは変更後の画像に新たな色を追加したりしません。

パラメータ

columns

rows

返り値

成功した場合に **TRUE** を返します。

Imagick::scaleImage

(No version information available, might be only in CVS)

Imagick::scaleImage — 画像のサイズを変更する

説明

bool **Imagick::scaleImage** (int $cols , int $rows)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のサイズを指定した大きさに変更します。 パラメータに 0 を指定すると、そのパラメータを自動的に計算します。

パラメータ

cols

rows

返り値

成功した場合に **TRUE** を返します。

Imagick::separateImageChannel

(No version information available, might be only in CVS)

Imagick::separateImageChannel — 画像からチャネルを分離する

説明

bool **Imagick::separateImageChannel** (int $channel)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像からチャネルを分離し、グレースケール画像を返します。 channel には、画像の各ピクセルの特定の色コンポーネントを指定します。

パラメータ

channel

返り値

成功した場合に TRUE を返します。

Imagick::sepiaToneImage

(No version information available, might be only in CVS)

Imagick::sepiaToneImage — 画像をセピア調にする

説明

bool **Imagick::sepiaToneImage** *(float $threshold)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像に特殊効果を施し、セピア調の写真のような画像にします。 threshold には 0 から QuantumRange までの値を指定し、 これがセピア調の効き具合を表します。threshold を 80 にすると、 とりあえずはそれなりの効き目になるでしょう。

パラメータ

threshold

返り値

成功した場合に TRUE を返します。

Imagick::setBackgroundColor

(No version information available, might be only in CVS)

Imagick::setBackgroundColor — オブジェクトのデフォルト背景色を設定する

説明

bool **Imagick::setBackgroundColor** *(ImagickPixel $background)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトのデフォルト背景色を設定します。

パラメータ

background

返り値

成功した場合に TRUE を返します。

Imagick::setCompressionQuality

(PECL imagick:0.9.10-0.9.9)

Imagick::setCompressionQuality — オブジェクトのデフォルトの圧縮品質を設定する

説明

bool **Imagick::setCompressionQuality** (*int $quality*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトのデフォルトの圧縮品質を設定します。

パラメータ

quality

返り値

成功した場合に **TRUE** を返します。

Imagick::setCompression

(No version information available, might be only in CVS)

Imagick::setCompression — オブジェクトのデフォルトの圧縮方式を設定する

説明

bool **Imagick::setCompression** (*int $compression*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトのデフォルトの圧縮方式を設定します。

パラメータ

compression

返り値

成功した場合に **TRUE** を返します。

Imagick::setFilename

(No version information available, might be only in CVS)

Imagick::setFilename — 画像を読み書きする前にファイル名を設定する

説明

bool **Imagick::setFilename** (*string $filename*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像を読み書きする前にファイル名を設定します。

パラメータ

filename

返り値

成功した場合に **TRUE** を返します。

Imagick::setFirstIterator

(No version information available, might be only in CVS)

Imagick::setFirstIterator — Imagick イテレータを最初の画像に設定する

説明

bool ***Imagick::setFirstIterator*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick イテレータを最初の画像に設定します。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setLastIterator

(No version information available, might be only in CVS)

Imagick::setLastIterator — Imagick イテレータを最後の画像に設定する

説明

bool ***Imagick::setLastIterator*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick イテレータを最後の画像に設定します。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setFormat

(No version information available, might be only in CVS)

Imagick::setFormat — Imagick オブジェクトのフォーマットを設定する

説明

bool ***Imagick::setFormat*** *(string $format)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトのフォーマットを設定します。

パラメータ

format

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setImageBackgroundColor

(No version information available, might be only in CVS)

Imagick::setImageBackgroundColor — 画像の背景色を設定する

説明

bool **Imagick::setImageBackgroundColor** (ImagickPixel $background)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の背景色を設定します。

パラメータ

background

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageBias

(No version information available, might be only in CVS)

Imagick::setImageBias — 画像を折りたたむ任意のメソッドについて画像のバイアスを設定する

説明

bool **Imagick::setImageBias** (float $bias)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像を折りたたむ任意のメソッド (Imagick::ConvolveImage() など) について画像のバイアスを設定します。

パラメータ

bias

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageBluePrimary

(No version information available, might be only in CVS)

Imagick::setImageBluePrimary — 青が一番強い点を設定する

説明

bool **Imagick::setImageBluePrimary** (float $x , float $y)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

青が一番強い点を設定します。

パラメータ

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageBorderColor

(No version information available, might be only in CVS)

Imagick::setImageBorderColor — *画像の前景色を設定する*

説明

bool ***Imagick::setImageBorderColor*** *(ImagickPixel $border)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の前景色を設定します。

パラメータ

border

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setImageChannelDepth

(No version information available, might be only in CVS)

Imagick::setImageChannelDepth — *特定の画像チャネルの深度を設定する*

説明

bool ***Imagick::setImageChannelDepth*** *(int $channel , int $depth)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

特定の画像チャネルの深度を設定します。

パラメータ

channel

depth

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setImageColormapColor

(No version information available, might be only in CVS)

Imagick::setImageColormapColor — *指定した色マップインデックスの色を設定する*

説明

bool ***Imagick::setImageColormapColor*** *(int $index , ImagickPixel $color)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した色マップインデックスの色を設定します。

パラメータ

index

color

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setImageColorspace

(No version information available, might be only in CVS)

Imagick::setImageColorspace — 画像の色空間を設定する

説明

bool **Imagick::setImageColorspace** (int $colorspace)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の色空間を設定します。

パラメータ

colorspace

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageCompose

(No version information available, might be only in CVS)

Imagick::setImageCompose — 画像の合成演算子を設定する

説明

bool **Imagick::setImageCompose** (int $compose)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の合成演算子を設定します。Imagick::montageImage() メソッドを使用して画像のサムネイルを作成する方法を指定する際に便利です。

パラメータ

compose

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageCompression

(No version information available, might be only in CVS)

Imagick::setImageCompression — 画像の圧縮を設定する

説明

bool **Imagick::setImageCompression** (int $compression)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の圧縮を設定します。

パラメータ

compression

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageDelay

(No version information available, might be only in CVS)

Imagick::setImageDelay — 画像の遅延を設定する

説明

bool **Imagick::setImageDelay** (int $delay)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の遅延を設定します。

パラメータ

delay

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageDepth

(No version information available, might be only in CVS)

Imagick::setImageDepth — 画像の深度を設定する

説明

bool **Imagick::setImageDepth** (int $depth)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の深度を設定します。

パラメータ

depth

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageDispose

(No version information available, might be only in CVS)

Imagick::setImageDispose — 画像の配置方法を設定する

説明

bool **Imagick::setImageDispose** (int $dispose)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の配置方法を設定します。

パラメータ

dispose

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageExtent

(No version information available, might be only in CVS)

Imagick::setImageExtent — 画像のサイズを設定する

説明

bool **Imagick::setImageExtent** (int $columns , int $rows)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のサイズ（つまりカラム数と行数）を設定します。

パラメータ

columns

rows

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageFilename

(No version information available, might be only in CVS)

Imagick::setImageFilename — 特定の画像のファイル名を設定する

説明

bool **Imagick::setImageFilename** (string $filename)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

シーケンス内の特定の画像のファイル名を設定します。

パラメータ

filename

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageFormat

(No version information available, might be only in CVS)

Imagick::setImageFormat — 特定の画像のフォーマットを設定する

説明

bool **Imagick::setImageFormat** (string $format)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

シーケンス内の特定の画像のフォーマットを設定します。

パラメータ

format

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageGamma

(No version information available, might be only in CVS)

Imagick::setImageGamma — 画像のガンマを設定する

説明

bool **Imagick::setImageGamma** (float $gamma)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のガンマを設定します。

パラメータ

gamma

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageGreenPrimary

(No version information available, might be only in CVS)

Imagick::setImageGreenPrimary — 緑が一番強い点を設定する

説明

bool **Imagick::setImageGreenPrimary** (float $x , float $y)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

緑が一番強い点を設定します。

パラメータ

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageIndex

(No version information available, might be only in CVS)

Imagick::setImageIndex — イテレータの位置を設定する

説明

bool **Imagick::setImageIndex** (int $index)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像リスト内でのイテレータの位置、index パラメータで指定した場所に設定します。 このメソッドは非推奨です。 Imagick::setIteratorIndex を参照ください。

パラメータ

index

イテレータを設定する位置。

返り値

成功した場合に **TRUE** を返します。

Imagick::setIteratorIndex

(No version information available, might be only in CVS)

Imagick::setIteratorIndex — イテレータの位置を設定する

説明

bool **Imagick::setIteratorIndex** (*int $index*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な *PHP* のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

イテレータの位置を、画像リスト内の *index* で指定した位置に設定します。

パラメータ

index

イテレータを設定する場所。

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageInterlaceScheme

(No version information available, might be only in CVS)

Imagick::setImageInterlaceScheme — 画像のインターレース手法を設定する

説明

bool **Imagick::setImageInterlaceScheme** (*int $interlace_scheme*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な *PHP* のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の圧縮を設定します。

パラメータ

interlace_scheme

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageInterpolateMethod

(No version information available, might be only in CVS)

Imagick::setImageInterpolateMethod — 画像のピクセル補間方式を設定する

説明

bool **Imagick::setImageInterpolateMethod** (*int $method*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のピクセル補間方式を設定します。

パラメータ

method

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setImageIterations

(No version information available, might be only in CVS)

Imagick::setImageIterations — 画像の反復を設定する

説明

bool ***Imagick::setImageIterations*** *(int $iterations)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の反復を設定します。

パラメータ

iterations

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setImageMatteColor

(No version information available, might be only in CVS)

Imagick::setImageMatteColor — 画像のマット色を設定する

説明

bool ***Imagick::setImageMatteColor*** *(ImagickPixel $matte)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のマット色を設定します。

パラメータ

matte

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setImageMatte

(No version information available, might be only in CVS)

Imagick::setImageMatte — 画像のマットチャネルを設定する

説明

bool ***Imagick::setImageMatte*** *(bool $matte)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のマットチャネルを設定します。

パラメータ

matte

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setImagePage

(No version information available, might be only in CVS)

Imagick::setImagePage — 画像のページのジオメトリを設定する

説明

*bool **Imagick::setImagePage** (int $width , int $height , int $x , int $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のページのジオメトリを設定します。

パラメータ

width

height

x

y

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setImageProfile

(No version information available, might be only in CVS)

Imagick::setImageProfile — 指定した名前の画像プロファイルを Imagick オブジェクトに追加する

説明

*bool **Imagick::setImageProfile** (string $name , string $profile)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した名前の画像プロファイルを Imagick オブジェクトに追加します。 同じ名前のプロファイルが存在する場合は、それを置き換えます。 このメソッドは Imagick::ProfileImage() メソッドとは異なり、 CMS カラープロファイルは適用しません。

パラメータ

name

profile

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setImageProperty

(No version information available, might be only in CVS)

Imagick::setImageProperty — *画像のプロパティを設定する*

説明

bool ***Imagick::setImageProperty*** *(string $name , string $value)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した名前のプロパティを画像に設定します。

パラメータ

name

value

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setImageRedPrimary

(No version information available, might be only in CVS)

Imagick::setImageRedPrimary — *赤が一番強い点を設定する*

説明

bool ***Imagick::setImageRedPrimary*** *(float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

赤が一番強い点を設定します。

パラメータ

x

y

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setImageRenderingIntent

(No version information available, might be only in CVS)

Imagick::setImageRenderingIntent — *画像のレンダリング方向を設定する*

説明

bool ***Imagick::setImageRenderingIntent*** *(int $rendering_intent)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のレンダリング方向を設定します。

パラメータ

rendering_intent

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::setImageResolution

(No version information available, might be only in CVS)

Imagick::setImageResolution — 画像の解像度を設定する

説明

bool **Imagick::setImageResolution** (float $x_resolution , float $y_resolution)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の解像度を設定します。

パラメータ

x_resolution

y_resolution

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageScene

(No version information available, might be only in CVS)

Imagick::setImageScene — 画像のシーンを設定する

説明

bool **Imagick::setImageScene** (int $scene)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のシーンを設定します。

パラメータ

scene

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageTicksPerSecond

(No version information available, might be only in CVS)

Imagick::setImageTicksPerSecond — 画像の ticks-per-second を設定する

説明

bool **Imagick::setImageTicksPerSecond** (int $ticks_per-second)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の ticks-per-second を設定します。

パラメータ

ticks_per-second

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageType

(No version information available, might be only in CVS)

Imagick::setImageType — 画像の型を設定する

説明

bool **Imagick::setImageType** (int $image_type)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の型を設定します。

パラメータ

image_type

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageUnits

(No version information available, might be only in CVS)

Imagick::setImageUnits — 画像の解像度の単位を設定する

説明

bool **Imagick::setImageUnits** (int $units)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の解像度の単位を設定します。

パラメータ

units

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageVirtualPixelMethod

(No version information available, might be only in CVS)

Imagick::setImageVirtualPixelMethod — 画像の仮想ピクセルメソッドを設定する

説明

bool **Imagick::setImageVirtualPixelMethod** (int $method)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の仮想ピクセルメソッドを設定します。

パラメータ

method

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageWhitePoint

(No version information available, might be only in CVS)

Imagick::setImageWhitePoint — 画像の色度が白い点を設定する

説明

bool **Imagick::setImageWhitePoint** (float $x , float $y)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の色度が白い点を設定します。

パラメータ

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::setInterlaceScheme

(No version information available, might be only in CVS)

Imagick::setInterlaceScheme — 画像の圧縮を設定する

説明

bool **Imagick::setInterlaceScheme** (int $interlace_scheme)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の圧縮を設定します。

パラメータ

interlace_scheme

返り値

成功した場合に **TRUE** を返します。

Imagick::setImageOrientation

(No version information available, might be only in CVS)

Imagick::setImageOrientation — 画像の方向を設定する

説明

bool **Imagick::setImageOrientation** (int $orientation)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の方向を設定します。

パラメータ

orientation

方向定数 のいずれか。

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setOption

(No version information available, might be only in CVS)

Imagick::setOption — オプションを設定する

説明

*bool **Imagick::setOption** (string $key , string $value)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ひとつあるいは複数のオプションを設定します。

パラメータ

key

value

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setPage

(No version information available, might be only in CVS)

Imagick::setPage — Imagick オブジェクトのページジオメトリを設定する

説明

*bool **Imagick::setPage** (int $width , int $height , int $x , int $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトのページジオメトリを設定します。

パラメータ

width

height

x

y

返り値

*成功した場合に **TRUE** を返します。*

Imagick::setResolution

(No version information available, might be only in CVS)

Imagick::setResolution — 画像の解像度を設定する

説明

*bool **Imagick::setResolution** (float $x_resolution , float $y_resolution)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の解像度を設定します。

パラメータ

x_resolution

y_resolution

返り値

成功した場合に **TRUE** を返します。

Imagick::setResourceLimit

(No version information available, might be only in CVS)

Imagick::setResourceLimit — 特定のリソースの制限をメガバイト単位で設定する

説明

bool **Imagick::setResourceLimit** (int $type , int $limit)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

特定のリソースの制限をメガバイト単位で設定します。

パラメータ

type

limit

返り値

成功した場合に **TRUE** を返します。

Imagick::setSamplingFactors

(No version information available, might be only in CVS)

Imagick::setSamplingFactors — 画像のサンプリング係数を設定する

説明

bool **Imagick::setSamplingFactors** (array $factors)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のサンプリング係数を設定します。

パラメータ

factors

返り値

成功した場合に **TRUE** を返します。

Imagick::setSizeOffset

(No version information available, might be only in CVS)

Imagick::setSizeOffset — Imagick オブジェクトのサイズのオフセットを設定する

説明

bool **Imagick::setSizeOffset** (*int $columns* , *int $rows* , *int $offset*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトのサイズとオフセットを設定します。 RGB や GRAY、CMYK といった生の画像フォーマットを読み込む前に設定します。

パラメータ

columns

rows

offset

返り値

成功した場合に **TRUE** を返します。

Imagick::setSize

(No version information available, might be only in CVS)

Imagick::setSize — Imagick オブジェクトのサイズを設定する

説明

bool **Imagick::setSize** (*int $columns* , *int $rows*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick オブジェクトのサイズを設定します。 RGB や GRAY、CMYK といった生の画像フォーマットを読み込む前に設定します。

パラメータ

columns

rows

返り値

成功した場合に **TRUE** を返します。

Imagick::setType

(No version information available, might be only in CVS)

Imagick::setType — 画像タイプ属性を設定する

説明

bool **Imagick::setType** (*int $image_type*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像タイプ属性を設定します。

パラメータ

image_type

返り値

成功した場合に **TRUE** を返します。

Imagick::shadeImage

(No version information available, might be only in CVS)

Imagick::shadeImage — 3D 効果を作成する

説明

bool **Imagick::shadeImage** (bool $gray , float $azimuth , float $elevation)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

遠くから光をあてているように見せる三次元効果を作成します。 光源の位置は azimuth と elevation で指定します。 azimuth は x 軸方向の角度で、elevation は Z 軸上のピクセル数です。

パラメータ

gray

azimuth

elevation

返り値

成功した場合に **TRUE** を返します。

Imagick::shadowImage

(No version information available, might be only in CVS)

Imagick::shadowImage — 画像の影をシミュレートする

説明

bool **Imagick::shadowImage** (float $opacity , float $sigma , int $x , int $y)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の影をシミュレートします。

パラメータ

opacity

sigma

x

y

返り値

成功した場合に **TRUE** を返します。

Imagick::sharpenImage

(No version information available, might be only in CVS)

Imagick::sharpenImage — 画像をシャープにする

説明

bool **Imagick::sharpenImage** (float $radius , float $sigma [, int $channel])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像をシャープにします。指定した半径と標準偏差（シグマ）によるガウス演算を施します。適切な結果を得るには、 半径をシグマより大きくする必

要があります。半径を 0 に指定すると、適切な半径を自動的に設定します。

パラメータ

radius

sigma

channel

返り値

成功した場合に TRUE を返します。

Imagick::shaveImage

(No version information available, might be only in CVS)

Imagick::shaveImage — 画像の輪郭からピクセルを刈り取る

説明

bool **Imagick::shaveImage** (int $columns , int $rows)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の輪郭からピクセルを刈り取ります。 新しい画像に必要なメモリを確保し、新しい画像へのポインタを返します。

パラメータ

columns

rows

返り値

成功した場合に TRUE を返します。

Imagick::shearImage

(No version information available, might be only in CVS)

Imagick::shearImage — 平行四辺形を作成する

説明

bool **Imagick::shearImage** (ImagickPixel $background , float $x_shear , float $y_shear)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像を X 軸方向あるいは Y 軸方向に押しつぶし、平行四辺形を作成します。 X 方向に押しつぶすと X 軸方向にスライドし、Y 方向に押しつぶすと Y 軸方向にスライドします。押しつぶす量は、角度で指定します。 X 方向に押しつぶす場合、x_shear は Y 軸からの相対角度となります。 同様に、Y 方向に押しつぶす場合は y_shear が X 軸からの相対角度となります。 変形によってできた空白部分は、背景色で埋められます。

パラメータ

background

x_shear

y_shear

返り値

成功した場合に TRUE を返します。

Imagick::sigmoidalContrastImage

(No version information available, might be only in CVS)

Imagick::sigmoidalContrastImage — 画像のコントラストを調整する

説明

bool **Imagick::sigmoidalContrastImage** (bool $sharpen , float $alpha , float $beta [, int $channel])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

非線形 S 字コントラストアルゴリズムを用いて画像のコントラストを調整します。 S 字変換関数を使用し、ハイライトや影を飽和させることなく画像のコントラストを強調します。 contrast は、どの程度コントラストを向上させるかを表します (0 は何もしない、3 は一般的な程度、20 は押し出す) mid-point は、結果の画像での中間色の扱いを表します (0 は白、50 はグレー、100 は黒)。sharpen を true にするとコントラストを強め、それ以外にするとコントラストを弱めます。

パラメータ

sharpen

alpha

beta

channel

返り値

成功した場合に **TRUE** を返します。

Imagick::sketchImage

(No version information available, might be only in CVS)

Imagick::sketchImage — 鉛筆画をシミュレートする

説明

bool **Imagick::sketchImage** (float $radius , float $sigma , float $angle)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

鉛筆画をシミュレートします。 指定した半径と標準偏差（シグマ）によるガウス演算によって画像を畳み込みます。 意味のある結果を得るには、半径がシグマより大きくなければなりません。 radius に 0 を指定すると、 Imagick::sketchImage() が適切な半径を自動的に設定します。 angle で、モーションブラーの角度を指定します。

パラメータ

radius

sigma

angle

返り値

成功した場合に **TRUE** を返します。

Imagick::solarizeImage

(No version information available, might be only in CVS)

Imagick::solarizeImage — 画像にソラリゼーション効果を適用する

説明

bool **Imagick::solarizeImage** (int $threshold)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像に特殊効果を施し、 印画紙の特定の箇所の露光時間を長くしたような画像になります。 threshold は 0 から QuantumRange までの値で、 ソラリゼーションの効き具合を指定します。

パラメータ

threshold

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::spliceImage

(No version information available, might be only in CVS)

Imagick::spliceImage — *無地の画像を作成する*

説明

bool ***Imagick::spliceImage*** *(int $width , int $height , int $x , int $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

無地の画像を作成します。

パラメータ

width

height

x

y

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::spreadImage

(No version information available, might be only in CVS)

Imagick::spreadImage — *ブロック内の各ピクセルをランダムに移動する*

説明

bool ***Imagick::spreadImage*** *(float $radius)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

radius パラメータで定義したブロック内の各ピクセルを、 ランダムに移動させます。

パラメータ

radius

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::steganoImage

(No version information available, might be only in CVS)

Imagick::steganoImage — *デジタル透かしを画像に埋め込む*

説明

Imagick ***Imagick::steganoImage*** *(Imagick $watermark_wand , int $offset)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

デジタル透かしを画像に埋め込みます。 あとでこの透かしが見つかれば、その画像の出所を証明することができます。 offset は、透かしの埋め込みを開始する位置を指定します。

パラメータ

watermark_wand

offset

返り値

*成功した場合に **TRUE** を返します。*

Imagick::stereoImage

(No version information available, might be only in CVS)

Imagick::stereoImage — ふたつの画像を合成する

説明

bool ***Imagick::stereoImage*** *(Imagick $offset_wand)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ふたつの画像を合成してひとつの画像にします。 左と右の画像を合成してステレオ効果を生み出します。

パラメータ

offset_wand

返り値

*成功した場合に **TRUE** を返します。*

Imagick::stripImage

(No version information available, might be only in CVS)

Imagick::stripImage — 画像からすべてのプロパティやコメントを除去する

説明

bool ***Imagick::stripImage*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像からすべてのプロパティやコメントを除去します。

パラメータ

返り値

*成功した場合に **TRUE** を返します。*

Imagick::swirlImage

(No version information available, might be only in CVS)

Imagick::swirlImage — 画像の中心から、ピクセルを渦巻状にする

説明

bool ***Imagick::swirlImage*** *(float $degrees)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の中心から、ピクセルを渦巻状にします。 degrees は個々のピクセルが移動する弧の量を表します。 degrees を 1 から 360 まで変化させると、劇的な効果が得られます。

パラメータ

degrees

返り値

*成功した場合に **TRUE** を返します。*

Imagick::textureImage

(No version information available, might be only in CVS)

Imagick::textureImage — テクスチャ画像をタイル状に並べる

説明

*bool **Imagick::textureImage** (Imagick $texture_wand)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テクスチャ画像をタイル状に繰り返し並べます。

パラメータ

texture_wand

返り値

*成功した場合に **TRUE** を返します。*

Imagick::thresholdImage

(No version information available, might be only in CVS)

Imagick::thresholdImage — 閾値にもとづいて個々のピクセルの値を変更する

説明

*bool **Imagick::thresholdImage** (float $threshold [, int $channel])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

個々のピクセルの輝度を閾値と比較し、その色を変更します。 変換結果は、コントラストの高い 2 色の画像となります。

パラメータ

threshold

channel

返り値

*成功した場合に **TRUE** を返します。*

Imagick::thumbnailImage

(No version information available, might be only in CVS)

Imagick::thumbnailImage — 画像のサイズを変更する

説明

bool ***Imagick::thumbnailImage*** *(int $columns , int $rows [, bool $fit])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像のサイズを指定したものに変更し、関連付けられたプロパティをすべて削除します。 ウェブ上での表示に適した小さなサムネイル画像を作成します。 3 番目のパラメータに true を指定すると、columns や rows にそれぞれの最大値を使用します。両方のパラメータが、 マッチするまであるいは指定したパラメータより小さくなるまで縮小されます。

パラメータ

columns

画像の幅。

rows

画像の高さ。

fit

最大値を強制的に使用するかどうか。

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::tintImage

(No version information available, might be only in CVS)

Imagick::tintImage — *色ベクトルを画像の各ピクセルに適用する*

説明

bool ***Imagick::tintImage*** *(ImagickPixel $tint , ImagickPixel $opacity)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

色ベクトルを画像の各ピクセルに適用します。 ベクトルの長さは、黒および白のときに 0、中間色のときに最大となります。 ベクトルの重み関数は f(x)=(1-(4.0((x-0.5)*(x-0.5)))) です。*

パラメータ

tint

opacity

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::transverseImage

(No version information available, might be only in CVS)

Imagick::transverseImage — *水平方向に反転させた画像を作成する*

説明

bool ***Imagick::transverseImage*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

水平方向に反転させた画像を作成します。 中央の y 軸に対して反転させ、270 度回転させます。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::trimImage

(No version information available, might be only in CVS)

Imagick::trimImage — 画像の輪郭を削除する

説明

bool **Imagick::trimImage** (float $fuzz)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の背景色と同じ色の輪郭を削除します。

パラメータ

fuzz

返り値

成功した場合に **TRUE** を返します。

Imagick::uniqueImageColors

(No version information available, might be only in CVS)

Imagick::uniqueImageColors — ある 1 色以外のすべての色のピクセルを削除する

説明

bool **Imagick::uniqueImageColors** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ある 1 色以外のすべての色のピクセルを削除します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

Imagick::unsharpMaskImage

(No version information available, might be only in CVS)

Imagick::unsharpMaskImage — 画像をシャープにする

説明

bool **Imagick::unsharpMaskImage** (float $radius , float $sigma , float $amount , float $threshold [, int $channel])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像をシャープにします。 指定した半径と標準偏差（シグマ）によるガウス演算によって画像を畳み込みます。 意味のある結果を得るには、半径がシグマより大きくなければなりません。 radius に 0 を指定すると、 Imagick::UnsharpMaskImage() が適切な半径を自動的に設定します。

パラメータ

radius

sigma

amount

threshold

channel

返り値

*成功した場合に **TRUE** を返します。*

Imagick::valid

(No version information available, might be only in CVS)

Imagick::valid — 現在のアイテムが有効かどうかを調べる

説明

*bool **Imagick::valid** (void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在のアイテムが有効かどうかを調べます。

パラメータ

返り値

*成功した場合に **TRUE** を返します。*

Imagick::vignetteImage

(No version information available, might be only in CVS)

Imagick::vignetteImage — ビネットフィルタを画像に追加する

説明

*bool **Imagick::vignetteImage** (float $blackPoint , float $whitePoint , int $x , int $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ビネットフィルタを画像に追加します。

パラメータ

blackPoint

whitePoint

x

y

返り値

*成功した場合に **TRUE** を返します。*

Imagick::waveImage

(No version information available, might be only in CVS)

Imagick::waveImage — ウェーブフィルタを画像に追加する

説明

*bool **Imagick::waveImage** (float $amplitude , float $length)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ウェーブフィルタを画像に追加します。

パラメータ

amplitude

length

返り値

*成功した場合に **TRUE** を返します。*

Imagick::whiteThresholdImage

(No version information available, might be only in CVS)

Imagick::whiteThresholdImage — 閾値に満たないすべてのピクセルを白にする

説明

*bool **Imagick::whiteThresholdImage** (ImagickPixel $threshold)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick::thresholdImage() と似ていますが、 これは閾値に満たないピクセルをすべて白にし、その他のピクセルはそのままにします。

パラメータ

threshold

返り値

*成功した場合に **TRUE** を返します。*

Imagick::writeImages

(PECL imagick:0.9-0.9.9)

Imagick::writeImages — 画像あるいは画像シーケンスを書き込む

説明

*bool **Imagick::writeImages** (string $filename , bool $adjoin)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像あるいは画像シーケンスを書き込みます。

パラメータ

filename

adjoin

返り値

*成功した場合に **TRUE** を返します。*

Imagick::writeImage

(PECL imagick:0.9-0.9.9)

Imagick::writeImage — 指定した名前で画像を書き込む

説明

bool **Imagick::writeImage** ([string $filename])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した名前で画像を書き込みます。filename パラメータが NULL の場合は、 Imagick::ReadImage() あるいは Imagick::SetImageFilename() で設定した名前で書き込みます。

パラメータ

filename

返り値

成功した場合に **TRUE** を返します。

Imagick::displayImage

(No version information available, might be only in CVS)

Imagick::displayImage — 画像を表示する

説明

bool **Imagick::displayImage** (string $servername)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドは、画像を X サーバに表示します。

パラメータ

servername

X サーバの名前。

返り値

成功した場合に **TRUE** を返します。

Imagick::displayImages

(No version information available, might be only in CVS)

Imagick::displayImages — 画像あるいは画像シーケンスを表示する

説明

bool **Imagick::displayImages** (string $servername)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像あるいは画像シーケンスを X サーバに表示します。

パラメータ

servername

X サーバの名前。

返り値

成功した場合に **TRUE** を返します。

Imagick::cropThumbnailImage

(No version information available, might be only in CVS)

Imagick::cropThumbnailImage — 切り取ってサムネイルを作成する

説明

bool **Imagick::cropThumbnailImage** (int $width , int $height)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

固定サイズのサムネイルを作成します。 まず画像のサイズを縮小し、指定した範囲を中心から切り取ります。

パラメータ

width

サムネイルの幅。

height

サムネイルの高さ。

返り値

成功した場合に **TRUE** を返します。

Imagick::roundCorners

(No version information available, might be only in CVS)

Imagick::roundCorners — 画像の角を丸める

説明

bool **Imagick::roundCorners** (float $x_rounding , float $y_rounding [, float $stroke_width [, float $displace [, float $size_correction]]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の角を丸めます。最後の 3 つのパラメータはオプションで、めったに使用することはないでしょう。

パラメータ

x_rounding

x 方向の丸め。

y_rounding

y 方向の丸め。

stroke_width

線幅。

displace

image displace

size_correction

サイズ補正。

返り値

成功した場合に **TRUE** を返します。

Imagick::polaroidImage

(No version information available, might be only in CVS)

Imagick::polaroidImage — ポラロイド写真をシミュレートする

説明

bool ***Imagick::polaroidImage*** *(ImagickDraw $properties , float $angle)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ポラロイド写真をシミュレートします。

パラメータ

properties

ポラロイドプロパティ。

angle

ポラロイド角度。

返り値

成功した場合に ***TRUE*** *を返します。*

Imagick::queryFonts

(No version information available, might be only in CVS)

Imagick::queryFonts — 設定したフォントを返す

説明

array ***Imagick::queryFonts*** *([string $pattern])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Imagick がサポートするフォントを返します。

パラメータ

pattern

クエリパターン。

返り値

フォントを含む配列を返します。 エラー時に ImagickException をスローします。

Imagick::queryFontMetrics

(No version information available, might be only in CVS)

Imagick::queryFontMetrics — フォントメトリクスを表す配列を返す

説明

array ***Imagick::queryFontMetrics*** *(ImagickDraw $properties , string $text [, bool $multiline])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

フォントメトリクスを表す、13 要素の配列を返します。

パラメータ

properties

フォントのプロパティを含む ImagickDraw。

text

テキスト。

multiline

マルチラインパラメータ。空にすると自動検出します。

返り値

Imagick がサポートするフォーマットを含む配列を返します。 エラー時に ImagickException をスローします。

ImagickDraw::affine

(No version information available, might be only in CVS)

ImagickDraw::affine — 現在のアフィン変換行列を設定する

説明

bool **ImagickDraw::affine** (array $affine)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在のアフィン変換行列を、指定したものに設定します。

パラメータ

affine

アフィン行列のパラメータ。

返り値

値を返しません。

ImagickDraw::annotation

(No version information available, might be only in CVS)

ImagickDraw::annotation — 画像上にテキストを描画する

説明

bool **ImagickDraw::annotation** (float $x , float $y , string $text)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像上にテキストを描画します。

パラメータ

x

テキストを描画する x 座標。

y

テキストを描画する y 座標。

text

画像上に描画するテキスト。

返り値

値を返しません。

ImagickDraw::arc

(No version information available, might be only in CVS)

ImagickDraw::arc — 円弧を描画する

説明

bool **ImagickDraw::arc** (float $sx , float $sy , float $ex , float $ey , float $sd , float $ed)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像上の指定した矩形内に円弧を描画します。

パラメータ

sx

矩形の開始位置の x 座標。

sy

矩形の開始位置の y 座標。

ex

矩形の終了位置の x 座標。

ey

矩形の終了位置の y 座標。

sd

開始位置の角度。

ed

終了位置の角度。

返り値

値を返しません。

ImagickDraw::bezier

(No version information available, might be only in CVS)

ImagickDraw::bezier — ベジエ曲線を描画する

説明

```
bool ImagickDraw::bezier ( array $coordinates )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像上に点を指定してベジエ曲線を描画します。

パラメータ

coordinates

多次元配列。array(array('x' => 1, 'y' => 2), array('x' => 3, 'y' => 4)) のような形式。

返り値

値を返しません。

ImagickDraw::circle

(No version information available, might be only in CVS)

ImagickDraw::circle — 円を描画する

説明

```
bool ImagickDraw::circle ( float $ox , float $oy , float $px , float $py )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像上に円を描画します。

パラメータ

ox

原点の x 座標。

oy

原点の y 座標。

px

周囲の x 座標。

py

周囲の y 座標。

返り値

値を返しません。

ImagickDraw::clear

(No version information available, might be only in CVS)

ImagickDraw::clear — ImagickDraw をクリアする

説明

bool **ImagickDraw::clear** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickDraw オブジェクト上のコマンドをすべてクリアし、 デフォルト設定で初期化します。

パラメータ

返り値

ImagickDraw オブジェクトを返します。

ImagickDraw::clone

(No version information available, might be only in CVS)

ImagickDraw::clone — 指定した ImagickDraw オブジェクトの完全なコピーを作成する

説明

ImagickDraw **ImagickDraw::clone** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した ImagickDraw オブジェクトの完全なコピーを作成します。

パラメータ

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

ImagickDraw::color

(No version information available, might be only in CVS)

ImagickDraw::color — 画像上に色を描画する

説明

bool **ImagickDraw::color** (float $x , float $y , int $paintMethod)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の塗りつぶし色を使用して、 指定した開始位置から指定した方法で画像上に色を描画します。

パラメータ

x

描画位置の x 座標。

y

描画位置の y 座標。

paintMethod

PAINT_ 定数のいずれか。

返り値

値を返しません。

ImagickDraw::comment

(No version information available, might be only in CVS)

ImagickDraw::comment — コメントを追加する

説明

bool **ImagickDraw::comment** (string $comment)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ベクター出力ストリームにコメントを追加します。

パラメータ

comment

ベクター出力ストリームに追加するコメント文字列。

返り値

値を返しません。

ImagickDraw::composite

(No version information available, might be only in CVS)

ImagickDraw::composite — 現在の画像上に別の画像を合成する

説明

bool **ImagickDraw::composite** (int $compose , float $x , float $y , float $width , float $height , Imagick $compositeWand)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の画像上に別の画像を合成します。 合成演算子、位置、サイズを指定します。

パラメータ

compose

合成演算子。COMPOSITE_ 定数のいずれか。

x

左上角の x 座標。

y

左上角の y 座標。

width

合成する画像の幅。

height

合成する画像の高さ。

compositeWand

合成する画像を取得する Imagick オブジェクト。

返り値

*成功した場合に **TRUE** を返します。*

ImagickDraw::__construct

(No version information available, might be only in CVS)

ImagickDraw::__construct — ImagickDraw コンストラクタ

説明

*ImagickDraw **ImagickDraw::__construct** (void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickDraw のコンストラクタです。

パラメータ

返り値

値を返しません。

ImagickDraw::destroy

(No version information available, might be only in CVS)

ImagickDraw::destroy — 関連付けられたすべてのリソースを開放する

説明

*bool **ImagickDraw::destroy** (void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この ImagickDraw オブジェクトに関連付けられたすべてのリソースを開放します。

パラメータ

返り値

値を返しません。

ImagickDraw::ellipse

(No version information available, might be only in CVS)

ImagickDraw::ellipse — 画像上に楕円を描画する

説明

bool **ImagickDraw::ellipse** (float $ox , float $oy , float $rx , float $ry , float $start , float $end)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像上に楕円を描画します。

パラメータ

ox

oy

rx

ry

start

end

返り値

値を返しません。

ImagickDraw::getClipPath

(No version information available, might be only in CVS)

ImagickDraw::getClipPath — 現在のクリッピングパスの ID を取得する

説明

string **ImagickDraw::getClipPath** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在のクリッピングパスの ID を取得します。

パラメータ

返り値

クリッピングパスの ID を含む文字列、あるいはパスが存在しない場合に false を返します。

ImagickDraw::getClipRule

(No version information available, might be only in CVS)

ImagickDraw::getClipRule — 現在の多角形塗りつぶしルールを返す

説明

int **ImagickDraw::getClipRule** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クリッピングパスで使用されている、現在の多角形塗りつぶしルールを返します。

パラメータ

返り値

FILLRULE_ 定数のいずれかを返します。

ImagickDraw::getClipUnits

(No version information available, might be only in CVS)

ImagickDraw::getClipUnits — *クリップパスの単位の解釈を返す*

説明

int ***ImagickDraw::getClipUnits*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クリップパスの単位の解釈を返します。

パラメータ

返り値

成功した場合に整数値を返します。

ImagickDraw::getFillColor

(No version information available, might be only in CVS)

ImagickDraw::getFillColor — *塗りつぶし色を返す*

説明

ImagickPixel ***ImagickDraw::getFillColor*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

描画オブジェクトで使用する塗りつぶし色を返します。

パラメータ

返り値

ImagickPixel オブジェクトを返します。

ImagickDraw::getFillOpacity

(No version information available, might be only in CVS)

ImagickDraw::getFillOpacity — *描画時の透過度を返す*

説明

float ***ImagickDraw::getFillOpacity*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

色やテクスチャによる塗りつぶしの際の透過度を返します。 完全に不透過の場合は 1.0 となります。

パラメータ

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

ImagickDraw::getFillRule

(No version information available, might be only in CVS)

ImagickDraw::getFillRule — *塗りつぶしルールを返す*

説明

int **ImagickDraw::getFillRule** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

多角形の描画時に使用する塗りつぶしルールを返します。

パラメータ

返り値

FILLRULE_ 定数を返します。

ImagickDraw::getFontFamily

(No version information available, might be only in CVS)

ImagickDraw::getFontFamily — フォントファミリーを返す

説明

string **ImagickDraw::getFontFamily** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントファミリーを返します。

パラメータ

返り値

現在選択されているフォントファミリー、 あるいはフォントファミリーが設定されていない場合に false を返します。

ImagickDraw::getFontSize

(No version information available, might be only in CVS)

ImagickDraw::getFontSize — フォントのポイント数を返す

説明

float **ImagickDraw::getFontSize** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントのポイント数を返します。

パラメータ

返り値

現在の ImagickDraw オブジェクトに関連付けられているフォントのサイズを返します。

ImagickDraw::getFontStyle

(No version information available, might be only in CVS)

ImagickDraw::getFontStyle — フォントのスタイルを返す

説明

int **ImagickDraw::getFontStyle** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントのスタイルを返します。

パラメータ

返り値

ImagickDraw オブジェクトに関連付けられたフォントのスタイルを表す定数 (STYLE_)、 あるいはスタイルが設定されていない場合に 0 を返します。

ImagickDraw::getFontWeight

(No version information available, might be only in CVS)

ImagickDraw::getFontWeight — フォントの重さを返す

説明

int **ImagickDraw::getFontWeight** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントの重さを返します。

パラメータ

返り値

成功した場合に整数値、重さが設定されていない場合に 0 を返します。

ImagickDraw::getFont

(No version information available, might be only in CVS)

ImagickDraw::getFont — フォントを返す

説明

string **ImagickDraw::getFont** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントを文字列で返します。

パラメータ

返り値

成功した場合に文字列、フォントが設定されていない場合に false を返します。

ImagickDraw::getGravity

(No version information available, might be only in CVS)

ImagickDraw::getGravity — テキストの配置時の gravity を返す

説明

int **ImagickDraw::getGravity** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するテキスト配置の gravity を返します。

パラメータ

返り値

成功した場合に GRAVITY_ 定数、gravity が設定されていない場合に 0 を返します。

ImagickDraw::getStrokeAntialias

(No version information available, might be only in CVS)

ImagickDraw::getStrokeAntialias — 現在の縁取りのアンチエイリアス設定を返す

説明

bool **ImagickDraw::getStrokeAntialias** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の縁取りのアンチエイリアス設定を返します。 縁取りの枠線は、デフォルトでアンチエイリアス処理が行われます。 アンチエイリアスを無効にすると、 縁取りの描画時には縁取り色あるいはキャンバスの色のどちらかが用いられることになります。

パラメータ

返り値

アンチエイリアスがオンの場合に true、オフの場合に false を返します。

ImagickDraw::getStrokeColor

(No version information available, might be only in CVS)

ImagickDraw::getStrokeColor — オブジェクトの縁取りに使用する色を返す

説明

ImagickPixel **ImagickDraw::getStrokeColor** (ImagickPixel $stroke_color)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの縁取りに使用する色を返します。

パラメータ

stroke_color

返り値

色を表す ImagickPixel オブジェクトを返します。

ImagickDraw::getStrokeDashArray

(No version information available, might be only in CVS)

ImagickDraw::getStrokeDashArray — パスの描画に使用する破線のパターンを表す配列を返す

説明

array **ImagickDraw::getStrokeDashArray** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パスの描画に使用する破線のパターンを表す配列を返します。

パラメータ

返り値

成功した場合に配列、設定されていない場合は空の配列を返します。

ImagickDraw::getStrokeDashOffset

(No version information available, might be only in CVS)

ImagickDraw::getStrokeDashOffset — 破線パターンにおける破線の開始オフセットを返す

説明

float **ImagickDraw::getStrokeDashOffset** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

破線パターンにおける破線の開始オフセットを返します。

パラメータ

返り値

オフセットを表す float 値、あるいは設定されていない場合に 0 を返します。

ImagickDraw::getStrokeLineCap

(No version information available, might be only in CVS)

ImagickDraw::getStrokeLineCap — 開かれたサブパスを描画する際に使用する端点の形状を返す

説明

int **ImagickDraw::getStrokeLineCap** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

開かれたサブパスを描画する際に使用する端点の形状を返します。

パラメータ

返り値

LINECAP_ 定数のいずれか、あるいは設定されていない場合に 0 を返します。

ImagickDraw::getStrokeLineJoin

(No version information available, might be only in CVS)

ImagickDraw::getStrokeLineJoin — パスの角を描画する際に使用する形状を返す

説明

int **ImagickDraw::getStrokeLineJoin** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パス（あるいはその他のベクター図形）の角を描画する際に使用する形状を返します。

パラメータ

返り値

LINEJOIN_ 定数のいずれか、あるいは設定されていない場合に 0 を返します。

ImagickDraw::getStrokeMiterLimit

(No version information available, might be only in CVS)

ImagickDraw::getStrokeMiterLimit — マイターリミットを返す

説明

int **ImagickDraw::getStrokeMiterLimit** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

マイターリミットを返します。 2 本の直線が小さい角度で連結され、かつ連結方法が 'lineJoin' に設定されている場合、 出来上がる線の角の部分が非常に長くなります。マイターリミットは、miter length (角の長さ) と 'lineWidth' の比率の最大値です。

パラメータ

返り値

マイターリミットを表す整数値、あるいは マイターリミットが設定されていない場合に 0 を返します。

ImagickDraw::getStrokeOpacity

(No version information available, might be only in CVS)

ImagickDraw::getStrokeOpacity — オブジェクトの枠線の透明度を返す

説明

float **ImagickDraw::getStrokeOpacity** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの枠線の透明度を返します。

パラメータ

返り値

透明度を表す double 値を返します。

ImagickDraw::getStrokeWidth

(No version information available, might be only in CVS)

ImagickDraw::getStrokeWidth — オブジェクトの枠線の描画に使用する線の幅を返す

説明

float **ImagickDraw::getStrokeWidth** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの枠線の描画に使用する線の幅を返します。

パラメータ

返り値

線幅を表す double 値を返します。

ImagickDraw::getTextAlignment

(No version information available, might be only in CVS)

ImagickDraw::getTextAlignment — テキストの配置を返す

説明

int **ImagickDraw::getTextAlignment** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際の配置を返します。

パラメータ

返り値

ALIGN_ 定数のいずれか、あるいは設定されていない場合に 0 を返します。

ImagickDraw::getTextAntialias

(No version information available, might be only in CVS)

ImagickDraw::getTextAntialias — 現在のテキストのアンチエイリアス設定を返す

説明

bool **ImagickDraw::getTextAntialias** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在のテキストのアンチエイリアス設定を返します。 これは、テキストのアンチエイリアス処理を行うかどうかを表すものです。 デフォルトでは、テキストのアンチエイリアス処理が行われます。

パラメータ

返り値

テキストをアンチエイリアス処理する場合に true、しない場合に false を返します。

ImagickDraw::getTextDecoration

(No version information available, might be only in CVS)

ImagickDraw::getTextDecoration — テキストの装飾を返す

説明

int **ImagickDraw::getTextDecoration** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に適用する装飾を返します。

パラメータ

返り値

DECORATION_ 定数のいずれか、あるいは装飾が設定されていない場合に 0 を返します。

ImagickDraw::getTextEncoding

(No version information available, might be only in CVS)

ImagickDraw::getTextEncoding — テキストによる注記の際に使用するコードセットを返す

説明

string **ImagickDraw::getTextEncoding** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するコードセットを表す文字列を返します。

パラメータ

返り値

コードセットを表す文字列、 あるいはエンコーディングが設定されていない場合に false を返します。

ImagickDraw::getTextUnderColor

(No version information available, might be only in CVS)

ImagickDraw::getTextUnderColor — テキストの背景色を返す

説明

ImagickPixel **ImagickDraw::getTextUnderColor** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記の背景に置く矩形の色を返します。

返り値

色を表す ImagickPixel オブジェクトを返します。

ImagickDraw::getVectorGraphics

(No version information available, might be only in CVS)

ImagickDraw::getVectorGraphics — ベクターグラフィックを含む文字列を返す

説明

string **ImagickDraw::getVectorGraphics** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickDraw オブジェクトのインスタンスが作成された後で、 グラフィックメソッドのコールによってできあがったベクターグラフィックを文字列で返します。

パラメータ

返り値

ベクターグラフィックを文字列で返します。

ImagickDraw::line

(No version information available, might be only in CVS)

ImagickDraw::line — 直線を描画する

説明

bool **ImagickDraw::line** (float $sx , float $sy , float $ex , float $ey)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の描画色、透明度そして線幅を使用して画像上に直線を描画します。

パラメータ

sx

開始位置の x 座標。

sy

開始位置の y 座標。

ex

終了位置の x 座標。

ey

終了位置の y 座標。

返り値

値を返しません。

ImagickDraw::matte

(No version information available, might be only in CVS)

ImagickDraw::matte — 画像の opacity チャネル上に描画する

説明

bool **ImagickDraw::matte** (float $x , float $y , int $paintMethod)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画像の opacity チャネル上に描画し、ピクセルの透過度に影響を与えます。

パラメータ

x

マットの x 座標。

y

マットの y 座標。

paintMethod

PAINT_ 定数。

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

ImagickDraw::pathClose

(No version information available, might be only in CVS)

ImagickDraw::pathClose — パス要素を現在のパスに追加する

説明

bool **ImagickDraw::pathClose** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パス要素を現在のパスに追加し、 現在の点から現在のサブパスの直近の開始点（通常は、直近の moveto の点となります）に直線を引くことで現在のパスを閉じます。

パラメータ

返り値

値を返しません。

ImagickDraw::pathCurveToAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToAbsolute — 三次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToAbsolute*** *(float $x1 , float $y1 , float $x2 , float $y2 , float $x , float $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの三次ベジエ曲線を描画します。 曲線の開始位置の制御点を (x1,y1)、終了位置の制御点を (x2,y2) とし、それぞれ絶対座標で指定します。このコマンドが終了した後は、 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x1

最初の制御点の x 座標。

y1

最初の制御点の y 座標。

x2

2 番目の制御点の x 座標。

y2

2 番目の制御点の y 座標。

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathCurveToQuadraticBezierAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToQuadraticBezierAbsolute — 二次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToQuadraticBezierAbsolute*** *(float $x1 , float $y1 , float $x , float $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの二次ベジエ曲線を描画します。 制御点を (x1,y1) とし、絶対座標で指定します。このコマンドが終了した後は、 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x1

制御点の x 座標。

y1

制御点の y 座標。

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathCurveToQuadraticBezierRelative

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToQuadraticBezierRelative — 二次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToQuadraticBezierRelative*** *(float $x1 , float $y1 , float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの二次ベジエ曲線を描画します。 制御点を (x1,y1) とし、相対座標で指定します。このコマンドが終了した後は、 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x1

始点の x 座標。

y1

始点の y 座標。

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathCurveToQuadraticBezierSmoothAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToQuadraticBezierSmoothAbsolute — 二次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToQuadraticBezierSmoothAbsolute*** *(float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの二次ベジエ曲線（相対座標）を描画します。 制御点の位置は、現在の点と直前のコマンドの制御点の位置から推測します (直線のコマンドがなかったり、直前のコマンドが DrawPathCurveToQuadraticBezierAbsolute, DrawPathCurveToQuadraticBezierRelative, DrawPathCurveToQuadraticBezierSmoothAbsolut あるいは DrawPathCurveToQuadraticBezierSmoothRelative のいずれかでなかった場合は、現在の点を制御点とみなします)。 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathCurveToQuadraticBezierSmoothRelative

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToQuadraticBezierSmoothRelative — 二次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToQuadraticBezierSmoothRelative*** *(float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの二次ベジエ曲線 (相対座標) を描画します。 制御点の位置は、現在の点と直前のコマンドの制御点の位置から推測します (直線のコマンドがなかったり、直前のコマンドが DrawPathCurveToQuadraticBezierAbsolute, DrawPathCurveToQuadraticBezierRelative, DrawPathCurveToQuadraticBezierSmoothAbsolut あるいは DrawPathCurveToQuadraticBezierSmoothRelative のいずれかでなかった場合は、現在の点を制御点とみなします)。 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathCurveToRelative

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToRelative — 三次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToRelative*** *(float $x1 , float $y1 , float $x2 , float $y2 , float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの三次ベジエ曲線を描画します。 曲線の開始位置の制御点を (x1,y1)、終了位置の制御点を (x2,y2) とし、それぞれ相対座標で指定します。このコマンドが終了した後は、 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x1

開始制御点の x 座標。

y1

開始制御点の y 座標。

x2

終了制御点の x 座標。

y2

終了制御点の y 座標。

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathCurveToSmoothAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToSmoothAbsolute — 三次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToSmoothAbsolute*** *(float $x2 , float $y2 , float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの三次ベジエ曲線 (絶対座標) を描画します。 最初の制御点の位置は、現在の点と直前のコマンドの 2 番目の制御点の位置から推測します (直線のコマンドがなかったり、直前のコマンドが DrawPathCurveToAbsolute, DrawPathCurveToRelative, DrawPathCurveToSmoothAbsolute あるいは DrawPathCurveToSmoothRelative のいずれかでなかった場合は、現在の点を最初の制御点とみなします)。 2 番目の制御点 (曲線の終了地点の制御点) は (x2,y2) となります。 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x2

2 番目の制御点の x 座標。

y2

2 番目の制御点の y 座標。

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathCurveToSmoothRelative

(No version information available, might be only in CVS)

ImagickDraw::pathCurveToSmoothRelative — 三次ベジエ曲線を描画する

説明

bool ***ImagickDraw::pathCurveToSmoothRelative*** *(float $x2 , float $y2 , float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から (x,y) までの三次ベジエ曲線 (相対座標) を描画します。 最初の制御点の位置は、現在の点と直前のコマンドの 2 番目の制御点の位置から推測します (直線のコマンドがなかったり、直前のコマンドが DrawPathCurveToAbsolute, DrawPathCurveToRelative, DrawPathCurveToSmoothAbsolute あるいは DrawPathCurveToSmoothRelative のいずれかでなかった場合は、現在の点を最初の制御点とみなします)。 2 番目の制御点 (曲線の終了地点の制御点) は (x2,y2) となります。 現在の位置は polybezier が使用する最後の (x,y) 座標の組となります。

パラメータ

x2

2 番目の制御点の x 座標。

y2

2 番目の制御点の y 座標。

x

終点の x 座標。

y

終点の y 座標。

返り値

値を返しません。

ImagickDraw::pathEllipticArcAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathEllipticArcAbsolute — 楕円弧を描画する

説明

bool ***ImagickDraw::pathEllipticArcAbsolute*** *(float $rx , float $ry , float $x_axis_rotation , bool $large_arc_flag , bool $sweep_flag , float $x , float $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の位置から (x, y) まで、絶対座標を使用して楕円弧を描画します。 楕円の大きさと方向は、2 つの半径 (rx, ry) と xAxisRotation で決まります。xAxisRotation は、楕円全体を座標系に対してどれだけ回転させるかを表すものです。 楕円の中心 (cx, cy) は、その他のパラメータの値から自動的に算出されます。 largeArcFlag および sweepFlag は自動計算を支援するためのもので、 楕円の描画方法を定義します。 largeArcFlag が true の場合は考えうる楕円のうち大きいほうを描画します。 sweepFlag が true の場合は時計回りの回転で楕円を描画します。

パラメータ

rx

x 方向の半径。

ry

y 方向の半径。

x_axis_rotation

x 軸の回転。

large_arc_flag

large arc フラグ。

sweep_flag

sweep フラグ。

x

x 座標。

y

y 座標。

返り値

値を返しません。

ImagickDraw::pathEllipticArcRelative

(No version information available, might be only in CVS)

ImagickDraw::pathEllipticArcRelative — 楕円弧を描画する

説明

bool ***ImagickDraw::pathEllipticArcRelative*** *(float $rx , float $ry , float $x_axis_rotation , bool $large_arc_flag , bool $sweep_flag , float $x , float $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の位置から (x, y) まで、相対座標を使用して楕円弧を描画します。 楕円の大きさと方向は、2 つの半径 (rx, ry) と xAxisRotation で決まります。xAxisRotation は、楕円全体を座標系に対してどれだけ回転させるかを表すものです。 楕円の中心 (cx, cy) は、その他のパラメータの値から自動的に算出されます。 largeArcFlag および sweepFlag は自動計算を支援するためのもので、 楕円の描画方法を定義します。 largeArcFlag が true の場合は考えうる楕円のうち大きいほうを描画します。 sweepFlag が true の場合は時計回りの回転で楕円を描画します。

パラメータ

rx

x 方向の半径。

ry

y 方向の半径。

x_axis_rotation

x 軸の回転。

large_arc_flag

large arc フラグ。

sweep_flag

sweep フラグ。

x

x 座標。

y

y 座標。

返り値

値を返しません。

ImagickDraw::pathFinish

(No version information available, might be only in CVS)

ImagickDraw::pathFinish — 現在のパスを終了する

説明

bool ***ImagickDraw::pathFinish*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在のパスを終了します。

パラメータ

返り値

値を返しません。

ImagickDraw::pathLineToAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathLineToAbsolute — 直線パスを描画する

説明

bool ***ImagickDraw::pathLineToAbsolute*** *(float $x , float $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から指定した座標まで、絶対座標を使用して直線を描画します。 座標は新しい点に移動します。

パラメータ

x

始点の x 座標。

y

終点の x 座標。

返り値

値を返しません。

ImagickDraw::pathLineToHorizontalAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathLineToHorizontalAbsolute — 水平直線パスを描画する

説明

bool **ImagickDraw::pathLineToHorizontalAbsolute** (float $x)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から指定した座標まで、絶対座標を使用して水平直線を描画します。 座標は新しい点に移動します。

パラメータ

x

x 座標。

返り値

値を返しません。

ImagickDraw::pathLineToHorizontalRelative

(No version information available, might be only in CVS)

ImagickDraw::pathLineToHorizontalRelative — 水平直線パスを描画する

説明

bool **ImagickDraw::pathLineToHorizontalRelative** (float $x)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から指定した座標まで、相対座標を使用して水平直線を描画します。 座標は新しい点に移動します。

パラメータ

x

x 座標。

返り値

値を返しません。

ImagickDraw::pathLineToRelative

(No version information available, might be only in CVS)

ImagickDraw::pathLineToRelative — 直線パスを描画する

説明

bool **ImagickDraw::pathLineToRelative** (*float $x , float $y*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から指定した座標まで、相対座標を使用して直線を描画します。 座標は新しい点に移動します。

パラメータ

x

始点の x 座標。

y

始点の y 座標。

返り値

値を返しません。

ImagickDraw::pathLineToVerticalAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathLineToVerticalAbsolute — 垂直直線パスを描画する

説明

bool **ImagickDraw::pathLineToVerticalAbsolute** (*float $y*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から指定した座標まで、絶対座標を使用して垂直直線を描画します。 座標は新しい点に移動します。

パラメータ

y

y 座標。

返り値

値を返しません。

ImagickDraw::pathLineToVerticalRelative

(No version information available, might be only in CVS)

ImagickDraw::pathLineToVerticalRelative — 垂直直線パスを描画する

説明

bool **ImagickDraw::pathLineToVerticalRelative** (*float $y*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の点から指定した座標まで、相対座標を使用して垂直直線を描画します。 座標は新しい点に移動します。

パラメータ

y

y 座標。

返り値

値を返しません。

ImagickDraw::pathMoveToAbsolute

(No version information available, might be only in CVS)

ImagickDraw::pathMoveToAbsolute — 新しいサブパスを開始する

説明

bool ***ImagickDraw::pathMoveToAbsolute*** *(float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

絶対座標で指定した座標から、新しいサブパスを開始します。 現在の点は指定した座標となります。

パラメータ

x

始点の x 座標。

y

始点の y 座標。

返り値

値を返しません。

ImagickDraw::pathMoveToRelative

(No version information available, might be only in CVS)

ImagickDraw::pathMoveToRelative — 新しいサブパスを開始する

説明

bool ***ImagickDraw::pathMoveToRelative*** *(float $x , float $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

相対座標で指定した座標から、新しいサブパスを開始します。 現在の点は指定した座標となります。

パラメータ

x

始点の x 座標。

y

始点の y 座標。

返り値

値を返しません。

ImagickDraw::pathStart

(No version information available, might be only in CVS)

ImagickDraw::pathStart — パス描画リストの開始を宣言する

説明

bool ***ImagickDraw::pathStart*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パス描画リストの開始を宣言します。このリストは、対応する DrawPathFinish() コマンドで終了します。 すべての DrawPath コマンドは、このコマンドと DrawPathFinish() コマンドの間にある必要があります。パス描画コマンドは従属コマンドであり、 それ自身だけでは機能しないからです。

パラメータ

返り値

値を返しません。

ImagickDraw::point

(No version information available, might be only in CVS)

ImagickDraw::point — 点を描画する

説明

bool **ImagickDraw::point** *(float $x , float $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の描画色と濃度を使用して、 指定した座標に点を描画します。

パラメータ

x

点の x 座標。

y

点の y 座標。

返り値

値を返しません。

ImagickDraw::polygon

(No version information available, might be only in CVS)

ImagickDraw::polygon — 多角形を描画する

説明

bool **ImagickDraw::polygon** *(array $coordinates)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の線と線幅、そして塗りつぶし色あるいはテクスチャを使用して、 指定した座標配列から多角形を描画します。

パラメータ

coordinates

多次元配列。 array(array('x' => 3, 'y' => 4), array('x' => 2, 'y' => 6)); のような形式となります。

返り値

*成功した場合に **TRUE** を返します。*

ImagickDraw::polyline

(No version information available, might be only in CVS)

ImagickDraw::polyline — 線分群を描画する

説明

bool **ImagickDraw::polyline** (array $coordinates)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の線と線幅、そして塗りつぶし色あるいはテクスチャを使用して、 指定した座標配列から線分群を描画します。

パラメータ

coordinates

x 座標および y 座標の配列。 array(array('x' => 4, 'y' => 6), array('x' => 8, 'y' => 10)) のような形式となります。

返り値

成功した場合に **TRUE** を返します。

ImagickDraw::popClipPath

(No version information available, might be only in CVS)

ImagickDraw::popClipPath — クリップパスの定義を終了する

説明

bool **ImagickDraw::popClipPath** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クリップパスの定義を終了します。

パラメータ

返り値

値を返しません。

ImagickDraw::popDefs

(No version information available, might be only in CVS)

ImagickDraw::popDefs — 定義リストを終了する

説明

bool **ImagickDraw::popDefs** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

定義リストを終了します。

パラメータ

返り値

値を返しません。

ImagickDraw::pop

(No version information available, might be only in CVS)

ImagickDraw::pop — スタック内の現在の ImagickDraw を破棄し、事前に push された ImagickDraw を返す

説明

bool **ImagickDraw::pop** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

スタック内の現在の ImagickDraw を破棄し、事前に push された ImagickDraw を返します。 複数の ImagickDraw が存在する可能性もあります。 push された回数より多くの ImagickDraw の pop を試みるとエラーとなります。 また、ImagickDrow が push されたときと同じ方式で pop する必要があります。

パラメータ

返り値

成功した場合に true、失敗した場合に false を返します。

ImagickDraw::popPattern

(No version information available, might be only in CVS)

ImagickDraw::popPattern — パターン定義を終了する

説明

bool **ImagickDraw::popPattern** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パターン定義を終了します。

パラメータ

返り値

成功した場合に true、失敗した場合に false を返します。

ImagickDraw::pushClipPath

(No version information available, might be only in CVS)

ImagickDraw::pushClipPath — クリップパスの定義を開始する

説明

bool **ImagickDraw::pushClipPath** (string $clip_mask_id)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クリップパスの定義を開始します。 これは、任意の数の描画コマンドで構成され、最後は ImagickDraw::popClipPath() コマンドとなります。

パラメータ

clip_mask_id

クリップマスク ID。

返り値

値を返しません。

ImagickDraw::pushDefs

(No version information available, might be only in CVS)

ImagickDraw::pushDefs — 後に続くコマンドが、処理の前に名前つき要素を作成することを示す

説明

bool **ImagickDraw::pushDefs** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

後に続く ImagickDraw::popDefs() までのコマンドが、 名前つき要素 (クリップパスやテクスチャなど) を作成することを示します。 これは、効率を考慮して事前に処理されます。

パラメータ

返り値

値を返しません。

ImagickDraw::push

(No version information available, might be only in CVS)

ImagickDraw::push — 現在の ImagickDraw をコピーしてスタックに格納する

説明

bool **ImagickDraw::push** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の ImagickDraw をコピーして新しい ImagickDraw を作成し、 それを ImagickDraw スタックに格納します。格納した ImagickDraw は pop() で取り出すことができます。ImagickDraw は ImagickDraw スタックに保存されます。Pop を行うには、事前に Push が行われている必要があります。

パラメータ

返り値

成功した場合に true、失敗した場合に false を返します。

ImagickDraw::pushPattern

(No version information available, might be only in CVS)

ImagickDraw::pushPattern — 後に続く ImagickDraw::opPattern() までのコマンドが、名前付きパターンを構成することを示す

説明

bool **ImagickDraw::pushPattern** (string $pattern_id , float $x , float $y , float $width , float $height)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

後に続く DrawPopPattern() までのコマンドが、 名前付きパターンを構成することを示します。 パターン空間は、左上の座標と幅、高さで指定します。 そして独自の描画空間となります。 描画される内容は、すべてこのパターン定義を使用します。 名前付きパターンは、線やブラシの定義で使用することができます。

パラメータ

pattern_id

パターン ID。

x

左上の角の x 座標。

y

左上の角の y 座標。

width

パターンの幅。

height

パターンの高さ。

返り値

成功した場合に true、失敗した場合に false を返します。

ImagickDraw::rectangle

(No version information available, might be only in CVS)

ImagickDraw::rectangle — 矩形を描画する

説明

bool **ImagickDraw::rectangle** (float $x1 , float $y1 , float $x2 , float $y2)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

2 つの座標を与え、現在の枠線や線幅、塗りつぶしの設定を使用して矩形を描画します。

パラメータ

x1

左上の角の x 座標。

y1

左上の角の y 座標。

x2

右下の角の x 座標。

y2

右下の角の y 座標。

返り値

値を返しません。

ImagickDraw::render

(No version information available, might be only in CVS)

ImagickDraw::render — これまでのすべての描画コマンドを画像上にレンダリングする

説明

bool **ImagickDraw::render** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

これまでのすべての描画コマンドを画像上にレンダリングします。

パラメータ

返り値

成功した場合に true、失敗した場合に false を返します。

ImagickDraw::rotate

(No version information available, might be only in CVS)

ImagickDraw::rotate — 指定した回転を現在の座標空間に適用する

説明

bool **ImagickDraw::rotate** (float $degrees)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した回転を現在の座標空間に適用します。

パラメータ

degrees

回転角度。

返り値

値を返しません。

ImagickDraw::roundRectangle

(No version information available, might be only in CVS)

ImagickDraw::roundRectangle — 角が丸い矩形を描画する

説明

bool **ImagickDraw::roundRectangle** (float $x1 , float $y1 , float $x2 , float $y2 , float $rx , float $ry)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

2 つの座標、そして角の x & y 方向の丸めを与え、 現在の枠線や線幅、塗りつぶしの設定を使用して角が丸い矩形を描画します。

パラメータ

x1

左上の角の x 座標。

y1

左上の角の y 座標。

x2

右下の角の x 座標。

y2

右下の角の y 座標。

rx

x 方向の丸め。

ry

y 方向の丸め。

返り値

値を返しません。

ImagickDraw::scale

(No version information available, might be only in CVS)

ImagickDraw::scale — 倍率を調整する

説明

bool **ImagickDraw::scale** (float $x , float $y)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の座標空間における、 水平方向および垂直方向の拡大率を調整します。

パラメータ

x

水平方向の拡大率。

y

垂直方向の拡大率。

返り値

値を返しません。

ImagickDraw::setClipPath

(No version information available, might be only in CVS)

ImagickDraw::setClipPath — 指定した名前のクリッピングパスを画像に関連付ける

説明

bool ***ImagickDraw::setClipPath*** *(string $clip_mask)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した名前のクリッピングパスを画像に関連付けます。 クリッピングパスの範囲内に描画された内容だけが反映されます。

パラメータ

clip_mask

クリッピングパスの名前。

返り値

値を返しません。

ImagickDraw::setClipRule

(No version information available, might be only in CVS)

ImagickDraw::setClipRule — クリッピングパスで使用する多角形塗りつぶしルールを設定する

説明

bool ***ImagickDraw::setClipRule*** *(int $fill_rule)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クリッピングパスで使用する多角形塗りつぶしルールを設定します。

パラメータ

fill_rule

FILLRULE_ 定数。

返り値

値を返しません。

ImagickDraw::setClipUnits

(No version information available, might be only in CVS)

ImagickDraw::setClipUnits — *クリップパスの単位の解釈を設定する*

説明

bool ***ImagickDraw::setClipUnits*** *(int $clip_units)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クリップパスの単位の解釈を設定します。

パラメータ

clip_units

クリップ単位の数。

返り値

値を返しません。

ImagickDraw::setFillAlpha

(No version information available, might be only in CVS)

ImagickDraw::setFillAlpha — *色やテクスチャによる塗りつぶしの際の透過度を設定する*

説明

bool ***ImagickDraw::setFillAlpha*** *(float $opacity)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

色やテクスチャによる塗りつぶしの際の透過度を設定します。 完全に不透明な状態が 1.0 となります。

パラメータ

opacity

塗りつぶしのアルファ値。

返り値

値を返しません。

ImagickDraw::setFillColor

(No version information available, might be only in CVS)

ImagickDraw::setFillColor — *オブジェクトの塗りつぶしに使用する色を設定する*

説明

bool ***ImagickDraw::setFillColor*** *(ImagickPixel $fill_pixel)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの塗りつぶしに使用する色を設定します。

パラメータ

fill_pixel

色を表す ImagickPixel。

返り値

値を返しません。

ImagickDraw::setFillOpacity

(No version information available, might be only in CVS)

ImagickDraw::setFillOpacity — 色やテクスチャによる塗りつぶしの際の透過度を設定する

説明

bool **ImagickDraw::setFillOpacity** (float $fillOpacity)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

色やテクスチャによる塗りつぶしの際の透過度を設定します。 完全に不透明な状態が 1.0 となります。

パラメータ

fillOpacity

塗りつぶしの透明度。

返り値

値を返しません。

ImagickDraw::setFillPatternURL

(No version information available, might be only in CVS)

ImagickDraw::setFillPatternURL — オブジェクトの塗りつぶしパターンとして使用する URL を設定する

説明

bool **ImagickDraw::setFillPatternURL** (string $fill_url)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの塗りつぶしパターンとして使用する URL を設定します。 現時点でサポートしているのは、ローカル URL ("#identifier") のみです。 このローカル URL は、DrawPushPattern/DrawPopPattern で塗りつぶしパターンを定義することによって作成されます。

パラメータ

fill_url

塗りつぶしパターンを取得する URL。

返り値

成功した場合に true、失敗した場合に false を返します。

ImagickDraw::setFillRule

(No version information available, might be only in CVS)

ImagickDraw::setFillRule — 多角形の描画時に使用する塗りつぶしルールを設定する

説明

bool **ImagickDraw::setFillRule** (int $fill_rule)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

多角形の描画時に使用する塗りつぶしルールを設定します。

パラメータ

fill_rule

FILLRULE_ 定数。

返り値

値を返しません。

ImagickDraw::setFontFamily

(No version information available, might be only in CVS)

ImagickDraw::setFontFamily — テキストによる注記を行う際に使用するフォントファミリーを設定する

説明

bool ***ImagickDraw::setFontFamily*** *(string $font_family)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントファミリーを設定します。

パラメータ

font_family

フォントファミリー。

返り値

成功した場合に ***TRUE*** *を返します。*

ImagickDraw::setFontSize

(No version information available, might be only in CVS)

ImagickDraw::setFontSize — テキストによる注記を行う際に使用するフォントのポイント数を設定する

説明

bool ***ImagickDraw::setFontSize*** *(float $pointsize)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントのポイント数を設定します。

パラメータ

pointsize

ポイント数。

返り値

値を返しません。

ImagickDraw::setFontStretch

(No version information available, might be only in CVS)

ImagickDraw::setFontStretch — テキストによる注記を行う際に使用するフォントの伸縮を設定する

説明

bool ***ImagickDraw::setFontStretch*** *(int $fontStretch)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントの伸縮を設定します。 AnyStretch は、ワイルドカードの "don't care" オプションとして働きます。

パラメータ

fontStretch

STRETCH_ 定数。

返り値

値を返しません。

ImagickDraw::setFontStyle

(No version information available, might be only in CVS)

ImagickDraw::setFontStyle — テキストによる注記を行う際に使用するフォントのスタイルを設定する

説明

bool ***ImagickDraw::setFontStyle*** *(int $style)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントのスタイルを設定します。 AnyStyle は、ワイルドカードの "don't care" オプションとして働きます。

パラメータ

style

STYLETYPE_ 定数。

返り値

値を返しません。

ImagickDraw::setFontWeight

(No version information available, might be only in CVS)

ImagickDraw::setFontWeight — フォントの重さを設定する

説明

bool ***ImagickDraw::setFontWeight*** *(int $font_weight)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントの重さを設定します。

パラメータ

font_weight

返り値

ImagickDraw::setFont

(No version information available, might be only in CVS)

ImagickDraw::setFont — テキストによる注記を行う際に使用するフォントを設定する

説明

bool ***ImagickDraw::setFont*** *(string $font_name)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するフォントを設定します。

パラメータ

font_name

返り値

成功した場合に ***TRUE*** *を返します。*

ImagickDraw::setGravity

(No version information available, might be only in CVS)

ImagickDraw::setGravity — テキストの配置時の gravity を設定する

説明

bool ***ImagickDraw::setGravity*** *(int $gravity)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するテキスト配置の gravity を設定します。

パラメータ

gravity

GRAVITY_ 定数。

返り値

値を返しません。

ImagickDraw::setStrokeAlpha

(No version information available, might be only in CVS)

ImagickDraw::setStrokeAlpha — オブジェクトの枠線の透明度を指定する

説明

bool ***ImagickDraw::setStrokeAlpha*** *(float $opacity)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの枠線の透明度を指定します。

パラメータ

opacity

透明度。

返り値

値を返しません。

ImagickDraw::setStrokeAntialias

(No version information available, might be only in CVS)

ImagickDraw::setStrokeAntialias — 縁取りの枠線をアンチエイリアス処理するかどうかを制御する

説明

bool **ImagickDraw::setStrokeAntialias** (bool $stroke_antialias)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

縁取りの枠線をアンチエイリアス処理するかどうかを制御します。 縁取りの枠線は、デフォルトでアンチエイリアス処理が行われます。 アンチエイリアスを無効にすると、 縁取りの描画時には縁取り色あるいはキャンバスの色のどちらかが用いられることになります。

パラメータ

stroke_antialias

アンチエイリアス設定。

返り値

値を返しません。

ImagickDraw::setStrokeColor

(No version information available, might be only in CVS)

ImagickDraw::setStrokeColor — オブジェクトの縁取りに使用する色を設定する

説明

bool **ImagickDraw::setStrokeColor** (ImagickPixel $stroke_pixel)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの縁取りに使用する色を設定します。

パラメータ

stroke_pixel

縁取り色。

返り値

値を返しません。

ImagickDraw::setStrokeDashArray

(No version information available, might be only in CVS)

ImagickDraw::setStrokeDashArray — パスの描画に使用する破線のパターンを指定する

説明

bool **ImagickDraw::setStrokeDashArray** (array $dashArray)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パスの描画に使用する破線のパターンを指定します。 strokeDashArray は数値の配列で、 互い違いに並べる破線と空白の長さをピクセルで表したものです。 If an odd number of values is provided, then the list of values is repeated to yield an even number of values. 既存の配列を削除するには、number_elements にゼロ、そして dash_array に null を渡します。 典型的な strokeDashArray_ 配列のメンバーは 5 3 2 となります。

パラメータ

dashArray

float の配列。

返り値

成功した場合に **TRUE** を返します。

ImagickDraw::setStrokeDashOffset

(No version information available, might be only in CVS)

ImagickDraw::setStrokeDashOffset — 破線パターンにおける破線の開始オフセットを指定する

説明

bool **ImagickDraw::setStrokeDashOffset** (float $dash_offset)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

破線パターンにおける破線の開始オフセットを指定します。

パラメータ

dash_offset

破線のオフセット。

返り値

値を返しません。

ImagickDraw::setStrokeLineCap

(No version information available, might be only in CVS)

ImagickDraw::setStrokeLineCap — 開かれたサブパスを描画する際に使用する端点の形状を指定する

説明

bool **ImagickDraw::setStrokeLineCap** (int $linecap)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

開かれたサブパスを描画する際に使用する端点の形状を指定します。

パラメータ

linecap

LINECAP_ 定数。

返り値

値を返しません。

ImagickDraw::setStrokeLineJoin

(No version information available, might be only in CVS)

ImagickDraw::setStrokeLineJoin — パスの角を描画する際に使用する形状を指定する

説明

bool **ImagickDraw::setStrokeLineJoin** (int $linejoin)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パス（あるいはその他のベクター図形）の角を描画する際に使用する形状を指定します。

パラメータ

linejoin

LINEJOIN_ 定数。

返り値

値を返しません。

ImagickDraw::setStrokeMiterLimit

(No version information available, might be only in CVS)

ImagickDraw::setStrokeMiterLimit — マイターリミットを指定する

説明

bool ***ImagickDraw::setStrokeMiterLimit*** *(int $miterlimit)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

マイターリミットを指定します。 2 本の直線が小さい角度で連結され、かつ連結方法が 'lineJoin' に設定されている場合、 出来上がる線の角の部分が非常に長くなります。マイターリミットは、miter length（角の長さ）と 'lineWidth' の比率の最大値です。

パラメータ

miterlimit

マイターリミット。

返り値

値を返しません。

ImagickDraw::setStrokeOpacity

(No version information available, might be only in CVS)

ImagickDraw::setStrokeOpacity — オブジェクトの枠線の透明度を指定する

説明

bool ***ImagickDraw::setStrokeOpacity*** *(float $stroke_opacity)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの枠線の透明度を指定します。

パラメータ

stroke_opacity

枠線の透明度。1.0 は完全に不透明な状態です。

返り値

値を返しません。

ImagickDraw::setStrokePatternURL

(No version information available, might be only in CVS)

ImagickDraw::setStrokePatternURL — オブジェクトの枠線の描画に使用するパターンを設定する

説明

bool ***ImagickDraw::setStrokePatternURL*** *(string $stroke_url)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの枠線の描画に使用するパターンを設定します。

パラメータ

stroke_url

URL。

返り値

imagick.imagickdraw.return.success;

ImagickDraw::setStrokeWidth

(No version information available, might be only in CVS)

ImagickDraw::setStrokeWidth — オブジェクトの枠線の描画に使用する線の幅を設定する

説明

bool **ImagickDraw::setStrokeWidth** (float $stroke_width)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

オブジェクトの枠線の描画に使用する線の幅を設定します。

パラメータ

stroke_width

線の幅。

返り値

値を返しません。

ImagickDraw::setTextAlignment

(No version information available, might be only in CVS)

ImagickDraw::setTextAlignment — テキストの配置を指定する

説明

bool **ImagickDraw::setTextAlignment** (int $alignment)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際の配置を指定します。

パラメータ

alignment

ALIGN_ 定数。

返り値

値を返しません。

ImagickDraw::setTextAntialias

(No version information available, might be only in CVS)

ImagickDraw::setTextAntialias — *テキストをアンチエイリアス処理するかどうかを制御する*

説明

bool ***ImagickDraw::setTextAntialias*** *(bool $antiAlias)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストをアンチエイリアス処理するかどうかを制御します。 デフォルトではテキストのアンチエイリアス処理を行います。

パラメータ

antiAlias

返り値

値を返しません。

ImagickDraw::setTextDecoration

(No version information available, might be only in CVS)

ImagickDraw::setTextDecoration — *装飾を指定する*

説明

bool ***ImagickDraw::setTextDecoration*** *(int $decoration)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に適用する装飾を指定します。

パラメータ

decoration

DECORATION_ 定数。

返り値

値を返しません。

ImagickDraw::setTextEncoding

(No version information available, might be only in CVS)

ImagickDraw::setTextEncoding — *テキストのコードセットを指定する*

説明

bool ***ImagickDraw::setTextEncoding*** *(string $encoding)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記を行う際に使用するコードセットを指定します。 現時点で指定できるエンコーディングは "UTF-8" のみで、 これはバイトシーケンスを Unicode で表記したものです。 空文字列を指定すると、システムのデフォルトを使用します。 Unicode によるテキストの注記を行うには、Unicode をサポートしたフォントが必要です。

パラメータ

encoding

エンコーディング名。

返り値

値を返しません。

ImagickDraw::setTextUnderColor

(No version information available, might be only in CVS)

ImagickDraw::setTextUnderColor — 背景の矩形の色を指定する

説明

bool **ImagickDraw::setTextUnderColor** (ImagickPixel $under_color)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストによる注記の背景に置く矩形の色を指定します。

パラメータ

under_color

背景色。

返り値

値を返しません。

ImagickDraw::setVectorGraphics

(No version information available, might be only in CVS)

ImagickDraw::setVectorGraphics — ベクターグラフィックを設定する

説明

bool **ImagickDraw::setVectorGraphics** (string $xml)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した ImagickDraw オブジェクトに関連付けるベクターグラフィックを設定します。 このメソッドを ImagickDraw::getVectorGraphics() と組み合わせて使用すると、 ベクターグラフィックの状態を持続させることができます。

パラメータ

xml

ベクターグラフィックを表す xml。

返り値

成功した場合に true、失敗した場合に false を返します。

ImagickDraw::setViewbox

(No version information available, might be only in CVS)

ImagickDraw::setViewbox — キャンバス船体の大きさを設定する

説明

bool **ImagickDraw::setViewbox** (int $x1 , int $y1 , int $x2 , int $y2)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ベクターデータの描画に使用するキャンバスの大きさを設定します。 通常は、これはキャンバス画像と同じ大きさに設定します。 ベクターデータを SVG や MVG 形式で保存する際に、 これを使用してキャンバス画像の大きさを指定します。 またビューアがベクターデータをレンダリングする際にもこれを使用します。

パラメータ

x1

左の x 座標。

y1

左の y 座標。

x2

右の x 座標。

y2

右の y 座標。

返り値

値を返しません。

ImagickDraw::skewX

(No version information available, might be only in CVS)

ImagickDraw::skewX — 現在の座標系を水平方向に傾ける

説明

bool ***ImagickDraw::skewX*** *(float $degrees)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の座標系を水平方向に傾けます。

パラメータ

degrees

傾斜角。

返り値

値を返しません。

ImagickDraw::skewY

(No version information available, might be only in CVS)

ImagickDraw::skewY — 現在の座標系を垂直方向に傾ける

説明

bool ***ImagickDraw::skewY*** *(float $degrees)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の座標系を垂直方向に傾けます。

パラメータ

degrees

傾斜角。

返り値

値を返しません。

ImagickDraw::translate

(No version information available, might be only in CVS)

ImagickDraw::translate — *現在の座標系に変換を適用する*

説明

bool ***ImagickDraw::translate*** *(float $x , float $y)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の座標系に変換を適用し、座標系の原点を指定した座標に移動します。

パラメータ

x

水平方向の変換。

y

垂直方向の変換。

返り値

値を返しません。

ImagickPixel::clear

(No version information available, might be only in CVS)

ImagickPixel::clear — *このオブジェクトに関連付けられたリソースを消去する*

説明

bool ***ImagickPixel::clear*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickPixel オブジェクトをクリアし、初期状態に戻します。 このオブジェクトに関連付けられている色も初期化します。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

ImagickPixel::__construct

(No version information available, might be only in CVS)

ImagickPixel::__construct — *ImagickPixel のコンストラクタ*

説明

ImagickPixel ***ImagickPixel::__construct*** *([string $color])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickPixel オブジェクトを作成します。color を指定した場合は、 オブジェクトを作成した後で色を初期化してから返します。

パラメータ

color

このオブジェクトの初期化に使用する色を表すオプションの文字列。

返り値

成功した場合に ImagickPixel オブジェクトを返します。 失敗した場合に ImagickPixelException をスローします。 failure.

ImagickPixel::destroy

(No version information available, might be only in CVS)

ImagickPixel::destroy — このオブジェクトに関連付けられているリソースの割り当てを解除する

説明

bool **ImagickPixel::destroy** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickPixel オブジェクトが使用しているリソースの割り当てを解除し、 色を初期化します。destroy 関数をコールした後は、このオブジェクトを使用することはできません。

パラメータ

返り値

成功した場合に **TRUE** を返します。

ImagickPixel::getColor

(No version information available, might be only in CVS)

ImagickPixel::getColor — 色を返す

説明

array **ImagickPixel::getColor** ([bool $normalized])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickPixel オブジェクトの色を、配列で返します。透明度が設定されている場合は、 4 つの要素からなる配列となります。

パラメータ

normalized

色を正規化する。

返り値

各チャネルの値を配列で返します。パラメータに true を指定した場合は、各値を正規化します。エラー時には ImagickPixelException をスローします。

ImagickPixel::getColorCount

(No version information available, might be only in CVS)

ImagickPixel::getColorCount — この色に関連付けられている色カウントを返す

説明

int **ImagickPixel::getColorCount** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この色に関連付けられている色カウントを返します。

パラメータ

返り値

成功した場合に色数を返します。失敗した場合に ImagickPixelException をスローします。

ImagickPixel::getColorValue

(No version information available, might be only in CVS)

ImagickPixel::getColorValue — 指定した色チャネルの値を正規化したものを取得する

説明

float **ImagickPixel::getColorValue** (int $color)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した色チャネルの値を取得します。 これは、0 から 1 までの間の浮動小数点数値となります。

パラメータ

color

調べたいチャネル。Imagick のチャネル定数で指定します。

返り値

チャネルの値を正規化した浮動小数点数値を返します。 エラー時には ImagickPixelException をスローします。

ImagickPixel::getHSL

(No version information available, might be only in CVS)

ImagickPixel::getHSL — ImagickPixel オブジェクトの HSL カラーを正規化したものを返す

説明

array **ImagickPixel::getHSL** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickPixel オブジェクトの HSL カラーを正規化したものを返します。 3 つの値は、それぞれ 0.0 から 1.0 までの間の浮動小数点数値となります。

パラメータ

返り値

HSL の値をそれぞれ "hue"、 "saturation"、そして "luminosity" というキーに保持する配列を返します。 失敗した場合に ImagickPixelException をスローします。

ImagickPixel::isSimilar

(No version information available, might be only in CVS)

ImagickPixel::isSimilar — この色と別の色の差を調べる

説明

bool **ImagickPixel::isSimilar** (ImagickPixel $color , float $fuzz)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この ImagickPixel オブジェクトと別のオブジェクトの色の差を調べます。 これは、両者の RGB 値を色立方体上にプロットすることで行います。ふたつの色の差が fuzz で指定した値よりも小さければ、 同じ色であるとみなされます。

パラメータ

color

このオブジェクトと比較したい ImagickPixel オブジェクト。

fuzz

同じ色であるとみなす最大の誤差。 理論上の最大値は、3 の平方根 (1.732) となります。

返り値

*成功した場合に **TRUE** を返します。*

ImagickPixel::setColorValue

(No version information available, might be only in CVS)

ImagickPixel::setColorValue — 指定したチャネルの正規化した値を設定する

説明

bool ***ImagickPixel::setColorValue*** *(int $color , float $value)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このオブジェクトの、指定したチャネルの値を正規化したものを設定します。 これは 0 から 1 までの値となります。この関数は、 ImagickPixel オブジェクトの透明度を指定する際にも使用します。

パラメータ

color

Imagick 色チャネル定数のいずれか。

value

そのチャネルに設定する値。0 から 1 までとなります。

返り値

*成功した場合に **TRUE** を返します。*

ImagickPixel::setColor

(No version information available, might be only in CVS)

ImagickPixel::setColor — 色を設定する

説明

bool ***ImagickPixel::setColor*** *(string $color)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この ImagickPixel オブジェクトの色を、文字列 (たとえば "blue"、"#0000ff"、"rgb(0,0,255)"、"cmyk(100,100,100,10)" などの形式) で設定します。

パラメータ

color

ImagickPixel オブジェクトを初期化する際に使用する色。

返り値

色が設定できた場合に true、それ以外の場合に false を返します。

ImagickPixel::setHSL

(No version information available, might be only in CVS)

ImagickPixel::setHSL — 正規化した HSL カラーを設定する

説明

bool **ImagickPixel::setHSL** (float $hue , float $saturation , float $luminosity)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

hue (色相)、saturation (彩度) および luminosity (明度) を正規化した値によって ImagickPixel オブジェクトの色を設定します。

パラメータ

hue

色相を正規化した値。これは、色相環上での位置 (0 から 1 まで) を表します。ゼロは赤です。

saturation

彩度を正規化した値。1 が最大です。

luminosity

明度を正規化した値。黒が 0、白が 1 で、 0.5 にすると完全な HS 値が再現されます。

返り値

成功した場合に **TRUE** を返します。

ImagickPixelIterator::clear

(No version information available, might be only in CVS)

ImagickPixelIterator::clear — PixelIterator に関連付けられたリソースを消去する

説明

bool **ImagickPixelIterator::clear** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

PixelIterator に関連付けられたリソースを消去します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

ImagickPixelIterator::__construct

(No version information available, might be only in CVS)

ImagickPixelIterator::__construct — ImagickPixelIterator のコンストラクタ

説明

ImagickPixelIterator **ImagickPixelIterator::__construct** (Imagick $wand)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ImagickPixelIterator のコンストラクタです。

パラメータ

返り値

成功した場合に **TRUE** を返します。

ImagickPixelIterator::destroy

(No version information available, might be only in CVS)

ImagickPixelIterator::destroy — PixelIterator に関連付けられているリソースの割り当てを解除する

説明

bool **ImagickPixelIterator::destroy** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

PixelIterator に関連付けられているリソースの割り当てを解除します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

ImagickPixelIterator::getCurrentIteratorRow

(No version information available, might be only in CVS)

ImagickPixelIterator::getCurrentIteratorRow — ImagickPixel オブジェクトのの現在の行を返す

説明

array **ImagickPixelIterator::getCurrentIteratorRow** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator の現在の行を、ImagickPixel オブジェクトの配列で返します。

パラメータ

返り値

ImagickPixel オブジェクトの配列形式で行を返します。これを順次処理することが可能です。

ImagickPixelIterator::getIteratorRow

(No version information available, might be only in CVS)

ImagickPixelIterator::getIteratorRow — pixel iterator の現在の行を返す

説明

int **ImagickPixelIterator::getIteratorRow** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator の現在の行を返します。

パラメータ

返り値

pixel iterator の現在の行を返します。 エラー時には ImagickPixelIteratorException をスローします。

ImagickPixelIterator::getNextIteratorRow

(No version information available, might be only in CVS)

ImagickPixelIterator::getNextIteratorRow — pixel iterator の次の行を返す

説明

array ***ImagickPixelIterator::getNextIteratorRow*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator の次の行を、pixel wands の配列として返します。

パラメータ

返り値

次の行を ImagickPixel オブジェクトの配列で返します。 エラー時には ImagickPixelIteratorException をスローします。

ImagickPixelIterator::getPreviousIteratorRow

(No version information available, might be only in CVS)

ImagickPixelIterator::getPreviousIteratorRow — 前の行を返す

説明

array ***ImagickPixelIterator::getPreviousIteratorRow*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator の前の行を、pixel wands の配列として返します。

パラメータ

返り値

ImagickPixelIterator の前の行を、ImagickPixelWand オブジェクトの配列として返します。エラー時には ImagickPixelIteratorException をスローします。

ImagickPixelIterator::newPixelIterator

(No version information available, might be only in CVS)

ImagickPixelIterator::newPixelIterator — 新しい pixel iterator を返す

説明

bool ***ImagickPixelIterator::newPixelIterator*** *(Imagick $wand)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい pixel iterator を返します。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。 ImagickPixelIteratorException をスローします。*

ImagickPixelIterator::newPixelRegionIterator

(No version information available, might be only in CVS)

ImagickPixelIterator::newPixelRegionIterator — 新しい pixel iterator を返す

説明

bool ***ImagickPixelIterator::newPixelRegionIterator*** *(Imagick $wand , int $x , int $y , int $columns , int $rows)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい pixel iterator を返します。

パラメータ

wand

x

y

columns

rows

返り値

成功した場合に新しい ImagickPixelIterator を返します。 失敗した場合は ImagickPixelIteratorException をスローします。

ImagickPixelIterator::resetIterator

(No version information available, might be only in CVS)

ImagickPixelIterator::resetIterator — pixel iterator をリセットする

説明

bool ***ImagickPixelIterator::resetIterator*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator をリセットします。 ImagickPixelIterator::getNextIteratorRow() と組み合わせて使用し、 ピクセルコンテナのすべてのピクセルを順にたどります。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

ImagickPixelIterator::setIteratorFirstRow

(No version information available, might be only in CVS)

ImagickPixelIterator::setIteratorFirstRow — pixel iterator を最初の行に設定する

説明

bool ***ImagickPixelIterator::setIteratorFirstRow*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator を最初の行に設定します。

パラメータ

返り値

成功した場合に ***TRUE*** *を返します。*

ImagickPixelIterator::setIteratorLastRow

(No version information available, might be only in CVS)

ImagickPixelIterator::setIteratorLastRow — pixel iterator を最後の行に設定する

説明

bool **ImagickPixelIterator::setIteratorLastRow** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator を最後の行に設定します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

ImagickPixelIterator::setIteratorRow

(No version information available, might be only in CVS)

ImagickPixelIterator::setIteratorRow — pixel iterator の行を設定する

説明

bool **ImagickPixelIterator::setIteratorRow** (int $row)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator の行を設定します。

パラメータ

row

返り値

成功した場合に **TRUE** を返します。

ImagickPixelIterator::syncIterator

(No version information available, might be only in CVS)

ImagickPixelIterator::syncIterator — pixel iterator を同期する

説明

bool **ImagickPixelIterator::syncIterator** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

pixel iterator を同期します。

パラメータ

返り値

成功した場合に **TRUE** を返します。

目次

- *Imagick::adaptiveResizeImage* — データに依存する三角測量にもとづいて画像のサイズを変更する
- *Imagick::adaptiveSharpenImage* — 順応して画像をシャープにする
- *Imagick::adaptiveThresholdImage* — 輝度の範囲にもとづいて各ピクセルの閾値を選択する
- *Imagick::addImage* — 新しい画像を Imagick オブジェクトの画像リストに追加する
- *Imagick::addNoiseImage* — ランダムなノイズを画像に追加する
- *Imagick::affineTransformImage* — 画像を変換する
- *Imagick::annotateImage* — 画像にテキストによる注記を加える
- *Imagick::appendImages* — 画像群を追加する
- *Imagick::averageImages* — 画像群を平均化する
- *Imagick::blackThresholdImage* — 閾値に満たないすべてのピクセルを黒にする
- *Imagick::blurImage* — blur（にじみ）フィルタを画像に追加する
- *Imagick::borderImage* — 画像の周りを枠線で囲む
- *Imagick::charcoalImage* — 木炭画をシミュレートする
- *Imagick::chopImage* — 画像の一部を取り除き、切り詰める
- *Imagick::clear* — Imagick オブジェクトに関連付けられたすべてのリソースをクリアする
- *Imagick::clipImage* — 8BIM プロファイルの最初のパスにそって切り取る
- *Imagick::clipPathImage* — 8BIM プロファイルの指定した名前のパスにそって切り取る
- *Imagick::clone* — Imagick オブジェクトの完全なコピーを作成する
- *Imagick::coalesceImages* — 複数の画像を合成する
- *Imagick::colorFloodfillImage* — 対象にマッチする任意の点の色の値を変更する
- *Imagick::colorizeImage* — 塗りつぶし色と画像を混合する
- *Imagick::clutImage* — ルックアップテーブルをもとに画像の色を置き換える
- *Imagick::combineImages* — ひとつあるいは複数の画像をひとつにまとめる
- *Imagick::commentImage* — コメントを画像に追加する
- *Imagick::compareImageChannels* — ひとつあるいは複数の画像の差を返す
- *Imagick::compareImageLayers* — 複数の画像の中で最大の境界範囲を返す
- *Imagick::compositeImage* — ある画像を別の画像に合成する
- *Imagick::__construct* — Imagick のコンストラクタ
- *Imagick::contrastImage* — 画像のコントラストを変更する
- *Imagick::contrastStretchImage* — カラー画像のコントラストを強調する
- *Imagick::convolveImage* — 独自の畳み込み関数を画像に適用する
- *Imagick::cropImage* — 画像の一部を抽出する
- *Imagick::current* — 画像のポインタを正しいシーケンスに設定する
- *Imagick::cycleColormapImage* — 画像のカラーマップを移動する
- *Imagick::deconstructImages* — 画像間の特定のピクセルの差を返す
- *Imagick::despeckleImage* — 画像内のスペックルノイズを軽減する
- *Imagick::destroy* — Imagick オブジェクトを破棄する
- *Imagick::drawImage* — 現在の画像上の ImagickDrawing オブジェクトをレンダリングする
- *Imagick::edgeImage* — 画像の輪郭を強調する
- *Imagick::embossImage* — グレースケール画像に三次元効果を施して返す
- *Imagick::enhanceImage* — ノイジーな画像の品質を向上させる
- *Imagick::equalizeImage* — 画像ヒストグラムを均等化する
- *Imagick::evaluateImage* — 式を画像に適用する
- *Imagick::flattenImages* — 画像シーケンスをマージする
- *Imagick::flipImage* — 垂直方向に反転した画像を作成する
- *Imagick::flopImage* — 水平方向に反転した画像を作成する
- *Imagick::frameImage* — 三次元の枠線をシミュレートする
- *Imagick::fxImage* — 式を画像の各ピクセルに適用する
- *Imagick::gammaImage* — 画像をガンマ補正する
- *Imagick::gaussianBlurImage* — 画像をぼかす
- *Imagick::getCompressionQuality* — オブジェクトの圧縮品質を取得する
- *Imagick::getCompression* — オブジェクトの圧縮形式を取得する
- *Imagick::getCopyright* — ImageMagick API の著作権情報を文字列定数で返す
- *Imagick::getFilename* — 画像シーケンスに関連付けられたファイル名を取得する
- *Imagick::getFormat* — Imagick オブジェクトのフォーマットを取得する
- *Imagick::getHomeURL* — ImageMagick のホーム URL を返す
- *Imagick::getImageBackgroundColor* — 画像の背景色を返す
- *Imagick::getImageBlob* — 画像シーケンスを blob で返す
- *Imagick::getImageBluePrimary* — 青が一番強い点を返す
- *Imagick::getImageBorderColor* — 画像の前景色を返す
- *Imagick::getImageChannelDepth* — 特定の画像チャネルの深度を返す
- *Imagick::getImageChannelDistortion* — 画像のチャネルを再構築した画像と比較する
- *Imagick::getImageChannelExtrema* — ひとつあるいは複数の画像チャネルの極値を取得する
- *Imagick::getImageChannelMean* — 平均値と標準偏差を取得する

- *Imagick::getImageChannelStatistics* — 画像の各チャネルの統計情報を返す
- *Imagick::getImageColormapColor* — 指定したインデックスに対応する色マップ上の色を返す
- *Imagick::getImageColorspace* — 画像の色空間を取得する
- *Imagick::getImageColors* — 画像で使われている色の数を取得する
- *Imagick::getImageCompose* — 画像の合成演算子を返す
- *Imagick::getImageDelay* — 画像の遅延を取得する
- *Imagick::getImageDepth* — 画像の深度を取得する
- *Imagick::getImageDispose* — 画像の配置方法を取得する
- *Imagick::getImageDistortion* — ある画像と再構築した画像を比較する
- *Imagick::getImageExtrema* — 画像の極値を取得する
- *Imagick::getImageFilename* — シーケンス内の特定の画像のファイル名を返す
- *Imagick::getImageSize* — 画像の長さをバイト数で返す
- *Imagick::getImageLength* — 画像の長さをバイト数で取得する
- *Imagick::getImageFormat* — シーケンス内の特定の画像のフォーマットを返す
- *Imagick::getImageGamma* — 画像のガンマを取得する
- *Imagick::getImageGeometry* — 幅と高さを連想配列で取得する
- *Imagick::getImageGreenPrimary* — 緑が一番強い点を返す
- *Imagick::getImageHeight* — 画像の高さを返す
- *Imagick::getImageHistogram* — 画像のヒストグラムを取得する
- *Imagick::getImageIndex* — 現在アクティブな画像のインデックスを取得する
- *Imagick::getIteratorIndex* — 現在アクティブな画像のインデックスを取得する
- *Imagick::getImageInterlaceScheme* — 画像のインターレース手法を取得する
- *Imagick::getImageInterpolateMethod* — 画像の補間方式を返す
- *Imagick::getImageIterations* — 画像の反復を取得する
- *Imagick::getImageMatteColor* — 画像のマット色を返す
- *Imagick::getImageMatte* — 画像がマットチャネルを持っているかどうかを返す
- *Imagick::getImagePage* — ページのジオメトリを返す
- *Imagick::getImagePixelColor* — 指定したピクセルの色を返す
- *Imagick::getImageProfile* — 指定した名前の画像プロファイルを返す
- *Imagick::getImageProfiles* — 画像プロファイルを返す
- *Imagick::getImageProperty* — 指定した名前の画像のプロパティを返す
- *Imagick::getImageProperties* — 画像のプロパティを返す
- *Imagick::getImageRedPrimary* — 赤が一番強い点を返す
- *Imagick::getImageRegion* — 画像の一部を抽出する
- *Imagick::getImageRenderingIntent* — 画像のレンダリング方向を取得する
- *Imagick::getImageResolution* — 画像の X 方向と Y 方向の解像度を取得する
- *Imagick::getImageScene* — 画像のシーンを取得する
- *Imagick::getImageSignature* — SHA-256 メッセージダイジェストを生成する
- *Imagick::getImageTicksPerSecond* — 画像の ticks-per-second を取得する
- *Imagick::getImageTotalInkDensity* — 画像の総インク密度を取得する
- *Imagick::getImageType* — 画像の型を取得する
- *Imagick::getImageUnits* — 画像の解像度の単位を取得する
- *Imagick::getImageVirtualPixelMethod* — 仮想ピクセルメソッドを取得する
- *Imagick::getImageWhitePoint* — 色度が白い点を返す
- *Imagick::getImageWidth* — 画像の幅を返す
- *Imagick::getImage* — 新しい Imagick オブジェクトを返す
- *Imagick::getInterlaceScheme* — オブジェクトのインターレース方式を取得する
- *Imagick::getImageOrientation* — 画像の方向を取得する
- *Imagick::getNumberImages* — オブジェクト内の画像の数を返す
- *Imagick::getOption* — 指定したキーに対応する値を返す
- *Imagick::getPackageName* — ImageMagick パッケージ名を返す
- *Imagick::getPage* — ページのジオメトリを返す
- *Imagick::getPixelIterator* — MagickPixelIterator を返す
- *Imagick::getPixelRegionIterator* — 画像セクションの ImagickPixelIterator を取得する
- *Imagick::getQuantumDepth* — quantum depth を取得する
- *Imagick::getQuantumRange* — Imagick quantum range を返す
- *Imagick::getReleaseDate* — ImageMagick のリリース日を返す
- *Imagick::getResourceLimit* — 指定したリソースの制限を返す
- *Imagick::getResource* — 指定したリソースのメモリ使用状況を返す
- *Imagick::getSamplingFactors* — 水平方向および垂直方向のサンプリング係数を取得する
- *Imagick::getSizeOffset* — サイズのオフセットを返す
- *Imagick::getSize* — Imagick オブジェクトのサイズを取得する
- *Imagick::getVersion* — ImageMagick API のバージョンを返す
- *Imagick::hasNextImage* — オブジェクトが次の画像を保持しているかどうかを調べる

- Imagick::hasPreviousImage — オブジェクトが前の画像を保持しているかどうかを調べる
- Imagick::identifyImage — 画像を識別し、属性を取得する
- Imagick::implodeImage — 新しい画像をコピーとして作成する
- Imagick::labelImage — ラベルを画像に追加する
- Imagick::levelImage — 画像のレベルを調節する
- Imagick::linearStretchImage — 画像の輝度を引き伸ばして飽和させる
- Imagick::magnifyImage — 画像を 2 倍に比例拡大する
- Imagick::matteFloodfillImage — 色の透明度を変更する
- Imagick::medianFilterImage — デジタルフィルタを適用する
- Imagick::minifyImage — 画像をその半分のサイズに比例縮小する
- Imagick::modulateImage — 明度、飽和度、色相を制御する
- Imagick::montageImage — 合成画像を作成する
- Imagick::morphImages — 複数の画像をモーフィングする
- Imagick::compareImages — ある画像を再構築された画像と比較する
- Imagick::mosaicImages — 画像からモザイクを作成する
- Imagick::motionBlurImage — モーションブラーをシミュレートする
- Imagick::negateImage — 画像の色を打ち消す
- Imagick::distortImage — さまざまな方式で画像を歪める
- Imagick::newImage — 新しい画像を作成する
- Imagick::setImage — オブジェクト内の画像を置き換える
- Imagick::setImageOpacity — 画像の不透明度を設定する
- Imagick::newPseudoImage — 新しい画像を作成する
- Imagick::nextImage — 次の画像に移動する
- Imagick::normalizeImage — カラー画像のコントラストを強調する
- Imagick::oilPaintImage — 油絵をシミュレートする
- Imagick::optimizeImageLayers — 画像の繰り返し部分を削除して最適化する
- Imagick::paintOpaqueImage — 色にマッチするピクセルを変更する
- Imagick::paintTransparentImage — 色にマッチするピクセルを塗りつぶし色に変更する
- Imagick::pingImageBlob — 手早く属性を取得する
- Imagick::pingImageFile — 画像の基本属性を手軽に取得する
- Imagick::pingImage — 画像の基本属性を取得する
- Imagick::posterizeImage — 指定した色数まで画像を減色する
- Imagick::previousImage — オブジェクト内の前の画像に移動する
- Imagick::profileImage — 画像のプロファイルを追加あるいは削除する
- Imagick::queryFormats — Imagick がサポートするフォーマットを返す
- Imagick::radialBlurImage — 画像にラジアルブラーを施す
- Imagick::raiseImage — 三次元のボタン風の効果をシミュレートする
- Imagick::randomThresholdImage — コントラストの高い 2 色の画像を作成する
- Imagick::readImageBlob — バイナリ文字列から画像を読み込む
- Imagick::readImageFile — オープンしているファイルハンドルから画像を読み込む
- Imagick::readImage — ファイルから画像を読み込む
- Imagick::reduceNoiseImage — 画像の輪郭をなめらかにする
- Imagick::removeImageProfile — 指定した名前の画像プロファイルを削除してそれを返す
- Imagick::removeImage — 画像リストから画像を削除する
- Imagick::render — それまでのすべての描画コマンドをレンダリングする
- Imagick::resampleImage — 画像を指定した解像度にリサンプリングする
- Imagick::resizeImage — 画像のサイズを変更する
- Imagick::rollImage — 画像を補正する
- Imagick::rotateImage — 画像を回転する
- Imagick::sampleImage — ピクセルのサンプリングによって画像の倍率を変更する
- Imagick::scaleImage — 画像のサイズを変更する
- Imagick::separateImageChannel — 画像からチャネルを分離する
- Imagick::sepiaToneImage — 画像をセピア調にする
- Imagick::setBackgroundColor — オブジェクトのデフォルト背景色を設定する
- Imagick::setCompressionQuality — オブジェクトのデフォルトの圧縮品質を設定する
- Imagick::setCompression — オブジェクトのデフォルトの圧縮方式を設定する
- Imagick::setFilename — 画像を読み書きする前にファイル名を設定する
- Imagick::setFirstIterator — Imagick イテレータを最初の画像に設定する
- Imagick::setLastIterator — Imagick イテレータを最後の画像に設定する
- Imagick::setFormat — Imagick オブジェクトのフォーマットを設定する
- Imagick::setImageBackgroundColor — 画像の背景色を設定する
- Imagick::setImageBias — 画像を折りたたむ任意のメソッドについて画像のバイアスを設定する
- Imagick::setImageBluePrimary — 青が一番強い点を設定する
- Imagick::setImageBorderColor — 画像の前景色を設定する

- Imagick::setImageChannelDepth — 特定の画像チャネルの深度を設定する
- Imagick::setImageColormapColor — 指定した色マップインデックスの色を設定する
- Imagick::setImageColorspace — 画像の色空間を設定する
- Imagick::setImageCompose — 画像の合成演算子を設定する
- Imagick::setImageCompression — 画像の圧縮を設定する
- Imagick::setImageDelay — 画像の遅延を設定する
- Imagick::setImageDepth — 画像の深度を設定する
- Imagick::setImageDispose — 画像の配置方法を設定する
- Imagick::setImageExtent — 画像のサイズを設定する
- Imagick::setImageFilename — 特定の画像のファイル名を設定する
- Imagick::setImageFormat — 特定の画像のフォーマットを設定する
- Imagick::setImageGamma — 画像のガンマを設定する
- Imagick::setImageGreenPrimary — 緑が一番強い点を設定する
- Imagick::setImageIndex — イテレータの位置を設定する
- Imagick::setIteratorIndex — イテレータの位置を設定する
- Imagick::setImageInterlaceScheme — 画像のインターレース手法を設定する
- Imagick::setImageInterpolateMethod — 画像のピクセル補間方式を設定する
- Imagick::setImageIterations — 画像の反復を設定する
- Imagick::setImageMatteColor — 画像のマット色を設定する
- Imagick::setImageMatte — 画像のマットチャネルを設定する
- Imagick::setImagePage — 画像のページのジオメトリを設定する
- Imagick::setImageProfile — 指定した名前の画像プロファイルを Imagick オブジェクトに追加する
- Imagick::setImageProperty — 画像のプロパティを設定する
- Imagick::setImageRedPrimary — 赤が一番強い点を設定する
- Imagick::setImageRenderingIntent — 画像のレンダリング方向を設定する
- Imagick::setImageResolution — 画像の解像度を設定する
- Imagick::setImageScene — 画像のシーンを設定する
- Imagick::setImageTicksPerSecond — 画像の ticks-per-second を設定する
- Imagick::setImageType — 画像の型を設定する
- Imagick::setImageUnits — 画像の解像度の単位を設定する
- Imagick::setImageVirtualPixelMethod — 画像の仮想ピクセルメソッドを設定する
- Imagick::setImageWhitePoint — 画像の色度が白い点を設定する
- Imagick::setInterlaceScheme — 画像の圧縮を設定する
- Imagick::setImageOrientation — 画像の方向を設定する
- Imagick::setOption — オプションを設定する
- Imagick::setPage — Imagick オブジェクトのページジオメトリを設定する
- Imagick::setResolution — 画像の解像度を設定する
- Imagick::setResourceLimit — 特定のリソースの制限をメガバイト単位で設定する
- Imagick::setSamplingFactors — 画像のサンプリング係数を設定する
- Imagick::setSizeOffset — Imagick オブジェクトのサイズのオフセットを設定する
- Imagick::setSize — Imagick オブジェクトのサイズを設定する
- Imagick::setType — 画像タイプ属性を設定する
- Imagick::shadeImage — 3D 効果を作成する
- Imagick::shadowImage — 画像の影をシミュレートする
- Imagick::sharpenImage — 画像をシャープにする
- Imagick::shaveImage — 画像の輪郭からピクセルを刈り取る
- Imagick::shearImage — 平行四辺形を作成する
- Imagick::sigmoidalContrastImage — 画像のコントラストを調整する
- Imagick::sketchImage — 鉛筆画をシミュレートする
- Imagick::solarizeImage — 画像にソラリゼーション効果を適用する
- Imagick::spliceImage — 無地の画像を作成する
- Imagick::spreadImage — ブロック内の各ピクセルをランダムに移動する
- Imagick::steganoImage — デジタル透かしを画像に埋め込む
- Imagick::stereoImage — ふたつの画像を合成する
- Imagick::stripImage — 画像からすべてのプロパティやコメントを除去する
- Imagick::swirlImage — 画像の中心から、ピクセルを渦巻状にする
- Imagick::textureImage — テクスチャ画像をタイル状に並べる
- Imagick::thresholdImage — 閾値にもとづいて個々のピクセルの値を変更する
- Imagick::thumbnailImage — 画像のサイズを変更する
- Imagick::tintImage — 色ベクトルを画像の各ピクセルに適用する
- Imagick::transverseImage — 水平方向に反転させた画像を作成する
- Imagick::trimImage — 画像の輪郭を削除する
- Imagick::uniqueImageColors — ある 1 色以外のすべての色のピクセルを削除する
- Imagick::unsharpMaskImage — 画像をシャープにする

- Imagick::valid — 現在のアイテムが有効かどうかを調べる
- Imagick::vignetteImage — ビネットフィルタを画像に追加する
- Imagick::waveImage — ウェーブフィルタを画像に追加する
- Imagick::whiteThresholdImage — 閾値に満たないすべてのピクセルを白にする
- Imagick::writeImages — 画像あるいは画像シーケンスを書き込む
- Imagick::writeImage — 指定した名前で画像を書き込む
- Imagick::displayImage — 画像を表示する
- Imagick::displayImages — 画像あるいは画像シーケンスを表示する
- Imagick::cropThumbnailImage — 切り取ってサムネイルを作成する
- Imagick::roundCorners — 画像の角を丸める
- Imagick::polaroidImage — ポラロイド写真をシミュレートする
- Imagick::queryFonts — 設定したフォントを返す
- Imagick::queryFontMetrics — フォントメトリクスを表す配列を返す
- ImagickDraw::affine — 現在のアフィン変換行列を設定する
- ImagickDraw::annotation — 画像上にテキストを描画する
- ImagickDraw::arc — 円弧を描画する
- ImagickDraw::bezier — ベジエ曲線を描画する
- ImagickDraw::circle — 円を描画する
- ImagickDraw::clear — ImagickDraw をクリアする
- ImagickDraw::clone — 指定した ImagickDraw オブジェクトの完全なコピーを作成する
- ImagickDraw::color — 画像上に色を描画する
- ImagickDraw::comment — コメントを追加する
- ImagickDraw::composite — 現在の画像上に別の画像を合成する
- ImagickDraw::__construct — ImagickDraw コンストラクタ
- ImagickDraw::destroy — 関連付けられたすべてのリソースを開放する
- ImagickDraw::ellipse — 画像上に楕円を描画する
- ImagickDraw::getClipPath — 現在のクリッピングパスの ID を取得する
- ImagickDraw::getClipRule — 現在の多角形塗りつぶしルールを返す
- ImagickDraw::getClipUnits — クリップパスの単位の解釈を返す
- ImagickDraw::getFillColor — 塗りつぶし色を返す
- ImagickDraw::getFillOpacity — 描画時の透過度を返す
- ImagickDraw::getFillRule — 塗りつぶしルールを返す
- ImagickDraw::getFontFamily — フォントファミリーを返す
- ImagickDraw::getFontSize — フォントのポイント数を返す
- ImagickDraw::getFontStyle — フォントのスタイルを返す
- ImagickDraw::getFontWeight — フォントの重さを返す
- ImagickDraw::getFont — フォントを返す
- ImagickDraw::getGravity — テキストの配置時の gravity を返す
- ImagickDraw::getStrokeAntialias — 現在の縁取りのアンチエイリアス設定を返す
- ImagickDraw::getStrokeColor — オブジェクトの縁取りに使用する色を返す
- ImagickDraw::getStrokeDashArray — パスの描画に使用する破線のパターンを表す配列を返す
- ImagickDraw::getStrokeDashOffset — 破線パターンにおける破線の開始オフセットを返す
- ImagickDraw::getStrokeLineCap — 開かれたサブパスを描画する際に使用する端点の形状を返す
- ImagickDraw::getStrokeLineJoin — パスの角を描画する際に使用する形状を返す
- ImagickDraw::getStrokeMiterLimit — マイターリミットを返す
- ImagickDraw::getStrokeOpacity — オブジェクトの枠線の透明度を返す
- ImagickDraw::getStrokeWidth — オブジェクトの枠線の描画に使用する線の幅を返す
- ImagickDraw::getTextAlignment — テキストの配置を返す
- ImagickDraw::getTextAntialias — 現在のテキストのアンチエイリアス設定を返す
- ImagickDraw::getTextDecoration — テキストの装飾を返す
- ImagickDraw::getTextEncoding — テキストによる注記の際に使用するコードセットを返す
- ImagickDraw::getTextUnderColor — テキストの背景色を返す
- ImagickDraw::getVectorGraphics — ベクターグラフィックを含む文字列を返す
- ImagickDraw::line — 直線を描画する
- ImagickDraw::matte — 画像の opacity チャネル上に描画する
- ImagickDraw::pathClose — パス要素を現在のパスに追加する
- ImagickDraw::pathCurveToAbsolute — 三次ベジエ曲線を描画する
- ImagickDraw::pathCurveToQuadraticBezierAbsolute — 二次ベジエ曲線を描画する
- ImagickDraw::pathCurveToQuadraticBezierRelative — 二次ベジエ曲線を描画する
- ImagickDraw::pathCurveToQuadraticBezierSmoothAbsolute — 二次ベジエ曲線を描画する
- ImagickDraw::pathCurveToQuadraticBezierSmoothRelative — 二次ベジエ曲線を描画する
- ImagickDraw::pathCurveToRelative — 三次ベジエ曲線を描画する
- ImagickDraw::pathCurveToSmoothAbsolute — 三次ベジエ曲線を描画する
- ImagickDraw::pathCurveToSmoothRelative — 三次ベジエ曲線を描画する

- ImagickDraw::pathEllipticArcAbsolute — 楕円弧を描画する
- ImagickDraw::pathEllipticArcRelative — 楕円弧を描画する
- ImagickDraw::pathFinish — 現在のパスを終了する
- ImagickDraw::pathLineToAbsolute — 直線パスを描画する
- ImagickDraw::pathLineToHorizontalAbsolute — 水平直線パスを描画する
- ImagickDraw::pathLineToHorizontalRelative — 水平直線パスを描画する
- ImagickDraw::pathLineToRelative — 直線パスを描画する
- ImagickDraw::pathLineToVerticalAbsolute — 垂直直線パスを描画する
- ImagickDraw::pathLineToVerticalRelative — 垂直直線パスを描画する
- ImagickDraw::pathMoveToAbsolute — 新しいサブパスを開始する
- ImagickDraw::pathMoveToRelative — 新しいサブパスを開始する
- ImagickDraw::pathStart — パス描画リストの開始を宣言する
- ImagickDraw::point — 点を描画する
- ImagickDraw::polygon — 多角形を描画する
- ImagickDraw::polyline — 線分群を描画する
- ImagickDraw::popClipPath — クリップパスの定義を終了する
- ImagickDraw::popDefs — 定義リストを終了する
- ImagickDraw::pop — スタック内の現在の ImagickDraw を破棄し、事前に push された ImagickDraw を返す
- ImagickDraw::popPattern — パターン定義を終了する
- ImagickDraw::pushClipPath — クリップパスの定義を開始する
- ImagickDraw::pushDefs — 後に続くコマンドが、処理の前に名前つき要素を作成することを示す
- ImagickDraw::push — 現在の ImagickDraw をコピーしてスタックに格納する
- ImagickDraw::pushPattern — 後に続く ImagickDraw::opPattern() までのコマンドが、名前付きパターンを構成することを示す
- ImagickDraw::rectangle — 矩形を描画する
- ImagickDraw::render — これまでのすべての描画コマンドを画像上にレンダリングする
- ImagickDraw::rotate — 指定した回転を現在の座標空間に適用する
- ImagickDraw::roundRectangle — 角が丸い矩形を描画する
- ImagickDraw::scale — 倍率を調整する
- ImagickDraw::setClipPath — 指定した名前のクリッピングパスを画像に関連付ける
- ImagickDraw::setClipRule — クリッピングパスで使用する多角形塗りつぶしルールを設定する
- ImagickDraw::setClipUnits — クリップパスの単位の解釈を設定する
- ImagickDraw::setFillAlpha — 色やテクスチャによる塗りつぶしの際の透過度を設定する
- ImagickDraw::setFillColor — オブジェクトの塗りつぶしに使用する色を設定する
- ImagickDraw::setFillOpacity — 色やテクスチャによる塗りつぶしの際の透過度を設定する
- ImagickDraw::setFillPatternURL — オブジェクトの塗りつぶしパターンとして使用する URL を設定する
- ImagickDraw::setFillRule — 多角形の描画時に使用する塗りつぶしルールを設定する
- ImagickDraw::setFontFamily — テキストによる注記を行う際に使用するフォントファミリーを設定する
- ImagickDraw::setFontSize — テキストによる注記を行う際に使用するフォントのポイント数を設定する
- ImagickDraw::setFontStretch — テキストによる注記を行う際に使用するフォントの伸縮を設定する
- ImagickDraw::setFontStyle — テキストによる注記を行う際に使用するフォントのスタイルを設定する
- ImagickDraw::setFontWeight — フォントの重さを設定する
- ImagickDraw::setFont — テキストによる注記を行う際に使用するフォントを設定する
- ImagickDraw::setGravity — テキストの配置時の gravity を設定する
- ImagickDraw::setStrokeAlpha — オブジェクトの枠線の透明度を指定する
- ImagickDraw::setStrokeAntialias — 縁取りの枠線をアンチエイリアス処理するかどうかを制御する
- ImagickDraw::setStrokeColor — オブジェクトの縁取りに使用する色を設定する
- ImagickDraw::setStrokeDashArray — パスの描画に使用する破線のパターンを指定する
- ImagickDraw::setStrokeDashOffset — 破線パターンにおける破線の開始オフセットを指定する
- ImagickDraw::setStrokeLineCap — 開かれたサブパスを描画する際に使用する端点の形状を指定する
- ImagickDraw::setStrokeLineJoin — パスの角を描画する際に使用する形状を指定する
- ImagickDraw::setStrokeMiterLimit — マイターリミットを指定する
- ImagickDraw::setStrokeOpacity — オブジェクトの枠線の透明度を指定する
- ImagickDraw::setStrokePatternURL — オブジェクトの枠線の描画に使用するパターンを設定する
- ImagickDraw::setStrokeWidth — オブジェクトの枠線の描画に使用する線の幅を設定する
- ImagickDraw::setTextAlignment — テキストの配置を指定する
- ImagickDraw::setTextAntialias — テキストをアンチエイリアス処理するかどうかを制御する
- ImagickDraw::setTextDecoration — 装飾を指定する
- ImagickDraw::setTextEncoding — テキストのコードセットを指定する
- ImagickDraw::setTextUnderColor — 背景の矩形の色を指定する
- ImagickDraw::setVectorGraphics — ベクターグラフィックを設定する
- ImagickDraw::setViewbox — キャンバス船体の大きさを設定する
- ImagickDraw::skewX — 現在の座標系を水平方向に傾ける
- ImagickDraw::skewY — 現在の座標系を垂直方向に傾ける
- ImagickDraw::translate — 現在の座標系に変換を適用する

- ImagickPixel::clear — このオブジェクトに関連付けられたリソースを消去する
- ImagickPixel::__construct — ImagickPixel のコンストラクタ
- ImagickPixel::destroy — このオブジェクトに関連付けられているリソースの割り当てを解除する
- ImagickPixel::getColor — 色を返す
- ImagickPixel::getColorCount — この色に関連付けられている色カウントを返す
- ImagickPixel::getColorValue — 指定した色チャネルの値を正規化したものを取得する
- ImagickPixel::getHSL — ImagickPixel オブジェクトの HSL カラーを正規化したものを返す
- ImagickPixel::isSimilar — この色と別の色の差を調べる
- ImagickPixel::setColorValue — 指定したチャネルの正規化した値を設定する
- ImagickPixel::setColor — 色を設定する
- ImagickPixel::setHSL — 正規化した HSL カラーを設定する
- ImagickPixelIterator::clear — PixelIterator に関連付けられたリソースを消去する
- ImagickPixelIterator::__construct — ImagickPixelIterator のコンストラクタ
- ImagickPixelIterator::destroy — PixelIterator に関連付けられているリソースの割り当てを解除する
- ImagickPixelIterator::getCurrentIteratorRow — ImagickPixel オブジェクトのの現在の行を返す
- ImagickPixelIterator::getIteratorRow — pixel iterator の現在の行を返す
- ImagickPixelIterator::getNextIteratorRow — pixel iterator の次の行を返す
- ImagickPixelIterator::getPreviousIteratorRow — 前の行を返す
- ImagickPixelIterator::newPixelIterator — 新しい pixel iterator を返す
- ImagickPixelIterator::newPixelRegionIterator — 新しい pixel iterator を返す
- ImagickPixelIterator::resetIterator — pixel iterator をリセットする
- ImagickPixelIterator::setIteratorFirstRow — pixel iterator を最初の行に設定する
- ImagickPixelIterator::setIteratorLastRow — pixel iterator を最後の行に設定する
- ImagickPixelIterator::setIteratorRow — pixel iterator の行を設定する
- ImagickPixelIterator::syncIterator — pixel iterator を同期する

IMAP、POP3 および NNTP 関数

導入

これらの関数は、IMAP プロトコルのほかに NNTP や POP3、 そしてローカルのメールボックスへのアクセスもサポートします。

しかし、IMAP 関数の中には POP プロトコルでは正常に動作しないものがあることに注意しましょう。

要件

この拡張モジュールは、c クライアントライブラリがインストールされている ことを要します。» ftp://ftp.cac.washington.edu/imap/ から最新版を入手し、コンパイルしてください。

IMAP のソースファイルをシステムの include ディレクトリに直接コピー しないことが大切です。そのようなことをすると既存のファイルと衝突を引き起こす可能性があります。そのかわりにシステムの include ディレクトリの 下に新しいディレクトリ（例： /usr/local/imap-2000b/ 場所や名前は、あなたのシステムの設定や IMAP のバージョンに依存します）を作成し、 この新規ディレクトリの下に lib/、include/ という新しいディレクトリを作成します。IMAP ソースツリーの c-client ディレクトリからすべての *.h ファイルを include/ に、そしてすべての *.c ファイルを lib/ にコピーします。 IMAP をコンパイルすると、c-client.a というファイルが つくられます。これも lib/ ディレクトリにコピーし、名前を libc-client.a に変更してください。

> **注意**: C クライアントライブラリを SSL や Kerberos のサポートつきでビルド するには、パッケージとともに提供されるドキュメントを参照ください。

> **注意**: Mandrake Linux では、IMAP ライブラリ（libc-client.a） が Kerberos のサポートなしでコンパイルされています。また、SSL 対応バージョン（client-PHP4.a）が別に インストールされます。Kerberos サポートを使用するには、これらの ライブラリを再コンパイルする必要があります。

インストール手順

これらの関数を動作させるには、--with-imap[=DIR] を指定して PHP をコンパイルする必要があります。DIR は c-client インストール時のプレフィックスです。上の例のような場合には --with-imap=/usr/local/imap-2000b を指定します。この場所は、上の指示にしたがってあなたが作成したディレクトリの 場所に依存します。Windows ユーザは、php.ini で php_imap.dll DLL を有効にします。 IMAP は Windows 2000 より前のバージョンではサポートされていません。 なぜなら、メールサーバとの SSL 接続を有効にするために暗号化関数を 使用しているからです。

> **注意**: c-client の設定によっては、これ以外に --with-imap-ssl=/path/to/openssl/ や --with-kerberos=/path/to/kerberos といったオプションを PHP の configure で指定する必要があります。

警告

IMAP、recode、 YAZ および Cyrus 拡張モジュールは、組み合わせて使用することはできません。これは、これらすべてが同一の内部シンボルを使用しているためです。

実行時設定

設定ディレクティブは定義されていません。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

NIL (integer)
OP_DEBUG (integer)
OP_READONLY (integer)
メールボックスを読み込み専用で開きます。
OP_ANONYMOUS (integer)
ニュースで .newsrc を使用せず、更新も行いません （NNTP のみ）。
OP_SHORTCACHE (integer)
OP_SILENT (integer)
OP_PROTOTYPE (integer)
OP_HALFOPEN (integer)
IMAP や NNTP において、接続はオープンするがメールボックスを開きません。
OP_EXPUNGE (integer)
OP_SECURE (integer)
CL_EXPUNGE (integer)
imap_close() のコール時に、メールボックスを 閉じる前に中身を削除します。
FT_UID (integer)
パラメータは UID です。
FT_PEEK (integer)
まだ設定されていない場合に \Seen フラグを設定しません。
FT_NOT (integer)
FT_INTERNAL (integer)
返される文字列は内部フォーマットであり、CRLF を正規化しません。
FT_PREFETCHTEXT (integer)
ST_UID (integer)
シーケンスの引数に、番号ではなく UID を含んでいます。
ST_SILENT (integer)
ST_SET (integer)
CP_UID (integer)
シーケンス番号が UID を含んでいます。
CP_MOVE (integer)
imap_mail_copy() でコピーを行った後に 現在のメールボックスからメッセージを削除します。
SE_UID (integer)
シーケンス番号のかわりに UID を返します。
SE_FREE (integer)
SE_NOPREFETCH (integer)
検索されたメッセージを事前にフェッチすることはありません。
SO_FREE (integer)
SO_NOSERVER (integer)
SA_MESSAGES (integer)
SA_RECENT (integer)
SA_UNSEEN (integer)
SA_UIDNEXT (integer)
SA_UIDVALIDITY (integer)
SA_ALL (integer)
LATT_NOINFERIORS (integer)
このメールボックスには「子供」はいません（このメールボックスの配下には メールボックスはありません）。
LATT_NOSELECT (integer)
これは単なるコンテナであり、メールボックスではありません。 オープンすることはできません。
LATT_MARKED (integer)
このメールボックスはマークされています。UW-IMAPD でのみ使用されます。
LATT_UNMARKED (integer)
このメールボックスはマークされていません。UW-IMAPD でのみ使用されます。
SORTDATE (integer)
imap_sort() のソート条件。 メッセージの日付でソートします。
SORTARRIVAL (integer)
imap_sort() のソート条件。 到着日でソートします。
SORTFROM (integer)
imap_sort() のソート条件。 メールボックスの最初の From アドレスでソートします。
SORTSUBJECT (integer)
imap_sort() のソート条件。 メッセージの表題でソートします。
SORTTO (integer)
imap_sort() のソート条件。 メールボックスの最初の To アドレスでソートします。
SORTCC (integer)
imap_sort() のソート条件。 メールボックスの最初の Cc アドレスでソートします。
SORTSIZE (integer)
imap_sort() のソート条件。 メッセージのサイズ（バイト単位）でソートします。
TYPETEXT (integer)
TYPEMULTIPART (integer)
TYPEMESSAGE (integer)
TYPEAPPLICATION (integer)
TYPEAUDIO (integer)
TYPEIMAGE (integer)
TYPEVIDEO (integer)
TYPEOTHER (integer)
ENC7BIT (integer)
ENC8BIT (integer)
ENCBINARY (integer)
ENCBASE64 (integer)
ENCQUOTEDPRINTABLE (integer)
ENCOTHER (integer)
IMAP_OPENTIMEOUT (integer)
IMAP_READTIMEOUT (integer)
IMAP_WRITETIMEOUT (integer)
IMAP_CLOSETIMEOUT (integer)
LATT_REFERRAL (integer)
LATT_HASCHILDREN (integer)
LATT_HASNOCHILDREN (integer)
TYPEMODEL (integer)

参考

この文書では、提供される関数に関する全ての話題の詳細について立ち入ることは できません。より詳細な情報については、C クライアントライブラリのソースに 付属するドキュメント (docs/internal.txt) および以下の RFC ドキュメントで提供されています。

- » RFC2821: Simple Mail Transfer Protocol (SMTP).
- » RFC2822: Standard for ARPA internet text messages.

- » RFC2060: Internet Message Access Protocol (IMAP) Version 4rev1.
- » RFC1939: Post Office Protocol Version 3 (POP3).
- » RFC977: Network News Transfer Protocol (NNTP).
- » RFC2076: Common Internet Message Headers.
- » RFC2045 , » RFC2046 , » RFC2047 , » RFC2048 & » RFC2049: Multipurpose Internet Mail Extensions (MIME).

詳しい概観については、David Wood による書籍 » Programming Internet Email や Dianna Mullet と Kevin Mullet による » Managing IMAP でも得ることができます。

---

imap_8bit

(PHP 4, PHP 5)

imap_8bit — 8 ビット文字列を quoted-printable 文字列に変換する

説明

string **imap_8bit** (string $string)

(» RFC2045, section 6.7 に基づき) 8 ビット文字列を quoted-printable 文字列に変換します。

パラメータ

string

変換する 8 ビット文字列。

返り値

quoted-printable 文字列を返します。

参考

- imap_qprint()

---

imap_alerts

(PHP 4, PHP 5)

imap_alerts — 発生した IMAP 警告メッセージを返す

説明

array **imap_alerts** (void)

直近に **imap_alerts()** をコールして以来、 またはページ処理を開始して以来生成された 全 IMAP 警告メッセージの配列を返します。

imap_alerts() がコールされた場合、 その処理後に警告のスタックはクリアされます。IMAP 規約では、 これらのメッセージをユーザに渡すことが規定されています。

返り値

全ての IMAP 警告メッセージの配列、あるいは警告メッセージが発生していない場合は **FALSE** を返します。

参考

- imap_errors()

---

imap_append

(PHP 4, PHP 5)

imap_append — 指定されたメールボックスに文字列メッセージを追加する

説明

bool **imap_append** (resource $imap_stream , string $mailbox , string $message [, string $options])

指定したメールボックス mbox に文字列メッセージ message を追加します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックスの名前。詳細は imap_open() を参照ください。

message

追加したいメッセージを表す文字列。

Cyrus IMAP サーバと通信する際には、改行コードとして "\n" のかわりに "\r\n" を使用する必要があります。さもなくば、操作は失敗します。

options

指定した場合は、 options もそのメールボックスに書きこまれます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imap_append() の例

```
<?php
$stream = imap_open("{imap.example.org}INBOX.Drafts", "username", "password");

$check = imap_check($stream);
echo "Msg Count before append: ". $check->Nmsgs . "¥n";

imap_append($stream, "{imap.example.org}INBOX.Drafts"
                    , "From: me@example.com¥r¥n"
                    . "To: you@example.com¥r¥n"
                    . "Subject: test¥r¥n"
                    . "¥r¥n"
                    . "this is a test message, please ignore¥r¥n"
                    );

$check = imap_check($stream);
echo "Msg Count after append : ". $check->Nmsgs . "¥n";

imap_close($stream);
?>
```

---

imap_base64

(PHP 4, PHP 5)

imap_base64 — BASE64 でエンコードされたテキストをデコードする

説明

string **imap_base64** (string $text)

BASE64 でエンコードされたテキスト text をデコードします。

パラメータ

text

エンコードされたテキスト。

返り値

デコードしたメッセージを文字列で返します。

参考

- imap_binary()
- base64_encode()
- base64_decode()
- » RFC2045, Section 6.8

---

imap_binary

(PHP 4, PHP 5)

imap_binary — 8 ビット文字列を base64 文字列に変換する

説明

string **imap_binary** (string $string)

（» RFC2045, Section 6.8 に基づき）8 ビット文字列を base64 文字列に変換します。

パラメータ

string

8 ビット文字列。

返り値

base64 エンコードされた文字列を返します。

参考

- imap_base64()

---

imap_body

(PHP 4, PHP 5)

imap_body — メッセージ本文を読む

説明

string **imap_body** (resource $imap_stream , int $msg_number [, int $options])

imap_body() は、現在のメールボックスにある msg_number 番目のメッセージの本文を返します。

imap_body() は、メッセージの本文と全く同じ コピーのみを返します。マルチパート MIME エンコードされたメッセージの 一部を展開するには、その構造を解析するために **imap_fetch_structure()** を使用し、単一の部分要素の コピーを展開する際には、 imap_fetchbody() を使用する必要があります。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

options

オプションの options はビットマスクであり、 以下の要素の組み合わせとなります。

- **FT_UID** - msg_number は UID です
- **FT_PEEK** - 既に設定されていない場合、\Seen フラグを設定しない
- **FT_INTERNAL** - 内部フォーマットで文字列を返す。 CRLF を正規化しない。

返り値

指定したメッセージの本文を文字列で返します。

---

imap_bodystruct

(PHP 4, PHP 5)

imap_bodystruct — 指定したメッセージの指定した body セクションの構造を読み込む

説明

object **imap_bodystruct** (resource $imap_stream , int $msg_number , string $section)

指定したメッセージの指定した body セクションの構造を読み込みます。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

section

読み込む body セクション。

返り値

オブジェクトの情報を返します。オブジェクトの構造やプロパティについての詳細は imap_fetchstructure() を参照ください。

参考

- imap_fetchstructure()

---

imap_check

(PHP 4, PHP 5)

imap_check — 現在のメールボックスをチェックする

説明

object **imap_check** (resource $imap_stream)

現在のメールボックスに関する情報を調べます。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

返り値

以下のプロパティをもつオブジェクトの情報を返します。

- Date - 現在のシステム時刻を » RFC2822 にしたがってフォーマットしたもの。
- Driver - メールボックスにアクセスする際に使用するプロトコル: POP3、IMAP、NNTP
- Mailbox - メールボックスの名前。
- Nmsgs - メールボックス内のメッセージの数。
- Recent - メールボックス内の新規メッセージの数。

失敗した場合には **FALSE** を返します。

例

Example#1 imap_check() の例

```
<?php

$imap_obj = imap_check($imap_stream);
var_dump($imap_obj);

?>
```

上の例の出力は、たとえば以下のようになります。

```
object(stdClass)(5) {
   ["Date"]=>
   string(37) "Wed, 10 Dec 2003 17:56:54 +0100 (CET)"
   ["Driver"]=>
   string(4) "imap"
   ["Mailbox"]=>
   string(54)
     "{www.example.com:143/imap/user="foo@example.com"}INBOX"
   ["Nmsgs"]=>
   int(1)
   ["Recent"]=>
   int(0)
}
```

---

imap_clearflag_full

(PHP 4, PHP 5)

imap_clearflag_full — メッセージのフラグをクリアする

説明

bool **imap_clearflag_full** (resource $imap_stream , string $sequence , string $flag [, string $options])

この関数は、指定したシーケンス sequence のメッセージのフラグから、指定したフラグ flag を削除します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

sequence

メッセージ番号のシーケンス。 X,Y のようにメッセージを列挙したり、 X:Y のようにしてその間のすべてのメッセージを指定したりできます。

flag

設定可能なフラグは "\\Seen"、 "\\Answered"、"\\Flagged"、"\\Deleted" および "\\Draft" です (» RFC2060 で定義されています)。

options

オプション options はビットマスクであり、 以下の組み合わせとなります。

- **ST_UID** - シーケンス引数はシーケンス番号の代わりに UID を含みます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_setflag_full()

---

imap_close

(PHP 4, PHP 5)

imap_close — IMAP ストリームをクローズする

説明

bool **imap_close** (resource $imap_stream [, int $flag])

IMAP ストリームをクローズします。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

flag

CL_EXPUNGE を指定した場合、メールボックスを閉じる前に暗黙のうちに 削除マークがついた全てのメッセージを削除します。 imap_expunge() を使用して、同じことを行うこともできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_open()

---

imap_createmailbox

(PHP 4, PHP 5)

imap_createmailbox — 新しいメールボックスを作る

説明

bool **imap_createmailbox** (resource $imap_stream , string $mailbox)

imap_createmailbox() は mailbox で指定された新しいメールボックスを作成します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。 この名前に国際化文字を含む場合には、 imap_utf7_encode() でエンコードする必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imap_createmailbox() example

```
<?php
$mbox = imap_open("{imap.example.org}", "username", "password", OP_HALFOPEN)
      or die("接続できません: " . imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode("phpnewb&ouml;x");

$newname = $name1;

echo "Newname will be '$name1'<br />¥n";

// ここでは inbox フォルダ内に新しいメールボックス "phptestbox" を
// 作成し、その後でメールボックスの状態を確認し、最後にそれを削除して
// もとの状態に戻します。
```

```
if (@imap_createmailbox($mbox, imap_utf7_encode("{imap.example.org}INBOX.$newname"))) {
    $status = @imap_status($mbox, "{imap.example.org}INBOX.$newname", SA_ALL);
    if ($status) {
        echo "your new mailbox '$name1' has the following status:<br />¥n";
        echo "Messages:    " . $status->messages    . "<br />¥n";
        echo "Recent:      " . $status->recent      . "<br />¥n";
        echo "Unseen:      " . $status->unseen      . "<br />¥n";
        echo "UIDnext:     " . $status->uidnext     . "<br />¥n";
        echo "UIDvalidity:" . $status->uidvalidity . "<br />¥n";

        if (imap_renamemailbox($mbox, "{imap.example.org}INBOX.$newname", "{imap.example.org}INBOX.$name2")) {
            echo "renamed new mailbox from '$name1' to '$name2'<br />¥n";
            $newname = $name2;
        } else {
            echo "imap_renamemailbox on new mailbox failed: " . imap_last_error() . "<br />¥n";
        }
    } else {
        echo "imap_status on new mailbox failed: " . imap_last_error() . "<br />¥n";
    }

    if (@imap_deletemailbox($mbox, "{imap.example.org}INBOX.$newname")) {
        echo "new mailbox removed to restore initial state<br />¥n";
    } else {
        echo "imap_deletemailbox on new mailbox failed: " . implode("<br />¥n", imap_errors()) . "<br />¥n";
    }

} else {
    echo "could not create new mailbox: " . implode("<br />¥n", imap_errors()) . "<br />¥n";
}

imap_close($mbox);
?>
```

参考

- imap_renamemailbox()
- imap_deletemailbox()

---

imap_delete

(PHP 4, PHP 5)

imap_delete — 現在のメールボックスから削除するメッセージに印を付ける

説明

bool **imap_delete** (int $imap_stream , int $msg_number [, int $options])

msg_number が指すメッセージに 削除予定のマークをします。削除マークを付けられたメッセージは、 imap_expunge() がコールされるか imap_close() に **CL_EXPUNGE** を付けてコールされるかのどちらかが行われるまでメールボックスに残ったままになります。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

options

FT_UID を指定すると、 引数 msg_number を UID として処理することを関数に指示できます。

返り値

Returns **TRUE**.

例

Example#1 imap_delete() の例

```
<?php

$mbox = imap_open("{imap.example.org}INBOX", "username", "password")
    or die("接続できません: " . imap_last_error());

$check = imap_mailboxmsginfo($mbox);
echo "Messages before delete: " . $check->Nmsgs . "<br />¥n";

imap_delete($mbox, 1);

$check = imap_mailboxmsginfo($mbox);
echo "Messages after  delete: " . $check->Nmsgs . "<br />¥n";

imap_expunge($mbox);

$check = imap_mailboxmsginfo($mbox);
echo "Messages after expunge: " . $check->Nmsgs . "<br />¥n";

imap_close($mbox);
```

```
?>
```

注意

注意: POP3 メールボックスは、コネクション間で保存可能なメッセージフラグを持っていません。 そのため、削除マークをつけたメッセージを本当に削除するためには、 同一の接続内で imap_expunge() をコールする必要があります。

参考

- imap_undelete()
- imap_expunge()
- imap_close()

---

imap_deletemailbox

(PHP 4, PHP 5)

imap_deletemailbox — メールボックスを削除する

説明

bool **imap_deletemailbox** (resource $imap_stream , string $mailbox)

指定したメールボックス mailbox を削除します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_createmailbox()
- imap_renamemailbox()
- mbox の書式については imap_open()

---

imap_errors

(PHP 4, PHP 5)

imap_errors — 発生したすべての IMAP エラーを返す

説明

array **imap_errors** (void)

ページのリクエストの間かエラースタックがリセットされて以来 生じた全ての IMAP エラー（存在すれば）を返します。

imap_errors() がコールされると、 処理の終了後にエラースタックがクリアされます。

返り値

この関数は、最後に **imap_errors()** コールを行ってから、またはそのページの処理を開始してから 発生した全ての IMAP エラーメッセージの配列を返します。 エラーメッセージが存在しない場合には **FALSE** を返します。

参考

- imap_last_error()
- imap_alerts()

---

imap_expunge

(PHP 4, PHP 5)

imap_expunge — 削除用にマークされたすべてのメッセージを削除する

説明

bool **imap_expunge** (resource $imap_stream)

imap_delete()、imap_mail_move()、 あるいは imap_setflag_full() で削除用マークを設定されたすべてのメッセージを削除します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

返り値

TRUE を返します。

---

imap_fetch_overview

(PHP 4, PHP 5)

imap_fetch_overview — 指定したメッセージのヘッダ情報の概要を読む

説明

array **imap_fetch_overview** (resource $imap_stream , string $sequence [, int $options])

この関数は、指定された sequence のメールヘッダを取得してその内容の概要を返します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

sequence

メッセージのシーケンスを指定します。 X,Y 形式で列挙したり、 X:Y 形式でその範囲内のすべてのメッセージを取得したりできます。

options

sequence には、メッセージのインデックスか、もし flags に FT_UID が含まれている場合は UID が含まれます。

返り値

オブジェクトの配列を返します。各要素が、 それぞれひとつのメッセージのヘッダを表します。 オブジェクトで定義されるのは、存在するプロパティのみです。 以下のプロパティがあります。

- subject - メッセージの題名(subject)
- from - 送信者
- to - 受信者
- date - 送信日
- message_id - Message-ID
- references - このメッセージ ID への参照です
- in_reply_to - このメッセージ ID への返信です
- size - サイズ（バイト数）
- uid - メールボックスにおけるこのメッセージの UID
- msgno - メールボックスにおけるこのメッセージのシーケンス番号
- recent - このメッセージには recent フラグが立てられています
- flagged - フラグが立てられています
- answered - 返信済みフラグが立てられています
- deleted - 削除フラグが立てられています
- seen - 既読フラグが立てられています
- draft - 草稿フラグが立てられています

例

Example#1 imap_fetch_overview() の例

```
<?php
$mbox = imap_open("{imap.example.org:143}INBOX", "username", "password")
      or die("接続できません: " . imap_last_error());

$MC = imap_check($mbox);

// INBOX のすべてのメッセージの概要を取得します
$result = imap_fetch_overview($mbox,"1:{$MC->Nmsgs}",0);
foreach ($result as $overview) {
     echo "#{$overview->msgno} ({$overview->date}) - From: {$overview->from}
       {$overview->subject}¥n";
}
imap_close($mbox);
?>
```

参考

- imap_fetchheader()

---

imap_fetchbody

(PHP 4, PHP 5)

imap_fetchbody — メッセージ本文中の特定のセクションを取り出す

説明

string **imap_fetchbody** (resource $imap_stream , int $msg_number , string $part_number [, int $options])

指定されたメッセージ本文中の特定のセクションを取得します。 本文パートは、この関数ではデコードされません。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

part_number

パート番号。ピリオドで区切られた整数文字列を指定します。 これは、IMAP4 仕様における本文パートのリストへのインデックスとなります。

options

ビットマスクであり、以下の組合わせとなります。

- **FT_UID** - msg_number は UID である
- **FT_PEEK** - 既に設定されていない場合、 \Seen フラグを設定しない
- **FT_INTERNAL** - 内部フォーマットで文字列を返す。CRLF は正規化しない。

返り値

指定されたメッセージ本文中の特定のセクションをテキスト文字列で返します。

参考

- imap_savebody()
- imap_fetchstructure()

---

imap_fetchheader

(PHP 4, PHP 5)

imap_fetchheader — メッセージのヘッダを返す

説明

string **imap_fetchheader** (resource $imap_stream , int $msg_number [, int $options])

指定したメッセージについて、フィルタリング されていない完全な » RFC2822 フォーマットのヘッダをテキスト文字列として取得します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

options

オプション options は次のようになります。

- **FT_UID** - 引数 msgno は UID です。
- **FT_INTERNAL** - 返される文字列は "internal" フォーマットです。CRLF は正規化しません。
- **FT_PREFETCHTEXT** - RFC822.TEXT を、 同時に事前に取得しておく必要があります。これは、メッセージテキスト 全体を取得したい場合（例:「ローカルファイルに保存する」操作）に IMAP 接続で余分な RTT を回避します。

返り値

指定したメッセージのヘッダをテキスト文字列で返します。

参考

- imap_fetch_overview()

---

imap_fetchstructure

(PHP 4, PHP 5)

imap_fetchstructure — 特定のメッセージの構造を読み込む

説明

object **imap_fetchstructure** (resource $imap_stream , int $msg_number [, int $options])

この関数は、指定したメッセージに関するすべての構造化された情報を取り出します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

options

オプションのパラメータで、FT_UID のみが指定可能です。 これは、msg_number を UID として処理することを関数に指定するためのものです。

返り値

オブジェクトを返します。このオブジェクトには、 MIME の添付の各要素に類似のオブジェクトとしてエンベロープ、 内部の日付、サイズ、フラグそして本体が含まれます。 返されるオブジェクトの構造は次のようになります。

imap_fetchstructure() が返すオブジェクト

type	最初の body 部の型
encoding	body 部を転送する際のエンコード法
ifsubtype	subtype 文字列がある場合に **TRUE**
subtype	MIME の subtype
ifdescription	description 文字列がある場合に **TRUE**
description	内容を記述する文字列
ifid	identification 文字列がある場合に **TRUE**
id	Identification 文字列
lines	行数
bytes	バイト数
ifdisposition	disposition 文字列がある場合に **TRUE**
disposition	Disposition 文字列
ifdparameters	dparameters 配列が存在する場合に **TRUE**
dparameters	オブジェクトの配列。各オブジェクトは "attribute" および "value" というプロパティを保持し、それぞれ Content-disposition MIME ヘッダの 対応するパラメータを表す。
ifparameters	配列 parameters が存在する場合に **TRUE**
parameters	オブジェクトの配列。各オブジェクトは "attribute" および "value" というプロパティを保持する。
parts	オブジェクトの配列であり、その構造はトップレベルオブジェクトと 同じです。それぞれが MIME body 部に対応しています。

最初の body 部の型

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

転送時のエンコーディング

0	7BIT

1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

参考

- imap_fetchbody()
- imap_bodystruct()

---

imap_get_quota

(PHP 4 >= 4.0.5, PHP 5)

imap_get_quota — クオータレベルの設定、メールボックス毎の使用状況を取得する

説明

array **imap_get_quota** (resource $imap_stream , string $quota_root)

クオータレベルの設定、メールボックス毎の使用状況を取得します。

この関数の非管理者向けバージョンは、 PHP の imap_get_quotaroot() 関数を参照ください。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

quota_root

quota_root は、通常 user.name という形式にする必要があります。 name は、情報を取得したいメールボックスの名前です。

返り値

指定したメールボックスの limit と usage をキーとした整数値を配列として返します。 limit の値は、このメールボックスで最大使用可能な大きさを表します。 usage の値は、このメールボックスの現在の使用状況を示します。 失敗した場合に **FALSE** を返します。

PHP 4.3 では、この関数は » RFC2087 で述べられている機能をより適切に反映するようになっています。 返り値の配列からは、サポートするリソース （例:メッセージ、あるいはサブフォルダ）数の制限をなくし、 名前つきリソースを独立した配列のキーとして受信するようにしました。 各キーの値は配列となっており、その中に usage と values の値が格納されています。

過去との互換性を保つため、旧来のアクセス方法も使用可能です。 しかし新しい方法に変更することを推奨します。

例

Example#1 imap_get_quota() の例

```php
<?php
$mbox = imap_open("{imap.example.org}", "mailadmin", "password", OP_HALFOPEN)
      or die("接続できません: " . imap_last_error());

$quota_value = imap_get_quota($mbox, "user.kalowsky");
if (is_array($quota_value)) {
     echo "Usage level is: " . $quota_value['usage'];
     echo "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
?>
```

Example#2 4.3 以降のバージョンでの imap_get_quota() の例

```php
<?php
$mbox = imap_open("{imap.example.org}", "mailadmin", "password", OP_HALFOPEN)
      or die("接続できません: " . imap_last_error());

$quota_values = imap_get_quota($mbox, "user.kalowsky");
if (is_array($quota_values)) {
   $storage = $quota_values['STORAGE'];
    echo "STORAGE usage level is: " . $storage['usage'];
    echo "STORAGE limit level is: " . $storage['limit'];

   $message = $quota_values['MESSAGE'];
    echo "MESSAGE usage level is: " . $message['usage'];
    echo "MESSAGE limit is: " . $message['limit'];

   /* ... */
}

imap_close($mbox);
?>
```

注意

この関数は、現在、c-client2000 以降のライブラリを使用しているユーザのみ使用可能です。

指定する imap_stream は、 メールの管理者としてオープンしたものである必要があります。 そうでない場合は、この関数は失敗します。

参考

- imap_open()
- imap_set_quota()
- imap_get_quotaroot()

---

imap_get_quotaroot

(PHP 4 >= 4.3.0, PHP 5)

imap_get_quotaroot — ユーザ単位のクォータ設定を取得する

説明

array **imap_get_quotaroot** (resource $imap_stream , string $quota_root)

ユーザ単位のクォータ設定を取得します。 limit の値は、このユーザがメールボックスで使用可能な総容量を表します。 usage の値は、ユーザが現在メールボックスで使用している容量を表します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

quota_root

quota_root はどのメールボックスを調べるかを 指定します（例: INBOX）。

返り値

指定したユーザのメールボックスに関連する整数値を配列で返します。 すべての値にはリソース名に基づいたキーがつけられており、 usage および limit 値を保持する配列が関連付けられています。

コールが失敗した場合、およびサーバからの応答内容をパースできなかった場合には この関数は **FALSE** を返します。

例

Example#1 imap_get_quotaroot() の例

```
<?php
$mbox = imap_open("{imap.example.org}", "kalowsky", "password", OP_HALFOPEN)
      or die("接続できません: " . imap_last_error());

$quota = imap_get_quotaroot($mbox, "INBOX");
if (is_array($quota)) {
   $storage = $quota_values['STORAGE'];
   echo "STORAGE usage level is: " . $storage['usage'];
   echo "STORAGE limit level is: " . $storage['limit'];

   $message = $quota_values['MESSAGE'];
   echo "MESSAGE usage level is: " . $message['usage'];
   echo "MESSAGE limit level is: " . $message['limit'];

   /* ... */

}

imap_close($mbox);
?>
```

注意

この関数は、現在は c-client2000 以降のライブラリを使用しているユーザのみ利用可能です。

imap_stream は、 調べたいメールボックスを所有するユーザがオープンしなければなりません。

参考

- imap_open()
- imap_set_quota()
- imap_get_quota()

---

imap_getacl

(PHP 5)

imap_getacl — 与えられたメールボックスの ACL を取得する

説明

array **imap_getacl** (resource $imap_stream , string $mailbox)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

注意

この関数は、現在は c-client2000 以降のライブラリを使用しているユーザのみ利用可能です。

参考

- imap_setacl()

---

imap_getmailboxes

(PHP 4, PHP 5)

imap_getmailboxes — メールボックスのリストを読み込み、各ボックスに関する詳細な情報を返す

説明

array **imap_getmailboxes** (resource $imap_stream , string $ref , string $pattern)

メールボックスの情報を取得します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

ref

通常、ref は imap_open() で述べられているサーバ定義です。

pattern

検索を開始するメールボックスの階層を指定します。

pattern の中で使用できる特別な文字として '*' および '%' があります。'*' は、全てのメールボックスを意味します。 pattern に '*' を指定した場合、メールボックス 階層全体のリストが得られます。'%' は現在のレベルのみを意味します。 '%' を pattern に指定した場合、 トップレベルのメールボックスのみが返されます。UW_IMAPD を使用した場合、 '~/mail/%' は ~/mail ディレクトリの全てのメールボックスを返しますが、 そのディレクトリのサブフォルダーにあるメールボックスは返しません。

返り値

メールボックス情報を有するオブジェクトの配列を返します。各オブジェクトには、 メールボックスの完全な名前である name 、 このメールボックスの階層の区切りを示す属性 delimiter 、 そして attributes が存在します。 attributes はビットマスクであり、 次のものについて調べることができます。

- **LATT_NOINFERIORS** - このメールボックスには 「子供」はいません（このボックスの中にメールボックスはありません）。
- **LATT_NOSELECT** - 単なるコンテナであり、 メールボックスではありません - これをオープンすることはできません。
- **LATT_MARKED** - このメールボックスにはマークが つけられています。UW-IMAPD でのみ使用されます。
- **LATT_UNMARKED** - このメールボックスにはマークが つけられていません。UW-IMAPD でのみ使用されます。

例

Example#1 imap_getmailboxes() の例

```
<?php
$mbox = imap_open("{imap.example.org}", "username", "password", OP_HALFOPEN)
       or die("接続できません: " . imap_last_error());

$list = imap_getmailboxes($mbox, "{imap.example.org}", "*");
if (is_array($list)) {
     foreach ($list as $key => $val) {
         echo "($key) ";
         echo imap_utf7_decode($val->name) . ",";
         echo "'" . $val->delimiter . "',";
         echo $val->attributes . "<br />¥n";
     }
} else {
     echo "imap_getmailboxes failed: " . imap_last_error() . "¥n";
}

imap_close($mbox);
```

```
?>
```

参考

- imap_getsubscribed()

---

imap_getsubscribed

(PHP 4, PHP 5)

imap_getsubscribed — 購読中の全メールボックスの一覧を取得する

説明

array **imap_getsubscribed** (resource $imap_stream , string $ref , string $pattern)

購読中のメールボックスの情報を取得します。

imap_getmailboxes() と同じ動作をしますが、 ユーザが購読しているメールボックスのみを返すという点が異なります。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

ref

ref は、通常は imap_open() で指定したサーバ情報です。

pattern

検索を開始するメールボックスの階層を指定します。

pattern の中で使用できる特別な文字として '*' および '%' があります。'*' は、全てのメールボックスを意味します。 pattern に '*' を指定した場合、メールボックス 階層全体のリストが得られます。'%' は現在のレベルのみを意味します。 '%' を pattern に指定した場合、 トップレベルのメールボックスのみが返されます。UW_IMAPD を使用した場合、 '~/mail/%' は ~/mail ディレクトリの全てのメールボックスを返しますが、 そのディレクトリのサブフォルダーにあるメールボックスは返しません。

返り値

メールボックス情報を有するオブジェクトの配列を返します。各オブジェクトには、 メールボックスの完全な名前である name 、 このメールボックスの階層の区切りを示す属性 delimiter 、 そして attributes が存在します。 attributes はビットマスクであり、 次のものについて調べることができます。

- **LATT_NOINFERIORS** - このメールボックスには 「子供」はいません（このボックスの中にメールボックスはありません）。
- **LATT_NOSELECT** - 単なるコンテナであり、 メールボックスではありません - これをオープンすることはできません。
- **LATT_MARKED** - このメールボックスにはマークが つけられています。UW-IMAPD でのみ使用されます。
- **LATT_UNMARKED** - このメールボックスにはマークが つけられていません。UW-IMAPD でのみ使用されます。

---

imap_header

(PHP 4, PHP 5)

imap_header — imap_headerinfo() のエイリアス

説明

この関数は次の関数のエイリアスです。 imap_headerinfo().

---

imap_headerinfo

(PHP 4, PHP 5)

imap_headerinfo — メッセージヘッダを読み込む

説明

object **imap_headerinfo** (resource $imap_stream , int $msg_number [, int $fromlength [, int $subjectlength [, string $defaulthost]]])

指定したメッセージ番号についての情報を、そのヘッダを読み込んで取得します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

fromlength

fetchfrom プロパティの文字数。 ゼロ以上でなければなりません。

subjectlength

fetchsubject プロパティの文字数。 ゼロ以上でなければなりません。

defaulthost

返り値

以下のプロパティをもつオブジェクトを返します。

- *toaddress* - 完全な *to:* 行。最大 *1024* 文字。
- *to* - *To:* 行から、次のプロパティを含むオブジェクトの配列を返します。 *personal*、*adl*、 *mailbox* および *host*
- *fromaddress* - 完全な *from:* 行。最大 *1024* 文字。
- *from* - *From:* 行から、次のプロパティを含むオブジェクトの配列を返します。 *personal*、*adl*、 *mailbox* および *host*
- *ccaddress* - 完全な *cc:* 行。最大 *1024* 文字。
- *cc* - *Cc:* 行から、次のプロパティを含むオブジェクトの配列を返します。 *personal*、*adl*、 *mailbox* および *host*
- *bccaddress* - 完全な *bcc:* 行。最大 *1024* 文字。
- *bcc* - *Bcc:* 行から、次のプロパティを含むオブジェクトの配列を返します。 *personal*、*adl*、 *mailbox* および *host*
- *reply_toaddress* - 完全な *Reply-To:* 行。最大 *1024* 文字。
- *reply_to* - *Reply-To:* 行から、次のプロパティを含むオブジェクトの配列を返します。 *personal*、*adl*、 *mailbox* および *host*
- *senderaddress* - 完全な *sender:* 行。最大 *1024* 文字。
- *sender* - *Sender:* 行から、次のプロパティを含むオブジェクトの配列を返します。 *personal*、*adl*、 *mailbox* および *host*
- *return_pathaddress* - 完全な *Return-Path:* 行。最大 *1024* 文字。
- *return_path* - *Return-Path:* 行から、次のプロパティを含むオブジェクトの配列を返します。 *personal*、 *adl*、*mailbox* および *host*
- *remail* -
- *date* - ヘッダにあるメッセージの日付。
- *Date* - *date* と同じ。
- *subject* - メッセージの件名。
- *Subject* - *subject* と同じ。
- *in_reply_to* -
- *message_id* -
- *newsgroups* -
- *followup_to* -
- *references* -
- *Recent* - 最近の既読メッセージは *R*、 最新の未読メッセージは *N*、 最新でないメッセージは *' '*。
- *Unseen* - 最新でない未読メッセージは *U*、 既読、あるいは最新の未読メッセージは *' '*。
- *Flagged* - フラグがたっている場合は *F*、 そうでない場合は *' '*。
- *Answered* - 返信した場合は *A*、 していない場合は *' '*。
- *Deleted* - 削除された場合は *D*、 されていない場合は *' '*。
- *Draft* - 草稿である場合は *X*、 そうでない場合は *' '*。
- *Msgno* - メッセージ番号。
- *MailDate* -
- *Size* - メッセージのサイズ。
- *udate* - メールメッセージの日付を *Unix time* で表したもの。
- *fetchfrom* - *from* 行を *fromlength* 文字に適合させたもの。
- *fetchsubject* - *subject* 行を *subjectlength* 文字に適合させたもの。

参考

- *imap_fetch_overview()*

---

imap_headers

(PHP 4, PHP 5)

imap_headers — メールボックス内のすべてのメッセージのヘッダを返す

説明

array ***imap_headers*** *(resource $imap_stream)*

メールボックス内のすべてのメッセージのヘッダを返します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

返り値

ヘッダ情報でフォーマットされた文字列の配列を返します。1 つの メールメッセージ毎に 1 つの要素が格納されます。

---

imap_last_error

(PHP 4, PHP 5)

imap_last_error — ページリクエスト時に生じた直近の IMAP エラーを返す

説明

string **imap_last_error** (void)

現在のページに生じた直近の IMAP エラーメッセージの全文を返します。 エラースタックは変更されません。 **imap_last_error()** を続けてコールした際、 コール間に新規エラーが生じていない場合は同じエラーが返されます。

返り値

現在のページに生じた直近の IMAP エラーメッセージの全文を返します。 エラーメッセージが存在しない場合は **FALSE** を返します。

参考

- imap_errors()

---

imap_list

(PHP 4, PHP 5)

imap_list — メールボックスのリストを読み込む

説明

array **imap_list** (resource $imap_stream , string $ref , string $pattern)

メールボックスのリストを読み込みます。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

ref

通常、ref は imap_open() で述べられているサーバ定義です。

pattern

検索を開始するメールボックスの階層を指定します。

pattern の中で使用できる特別な文字として '*' および '%' があります。'*' は、全てのメールボックスを意味します。 pattern に '*' を指定した場合、メールボックス 階層全体のリストが得られます。'%' は現在のレベルのみを意味します。 '%' を pattern に指定した場合、 トップレベルのメールボックスのみが返されます。UW_IMAPD を使用した場合、 '~/mail/%' は ~/mail ディレクトリの全てのメールボックスを返しますが、 そのディレクトリのサブフォルダーにあるメールボックスは返しません。

返り値

メールボックスの名前を含む配列を返します。

例

Example#1 imap_list() の例

```
<?php
$mbox = imap_open("{imap.example.org}", "username", "password", OP_HALFOPEN)
       or die("接続できません: " . imap_last_error());

$list = imap_list($mbox, "{imap.example.org}", "*");
if (is_array($list)) {
     foreach ($list as $val) {
          echo imap_utf7_decode($val) . "¥n";
     }
} else {
     echo "imap_list が失敗しました: " . imap_last_error() . "¥n";
}

imap_close($mbox);
?>
```

参考

- imap_getmailboxes()

- imap_lsub()

---

imap_listmailbox

(PHP 4, PHP 5)

imap_listmailbox — imap_list() のエイリアス

説明

この関数は次の関数のエイリアスです。 imap_list().

---

imap_listscan

(No version information available, might be only in CVS)

imap_listscan — 指定したテキストにマッチするメールボックスの一覧を返す

説明

array **imap_listscan** (resource $imap_stream , string $ref , string $pattern , string $content)

content をテキストに持つメールボックスの名前を 配列で返します。

この関数は imap_listmailbox() に似ていますが、 メールボックス内に文字列 content が存在するか どうかを調べる点が違います。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

ref

通常、ref は imap_open() で述べられているサーバ定義です。

pattern

検索を開始するメールボックスの階層を指定します。

pattern の中で使用できる特別な文字として '*' および '%' があります。'*' は、全てのメールボックスを意味します。 pattern に '*' を指定した場合、メールボックス 階層全体のリストが得られます。'%' は現在のレベルのみを意味します。 '%' を pattern に指定した場合、 トップレベルのメールボックスのみが返されます。UW_IMAPD を使用した場合、 '~/mail/%' は ~/mail ディレクトリの全てのメールボックスを返しますが、 そのディレクトリのサブフォルダーにあるメールボックスは返しません。

content

検索する文字列。

返り値

content をテキストに持つメールボックスの名前を 配列で返します。

参考

- imap_listmailbox()
- imap_search()

---

imap_listsubscribed

(PHP 4, PHP 5)

imap_listsubscribed — imap_lsub() のエイリアス

説明

この関数は次の関数のエイリアスです。 imap_lsub().

---

imap_lsub

(PHP 4, PHP 5)

imap_lsub — 購読しているすべてのメールボックスの一覧を得る

説明

array **imap_lsub** (resource $imap_stream , string $ref , string $pattern)

購読しているすべてのメールボックスの配列を取得します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

ref

通常、ref は imap_open() で述べられているサーバ定義です。

pattern

検索を開始するメールボックスの階層を指定します。

pattern の中で使用できる特別な文字として '*' および '%' があります。'*' は、全てのメールボックスを意味します。 pattern に '*' を指定した場合、メールボックス 階層全体のリストが得られます。'%' は現在のレベルのみを意味します。 '%' を pattern に指定した場合、 トップレベルのメールボックスのみが返されます。UW_IMAPD を使用した場合、 '~/mail/%' は ~/mail ディレクトリの全てのメールボックスを返しますが、 そのディレクトリのサブフォルダーにあるメールボックスは返しません。

返り値

購読しているすべてのメールボックスの配列を返します。

参考

- imap_list()
- imap_getmailboxes()

---

imap_mail_compose

(PHP 4, PHP 5)

imap_mail_compose — 指定したエンベロープおよびボディセクションに基づいて MIME メッセージを作成する

説明

string **imap_mail_compose** (array $envelope , array $body)

指定したエンベロープ envelope およびボディセクション body に基づいて MIME メッセージを作成します。

パラメータ

envelope

ヘッダフィールドの連想配列。

body

本文の配列。

本文は、以下のキー "type"、"encoding"、"subtype"、"description" および "contents.data" からなる連想配列です。

返り値

MIME メッセージを返します。

例

Example#1 imap_mail_compose() の例

```php
<?php

$envelope["from"]= "joe@example.com";
$envelope["to"]  = "foo@example.com";
$envelope["cc"]  = "bar@example.com";

$part1["type"] = TYPEMULTIPART;
$part1["subtype"] = "mixed";

$filename = "/tmp/imap.c.gz";
$fp = fopen($filename, "r");
$contents = fread($fp, filesize($filename));
fclose($fp);

$part2["type"] = TYPEAPPLICATION;
$part2["encoding"] = ENCBINARY;
$part2["subtype"] = "octet-stream";
$part2["description"] = basename($filename);
$part2["contents.data"] = $contents;

$part3["type"] = TYPETEXT;
$part3["subtype"] = "plain";
$part3["description"] = "description3";
$part3["contents.data"] = "contents.data3¥n¥n¥n¥t";

$body[1] = $part1;
$body[2] = $part2;
$body[3] = $part3;
```

```
echo nl2br(imap_mail_compose($envelope, $body));
?>
```

---

imap_mail_copy

(PHP 4, PHP 5)

imap_mail_copy — 指定されたメッセージをメールボックスにコピーする

説明

bool **imap_mail_copy** (resource $imap_stream , string $msglist , string $mailbox [, int $options])

msglist で指定されたメッセージを、 指定したメールボックスにコピーします。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msglist

msglist は、 (» RFC2060 に記述されたように) ただのメッセージ番号ではなく、範囲を示します。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

options

options はビットマスクであり、以下の組み合わせです。

- **CP_UID** - UIDS を含む処理の数
- **CP_MOVE** - コピー後にメールボックスからメッセージを削除する

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_mail_move()

---

imap_mail_move

(PHP 4, PHP 5)

imap_mail_move — 指定されたメッセージをメールボックスに移動する

説明

bool **imap_mail_move** (resource $imap_stream , string $msglist , string $mailbox [, int $options])

msglist で指定されたメッセージを、 指定されたメールボックス mbox に移動します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msglist

msglist は、 (» RFC2060 に記述されたように) ただのメッセージ番号ではなく、範囲を示します。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

options

options はビットマスクであり、ひとつのオプションをとります。

- **CP_UID** - UIDS を含む処理の数

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_mail_copy()

imap_mail

(PHP 4, PHP 5)

imap_mail — e-mail メッセージを送信する

説明

bool **imap_mail** (string $to , string $subject , string $message [, string $additional_headers [, string $cc [, string $bcc [, string $rpath]]]])

この関数は、Cc および Bcc 受信者の正確な処理を行って email を送信することが可能です。

パラメータ to 、cc および bcc は全て文字列で、 » RFC822 アドレスリストとしてパースされます。

パラメータ

to

受信者。

subject

メールの件名。

message

メールの本文。

additional_headers

メールに設定する追加ヘッダ文字列。

cc

bcc

bcc で指定した受信者は mail を受信しますが、 ヘッダからは除外されます。

rpath

リターンパスを指定するために rpath パラメータを使用してください。この関数は、 複数のユーザ用のメールクライアントとして PHP を使用する際に有用です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mail()

---

imap_mailboxmsginfo

(PHP 4, PHP 5)

imap_mailboxmsginfo — 現在のメールボックスに関する情報を得る

説明

object **imap_mailboxmsginfo** (resource $imap_stream)

サーバにおける現在のメールボックスのステータスを調べます。この関数は imap_status() に似ていますが、 メールボックス内の全メッセージのサイズを合計します。このため、 実行時間が幾分余計にかかります。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

返り値

以下のプロパティを有するオブジェクトを返します。

Mailbox のプロパティ

Date	最終変更日
Driver	ドライバ
Mailbox	メールボックスの名前
Nmsgs	メッセージ数
Recent	最近のメッセージの数

Unread	未読のメッセージの数
Deleted	削除されたメッセージの数
Size	メールボックスのサイズ

Returns ***FALSE*** *on failure.*

例

Example#1 imap_mailboxmsginfo() example

```
<?php

$mbox = imap_open("{imap.example.org}INBOX", "username", "password")
      or die("接続できません: " . imap_last_error());

$check = imap_mailboxmsginfo($mbox);

if ($check) {
    echo "Date: "      . $check->Date     . "<br />¥n" ;
    echo "Driver: "    . $check->Driver   . "<br />¥n" ;
    echo "Mailbox: "   . $check->Mailbox  . "<br />¥n" ;
    echo "Messages: "  . $check->Nmsgs    . "<br />¥n" ;
    echo "Recent: "    . $check->Recent   . "<br />¥n" ;
    echo "Unread: "    . $check->Unread   . "<br />¥n" ;
    echo "Deleted: "   . $check->Deleted  . "<br />¥n" ;
    echo "Size: "      . $check->Size     . "<br />¥n" ;
} else {
    echo "imap_check() に失敗: " . imap_last_error() . "<br />¥n";
}

imap_close($mbox);

?>
```

---

imap_mime_header_decode

(PHP 4, PHP 5)

imap_mime_header_decode — MIME ヘッダ要素をデコードする

説明

array ***imap_mime_header_decode*** *(string $text)*

非 ASCII テキストの MIME メッセージヘッダエクステンションをデコードします (» RFC2047 を参照ください)。

パラメータ

text

MIME テキスト。

返り値

デコードされた要素は、オブジェクトの配列で返されます。 各オブジェクトは、二つのプロパティ charset と text を有しています。

要素がエンコードされていない場合、言い替えるとプレーンな US-ASCII の場合は この要素の charset プロパティは default に設定されます。

例

Example#1 imap_mime_header_decode() の例

```
<?php
$text = "=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?=  <keld@example.com>";

$elements = imap_mime_header_decode($text);
for ($i=0; $i<count($elements); $i++) {
    echo "Charset:  {$elements[$i]->charset}¥n";
    echo "Text:  {$elements[$i]->text}¥n¥n";
}
?>
```

上の例の出力は以下となります。

```
Charset:  ISO-8859-1
Text: Keld Jørn Simonsen

Charset:  default
Text:   <keld@example.com>
```

上の例には二つの要素があります。最初の要素は ISO-8859-1、 2 番目の要素はプレーンな US-ASCII で事前にエンコードされています。

参考

- imap_utf8()

---

imap_msgno

(PHP 4, PHP 5)

imap_msgno — 指定した UID のメッセージ番号を返す

説明

int **imap_msgno** (resource $imap_stream , int $uid)

指定した uid のメッセージ番号を返します。

この関数は、imap_uid() の逆の動作となります。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

uid

メッセージの UID。

返り値

指定した uid のメッセージ番号を返します。

参考

- imap_uid()

---

imap_num_msg

(PHP 4, PHP 5)

imap_num_msg — 現在のメールボックスのメッセージ数を取得する

説明

int **imap_num_msg** (resource $imap_stream)

現在のメールボックスにあるメッセージ数を返します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

返り値

現在のメールボックスのメッセージ数を返します。

参考

- imap_num_recent()
- imap_status()

---

imap_num_recent

(PHP 4, PHP 5)

imap_num_recent — 現在のメールボックスにある新規メッセージ数を取得する

説明

int **imap_num_recent** (resource $imap_stream)

現在のメールボックスにある新しいメッセージの数を返します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

返り値

現在のメールボックスにある新しいメッセージの数を返します。

参考

- imap_num_msg()
- imap_status()

---

imap_open

(PHP 4, PHP 5)

imap_open — メールボックスへの IMAP ストリームをオープンする

説明

resource **imap_open** (string $mailbox , string $username , string $password [, int $options [, int $n_retries]])

mailbox への IMAP ストリームをオープンします。

この関数は、POP3 や NNTP サーバへのストリームをオープンする際にも使用可能です。 しかし、いくつかの関数および機能は IMAP サーバでしか利用できません。

パラメータ

mailbox

メールボックス名(mailbox)は、サーバ名の部分と使用するサーバにおける メールボックスへのパスから構成されます。特別な名前 INBOX は、 カレントユーザの個人メールボックスを意味します。 ASCII 空間で出力可能な文字以外の外国文字を含むメールボックス名は imap_utf7_encode() でエンコードする必要があります。

サーバ部は '{' および '}' で括られ、サーバ名または IP アドレス、オプションの ':' から始まるポート指定子、('/' で始まる)オプションのプロトコル指定子 から構成されます。

サーバ部は、全ての mailbox パラメータで必須です。

{ で始まる名前はすべてリモート名で、 "{" remote_system_name [":" port] [flags] "}" [mailbox_name] のような形式となります。

- remote_system_name - Internet ドメイン名 あるいは括弧でかこまれたサーバの IP アドレス。
- port - オプションの TCP ポート番号。デフォルトは そのサービスのデフォルトポート。
- flags - オプションのフラグ。以下の表を参照ください。
- mailbox_name - リモートメールボックス名。デフォルトは INBOX 。

オプションのフラグ名

フラグ	説明
/service=service	メールボックスにアクセスするサービス。デフォルトは "imap" 。
/user=user	サーバへのログイン時のユーザ名。
/authuser=user	リモートの認証ユーザ。指定されていた場合は、このユーザのパスワードが 認証に使用されます（例: administrator）。
/anonymous	匿名ユーザとしてアクセスします。
/debug	プロトコルの通信内容をアプリケーションのデバッグログに記録します。
/secure	ネットワーク越しにプレーンテキストのパスワードを送信しません。
/imap, /imap2, /imap2bis, /imap4, /imap4rev1	/service=imap と同じです。
/pop3	/service=pop3 と同じです。
/nntp	/service=nntp と同じです。
/norsh	事前に認証済みの IMAP セッションを確立する際に、rsh や ssh を使用しません。
/ssl	セッションの暗号化に SSL（Secure Socket Layer）を使用します。
/validate-cert	TLS/SSL サーバの証明書を検証します（デフォルトの挙動です）。
/novalidate-cert	TLS/SSL サーバの証明書を検証しません。サーバが自己証明の 証明書を使用している際に必要となります。
/tls	セッションの暗号化に start-TLS の使用を強制し、それを サポートしていないサーバとの接続を拒否します。
/notls	たとえサーバがそれをサポートしていたとしても、 セッションで start-TLS による暗号化を使用しません。
/readonly	読み込み専用でのメールボックスのオープンを要求します（IMAP のみ。 NNTP では無視され、SMTP および POP3 ではエラーとなります）。

username

ユーザ名。

password

username のパスワード。

options

options はビットマスクであり、以下の組み合わせとなります。

- **OP_READONLY** - メールボックスを読み込み専用でオープンします。
- **OP_ANONYMOUS** - news に関して .newsrc を使用せず、更新もしません（NNTP のみ）。
- **OP_HALFOPEN** - IMAP 及び NNTP 名について、 接続をオープンしますがメールボックスはオープンしません。

- **CL_EXPUNGE** - メールボックスを閉じる際に、 自動的にメールボックスを削除します (imap_delete() および imap_expunge() も参照ください)。
- **OP_DEBUG** - プロトコルネゴシエーションをデバッグします。
- **OP_SHORTCACHE** - 短い (elt-only) キャッシングを行います。
- **OP_SILENT** - イベントを受信しません (内部で使用します)。
- **OP_PROTOTYPE** - ドライバのプロトタイプを返します。
- **OP_SECURE** - セキュアでない認証を行いません。

n_retries

接続試行の最大数。

返り値

成功した場合は IMAP ストリームを、失敗した場合は **FALSE** を返します。

変更履歴

バージョン	説明
5.2.0	*n_retries* が追加されました。

例

Example#1 imap_open() のさまざまな使用例

```
<?php
// ローカルマシンのポート 143 で稼動している IMAP サーバに接続するには
// 以下のようにします。
$mbox = imap_open("{localhost:143}INBOX", "user_id", "password");

// ローカルマシンのポート 110 で稼動している POP3 サーバに接続するには、
$mbox = imap_open ("{localhost:110/pop3}INBOX", "user_id", "password");

// SSL IMAP あるいは POP3 サーバに接続するには、プロトコル指定のあとに
// /ssl を追加します。
$mbox = imap_open ("{localhost:993/imap/ssl}INBOX", "user_id", "password");

// 自分でサインした証明書で SSL IMAP あるいは POP3 サーバに接続するには、
// プロトコル指定のあとに /ssl/novalidate-cert を追加します。
$mbox = imap_open ("{localhost:995/pop3/ssl/novalidate-cert}", "user_id", "password");

// ローカルマシンのポート 119 で稼動している NNTP サーバに接続するには、
$nntp = imap_open ("{localhost:119/nntp}comp.test", "", "");
// リモートサーバに接続する際は、"localhost" を接続したいサーバの
// 名前または IP アドレスに置き換えます。
?>
```

Example#2 imap_open() の例

```
<?php
$mbox = imap_open("{imap.example.org:143}", "username", "password");

echo "<h1>Mailboxes</h1>¥n";
$folders = imap_listmailbox($mbox, "{imap.example.org:143}", "*");

if ($folders == false) {
    echo "コールが失敗しました<br />\n";
} else {
     foreach ($folders as $val) {
         echo $val . "<br />¥n";
     }
}

echo "<h1>INBOX のヘッダ</h1>\n";
$headers = imap_headers($mbox);

if ($headers == false) {
    echo "コールが失敗しました<br />\n";
} else {
     foreach ($headers as $val) {
         echo $val . "<br />¥n";
     }
}

imap_close($mbox);
?>
```

参考

- imap_close()

---

imap_ping

(PHP 4, PHP 5)

imap_ping — IMAP ストリームがアクティブかどうかを調べる

説明

bool **imap_ping** (resource $imap_stream)

imap_ping() はストリームに ping を行い、 まだアクティブであるかどうかを調べます。これにより新しいメールの到着を知ることもあります。これは、定期的に「メールのチェック」を行い、サーバとの 「接続を保持し続ける」ためのお勧めの方法です。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

返り値

ストリームがまだ有効であれば **TRUE** を、そうでなければ **FALSE** を返します。

例

Example#1 imap_ping() の例

```
<?php

$imap = imap_open("{imap.example.org}", "mailadmin", "password");

// しばらく休んだ後
if (!imap_ping($imap)) {
    // 再接続の処理
}

?>
```

---

imap_qprint

(PHP 4, PHP 5)

imap_qprint — quoted-printable 文字列を 8 ビット文字列に変換する

説明

string **imap_qprint** (string $string)

（» RFC2045, section 6.7 に基づき）quoted-printable 文字列を 8 ビット文字列に変換します。

パラメータ

string

quoted-printable 文字列。

返り値

8 ビット文字列を返します。

参考

- imap_8bit()

---

imap_renamemailbox

(PHP 4, PHP 5)

imap_renamemailbox — メールボックスの名前を変更する

説明

bool **imap_renamemailbox** (resource $imap_stream , string $old_mbox , string $new_mbox)

この関数は、古いメールボックスを新しいメールボックスにリネーム します（mbox 名のフォーマットについては imap_open() を参照ください）。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

old_mbox

古いメールボックス名。詳細は imap_open() を参照ください。

new_mbox

新しいメールボックス名。詳細は imap_open() を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_createmailbox()
- imap_deletemailbox()

---

imap_reopen

(PHP 4, PHP 5)

imap_reopen — 新規メールボックスへの IMAP ストリームを再度オープンする

説明

bool **imap_reopen** (resource $imap_stream , string $mailbox [, int $options [, int $n_retries]])

IMAP または NNTP サーバ上の新しい mailbox に対して指定されたストリームを再オープンします。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

options

options はビットマスクであり、以下の組み合わせとなります。

- **OP_READONLY** - メールボックスを読み込み専用でオープンします。
- **OP_ANONYMOUS** - ニュースに関して .newsrc を使用せず、更新もしません (NNTP のみ)。
- **OP_HALFOPEN** - IMAP および NNTP 名について、接続をオープンするが メールボックスをオープンしません。
- **OP_EXPUNGE** - 何も警告せずにリサイクルストリームを削除します。
- **CL_EXPUNGE** - メールボックスを閉じた際に、自動的に メールボックスを削除する (imap_delete() および imap_expunge() も参照ください)。

n_retries

接続試行の最大数。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.2.0	n_retries が追加されました。

例

Example#1 imap_reopen() の例

```
<?php
$mbox = imap_open("{imap.example.org:143}INBOX", "username", "password") or die(implode(", ", imap_errors()));
// ...
imap_reopen($mbox, "{imap.example.org:143}INBOX.Sent") or die(implode(", ", imap_errors()));
// ..
?>
```

---

imap_rfc822_parse_adrlist

(PHP 4, PHP 5)

imap_rfc822_parse_adrlist — アドレス文字列を解釈します

説明

array **imap_rfc822_parse_adrlist** (string $address , string $default_host)

» RFC2822 の定義に基づき、アドレス文字列をパースします。

パラメータ

address

アドレスを含む文字列。

default_host

デフォルトのホスト名。

返り値

オブジェクトの配列を返します。オブジェクトのプロパティは以下のとおりです。

- mailbox - メールボックス名（ユーザ名）。
- host - ホスト名。
- personal - 個人名。
- adl - ドメインソースルート。

例

Example#1 imap_rfc822_parse_adrlist() の例

```
<?php

$address_string = "Joe Doe <doe@example.com>, postmaster@example.com, root";
$address_array  = imap_rfc822_parse_adrlist($address_string, "example.com");
if (!is_array($address_array) || count($address_array) < 1) {
    die("something is wrong¥n");
}

foreach ($address_array as $id => $val) {
    echo "# $id¥n";
    echo "  mailbox : " . $val->mailbox . "¥n";
    echo "  host    : " . $val->host . "¥n";
    echo "  personal: " . $val->personal . "¥n";
    echo "  adl     : " . $val->adl . "¥n";
}
?>
```

上の例の出力は以下となります。

```
# 0
  mailbox : doe
  host    : example.com
  personal: Joe Doe
  adl     :
# 1
  mailbox : postmaster
  host    : example.com
  personal:
  adl     :
# 2
  mailbox : root
  host    : example.com
  personal:
  adl     :
```

参考

- imap_rfc822_parse_headers()

---

imap_rfc822_parse_headers

(PHP 4, PHP 5)

imap_rfc822_parse_headers — 文字列からメールヘッダを解釈する

説明

object **imap_rfc822_parse_headers** (string $headers [, string $defaulthost])

複数のヘッダ要素を含むオブジェクトを取得します。 imap_header() と同じようなものです。

パラメータ

headers

パースされたヘッダデータ。

defaulthost

デフォルトのホスト名。

返り値

imap_header() が返すオブジェクトと似たものを返します。 フラグおよび他の要素は IMAP サーバから取得するいう点が異なります。

参考

- imap_rfc822_parse_adrlist()

---

imap_rfc822_write_address

(PHP 4, PHP 5)

imap_rfc822_write_address — 指定したメールボックス、ホスト、個人情報を、 電子メールアドレスとして適当な形式にして返す

説明

string **imap_rfc822_write_address** (string $mailbox , string $host , string $personal)

指定された情報を » RFC822 の定義に基づき 適切にフォーマットされた電子メールアドレスにして返します。

パラメータ

mailbox

メールボックス名。詳細は imap_open() を参照ください。

host

email のホスト部分。

personal

アカウント所有者の名前。

返り値

» RFC822 の定義に基づき 適切にフォーマットされた電子メールアドレスを返します。

例

Example#1 imap_rfc822_write_address() の例

```
<?php
echo imap_rfc822_write_address("hartmut", "example.com", "Hartmut Holzgraefe");
?>
```

上の例の出力は以下となります。

```
Hartmut Holzgraefe <hartmut@example.com>
```

---

imap_savebody

(PHP 5 >= 5.1.3)

imap_savebody — 指定した本文部をファイルに保存する

説明

bool **imap_savebody** (resource $imap_stream , mixed $file , int $msg_number [, string $part_number [, int $options]])

指定したメッセージの本文全体、あるいはその一部を保存します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

file

保存先ファイルへのパスを表す文字列、あるいは fopen() が返すファイル記述子。

msg_number

メッセージ番号。

part_number

パート番号。ピリオドで区切られた整数文字列を指定します。 これは、IMAP4 仕様における本文パートのリストへのインデックスとなります。

options

ビットマスクであり、以下の組合わせとなります。

- **FT_UID** - msg_number は UID である
- **FT_PEEK** - 既に設定されていない場合、 \Seen フラグを設定しない
- **FT_INTERNAL** - 内部フォーマットで文字列を返す。CRLF は正規化しない。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_fetchbody()

---

imap_scanmailbox

(PHP 4, PHP 5)

imap_scanmailbox — imap_listscan() のエイリアス

説明

この関数は次の関数のエイリアスです。 imap_listscan().

---

imap_search

(PHP 4, PHP 5)

imap_search — 指定した検索条件にマッチするメッセージを配列として返す

説明

array **imap_search** (resource $imap_stream , string $criteria [, int $options [, string $charset]])

この関数は、指定した imap ストリームの現在オープンしているメールボックス において検索を行います。

例えば、Mom から送られた全ての未回答のメッセージにマッチさせるには、 "UNANSWERED FROM mom" を使用します。 検索は、大文字小文字が区別されずに行われます。 検索条件のリストは、UW C クライアントのソースコードからのものであり、 不完全または不正確である可能性があります（» RFC2060, section 6.4.4 も参照ください）。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

criteria

空白で区切られた文字列で、以下のキーワードが使用可能です。 複数の単語からなるキーワード（例 FROM "joe smith"） は全て引用符で括る必要があります。

- ALL - 他の検索条件にマッチする全メッセージを返す
- ANSWERED - \\ANSWERED フラグが設定されているメッセージにマッチ
- BCC "string" - Bcc: フィールドに "string" を有するメッセージにマッチ
- BEFORE "date" - "date" より前の Date: を有するメッセージにマッチ
- BODY "string" - メッセージ本体に "string" を有するメッセージにマッチ
- CC "string" - Cc: フィールドに "string" を有するメッセージにマッチ
- DELETED - 削除されたメッセージにマッチ
- FLAGGED - \\FLAGGED フラグ（重要または緊急を表すものとして 使用されることがあります）が設定されているメッセージにマッチ
- FROM "string" - From: フィールドに "string" を有するメッセージにマッチ
- KEYWORD "string" - キーワードとして "string" を有するメッセージにマッチ
- NEW - 新規メッセージにマッチ
- OLD - 古いメッセージにマッチ
- ON "date" - Date: が "date" であるメッセージにマッチ
- RECENT - \\RECENT フラグが設定されているメッセージにマッチ
- SEEN - 既読の（\\SEEN フラグが設定されている）メッセージにマッチ
- SINCE "date" - Date: が "date" 以降であるメッセージにマッチ
- SUBJECT "string" - Subject: に "string" を有するメッセージにマッチ
- TEXT "string" - テキスト "string" を有するメッセージにマッチ
- TO "string" - To: に "string" を有するメッセージにマッチ
- UNANSWERED - 未回答のメッセージにマッチ
- UNDELETED - 削除されていないメッセージにマッチ
- UNFLAGGED - フラグが設定されていないメッセージにマッチ
- UNKEYWORD "string" - キーワード "string" を有さないメッセージにマッチ
- UNSEEN - 未読のメッセージにマッチ

options

options に指定できる値は **SE_UID** です。これを指定すると、 メッセージ番号ではなく UID を含む配列を返します。

charset

返り値

メッセージ番号あるいは UID の配列を返します。

検索条件 criteria を処理できなかった場合や 結果が見つからなかった場合は、**FALSE** を返します。

変更履歴

バージョン	説明
4.3.3	パラメータ charset が追加されました。

参考

- imap_listscan()

---

imap_set_quota

(PHP 4 >= 4.0.5, PHP 5)

imap_set_quota — 指定したメールボックスにクォータを設定する

説明

bool **imap_set_quota** (resource $imap_stream , string $quota_root , int $quota_limit)

メールボックス単位でクォータ上限(最大容量)を設定します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

quota_root

クォータを設定するメールボックス。 これは、メールボックスの IMAP 標準フォーマット 'user.name' に基づき指定する必要があります。

quota_limit

quota_root の最大サイズ (KB 単位)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imap_set_quota() の例

```
<?php
$mbox = imap_open("{imap.example.org:143}", "mailadmin", "password");

if (!imap_set_quota($mbox, "user.kalowsky", 3000)) {
    echo "クォータの設定に失敗\n";
     return;
}

imap_close($mbox);
?>
```

注意

この関数は、現在は c-client2000 以降のライブラリのユーザのみ使用可能です。

メール管理者のアカウントでオープンされている imap_stream を必要とします。 他のユーザでオープンされている場合は、この関数は動作しません。

参考

- imap_open()
- imap_get_quota()

---

imap_setacl

(PHP 4 >= 4.0.7, PHP 5)

imap_setacl — 指定したメールボックスの ACL を設定する

説明

bool **imap_setacl** (resource $imap_stream , string $mailbox , string $id , string $rights)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

id

rights

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

この関数は、現在 c-client2000 以降のユーザのみ使用可能です。

参考

- imap_getacl()

---

imap_setflag_full

(PHP 4, PHP 5)

imap_setflag_full — メッセージにフラグをセットする

説明

bool **imap_setflag_full** (resource $imap_stream , string $sequence , string $flag [, int $options])

この関数は、指定した sequence のメッセージの フラグに指定した flag を設定し、保存します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

sequence

メッセージ番号のシーケンス。 X,Y 形式でメッセージを列挙したり、 X:Y 形式で範囲内のすべてのメッセージを指定したりすることができます。

flag

設定可能なフラグは、(» RFC2060 で定義された) ¥¥Seen、 ¥¥Answered、¥¥Flagged、 ¥¥Deleted および ¥¥Draft です。

options

options はビットマスクであり、以下の組み合わせとなります。

- **ST_UID** - シーケンス引数はシーケンス番号の代わりに UID を含みます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 imap_setflag_full() の例

```
<?php
$mbox = imap_open("{imap.example.org:143}", "username", "password")
      or die("接続できません: " . imap_last_error());

$status = imap_setflag_full($mbox, "2,5", "¥¥Seen ¥¥Flagged");

echo gettype($status) . "¥n";
echo $status . "¥n";

imap_close($mbox);
?>
```

参考

- imap_clearflag_full()

---

imap_sort

(PHP 4, PHP 5)

imap_sort — メッセージヘッダの配列をソートする

説明

array **imap_sort** (resource $imap_stream , int $criteria , int $reverse [, int $options [, string $search_criteria [, string $charset]]])

指定したパラメータにより取得したメッセージ番号をソートします。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

criteria

criteria は、次のどれかとします（ひとつのみ）。

- ***SORTDATE*** *- メッセージの日付*
- ***SORTARRIVAL*** *- 到着日付*
- ***SORTFROM*** *- 最初の From アドレスのメールボックス*
- ***SORTSUBJECT*** *- メッセージ Subject*
- ***SORTTO*** *- 最初の To アドレスのメールボックス*
- ***SORTCC*** *- 最初の cc アドレスのメールボックス*
- ***SORTSIZE*** *- メッセージのサイズ（バイト単位）*

reverse

これを 1 にすると、逆順にソートします。

options

options はビットマスクで、以下の組み合わせとなります。

- ***SE_UID*** *- シーケンス番号の変わりに UID を返す*
- ***SE_NOPREFETCH*** *- 検索したメッセージを事前取得しない*

search_criteria

charset

返り値

指定したパラメータでソートしたメッセージ番号の配列を返します。

変更履歴

バージョン	説明
4.3.3	パラメータ *charset* が追加されました。

---

imap_status

(PHP 4, PHP 5)

imap_status — 現在のメールボックス以外のメールボックスのステータス情報を返す

説明

object ***imap_status*** *(resource $imap_stream , string $mailbox , int $options)*

指定したメールボックス mailbox についてのステータス情報を取得します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

options

以下のフラグが使用できます。

- ***SA_MESSAGES*** *- status->messages にメールボックスのメッセージ数を設定する*
- ***SA_RECENT*** *- status->recent にメールボックスの最近のメッセージ数を設定する*
- ***SA_UNSEEN*** *- status->unseen にメールボックスの未読の(新規)メッセージ数を設定する*
- ***SA_UIDNEXT*** *- status->uidnext にメールボックスの次の UID を設定する*
- ***SA_UIDVALIDITY*** *- メールボックスの UID がもはや有効ではない場合に変化する定数を status->uidvalidity に設定する*
- ***SA_ALL*** *- 上記のものをすべて設定する*

返り値

この関数は、ステータス情報を含むオブジェクトを返します。このオブジェクトには messages、 recent、unseen、 uidnext および uidvalidity というプロパティが含まれます。

flags にも、 上の各定数に対応するビットマスクを設定することができます。

例

Example#1 imap_status() の例

```
<?php
$mbox = imap_open("{imap.example.com}", "username", "password", OP_HALFOPEN)
       or die("接続できません: " . imap_last_error());

$status = imap_status($mbox, "{imap.example.org}INBOX", SA_ALL);
if ($status) {
   echo "Messages:    " . $status->messages     . "<br />¥n";
   echo "Recent:      " . $status->recent       . "<br />¥n";
   echo "Unseen:      " . $status->unseen       . "<br />¥n";
   echo "UIDnext:     " . $status->uidnext      . "<br />¥n";
```

```
    echo "UIDvalidity:" . $status->uidvalidity . "<br />¥n";
} else {
    echo "imap_status failed: " . imap_last_error() . "¥n";
}

imap_close($mbox);
?>
```

---

imap_subscribe

(PHP 4, PHP 5)

imap_subscribe — メールボックスを購読する

説明

bool **imap_subscribe** (resource $imap_stream , string $mailbox)

新規メールボックスを購読します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_unsubscribe()

---

imap_thread

(PHP 4 >= 4.0.7, PHP 5)

imap_thread — スレッド化したメッセージのツリーを返す

説明

array **imap_thread** (resource $imap_stream [, int $options])

スレッド化されたメッセージのツリーを取得します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

options

返り値

imap_thread() は、REFERENCES でスレッド化したメッセージのツリーを含む連想配列を返します。 エラー時には **FALSE** を返します。

現在のメールボックス内のすべてのメッセージが、結果の配列の 3 つの エントリで表されます。

- $thread["XX.num"] - 現在のメッセージ番号。
- $thread["XX.next"]
- $thread["XX.branch"]

例

Example#1 imap_thread() の例

```
<?php

// ここでは、ニュースグループのスレッドを HTML で出力します。

$nntp = imap_open('{news.example.com:119/nntp}some.newsgroup', '', '');
$threads = imap_thread($nntp);

foreach ($threads as $key => $val) {
  $tree = explode('.', $key);
  if ($tree[1] == 'num') {
    $header = imap_headerinfo($nntp, $val);
    echo "<ul>¥n¥t<li>" . $header->fromaddress . "¥n";
```

```
  } elseif ($tree[1] == 'branch') {
    echo "¥t</li>¥n</ul>¥n";
  }
}

imap_close($nntp);

?>
```

---

imap_timeout

(PHP 4 >= 4.3.3, PHP 5)

imap_timeout — imap タイムアウトを設定あるいは取得する

説明

mixed **imap_timeout** (int $timeout_type [, int $timeout])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

imap_uid

(PHP 4, PHP 5)

imap_uid — 指定したメッセージシーケンス番号の UID を返す

説明

int **imap_uid** (resource $imap_stream , int $msg_number)

この関数は、指定したメッセージシーケンス番号の UID を返します。 メッセージ番号はメールボックスの内容を変更する度に変わる可能性が ありますが、UID はユニークな ID であり時間が経過しても変わりません。

この関数は、imap_msgno() の逆関数です。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

返り値

指定したメッセージの UID を返します。

注意

注意: この関数は、POP3 メールボックスではサポートされません。

参考

- imap_msgno()

---

imap_undelete

(PHP 4, PHP 5)

imap_undelete — 削除マークがついているメッセージのマークをはずす

説明

bool **imap_undelete** (resource $imap_stream , int $msg_number [, int $flags])

指定したメッセージについて、 imap_delete() または imap_mail_move() で設定された削除フラグをはずします。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

msg_number

メッセージ番号。

flags

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_delete()
- imap_mail_move()

---

imap_unsubscribe

(PHP 4, PHP 5)

imap_unsubscribe — メールボックスの購読をやめる

説明

bool **imap_unsubscribe** (string $imap_stream , string $mailbox)

指定されたメールボックス mailbox の購読を中止します。

パラメータ

imap_stream

imap_open() が返す IMAP ストリーム。

mailbox

メールボックス名。詳細は imap_open() を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- imap_subscribe()

---

imap_utf7_decode

(PHP 4, PHP 5)

imap_utf7_decode — 修正版 UTF-7 エンコードされた文字列をデコードする

説明

string **imap_utf7_decode** (string $text)

修正版 UTF-7 の text を ISO-8859-1 文字列に デコードします。

この関数は、ある種の非 ASCII 文字を含むメールボックス名をデコードする際に 必要となります。

パラメータ

text

修正版 UTF-7 エンコーディングの文字列。このエンコーディングについては » RFC 2060, section 5.1.3 で定義されています (元の UTF-7 は » RFC1642 で定義されています)。

返り値

text と同じ内容の文字を ISO-8859-1 でエンコード した文字列を返します。text に修正版 UTF-7 として 不正な文字が含まれていた場合、あるいは text に ISO-8859-1 文字セットの範囲外の文字が含まれていた場合には **FALSE** を返します。

参考

- imap_utf7_encode()

---

imap_utf7_encode

(PHP 4, PHP 5)

imap_utf7_encode — ISO-8859-1 文字列を修正版 UTF-7 テキストに変換する

説明

string **imap_utf7_encode** (string $data)

data を修正版 UTF-7 テキストに変換します。

この関数は、ある種の非 ASCII 文字を含むメールボックス名をエンコード する際に必要となります。

パラメータ

data

ISO-8859-1 文字列。

返り値

data を修正版 UTF-7 でエンコードした文字列を返します。 このエンコーディングについては » RFC 2060, section 5.1.3 で定義されています (元の UTF-7 は » RFC1642 で定義されています)。

参考

- imap_utf7_decode()

---

imap_utf8

(PHP 4, PHP 5)

imap_utf8 — MIME エンコードされたテキストを UTF-8 に変換する

説明

string **imap_utf8** (string $mime_encoded_text)

指定した mime_encoded_text を UTF-8 に変換します。

パラメータ

mime_encoded_text

MIME エンコードされた文字列。 MIME エンコーディング方法および UTF-8 の仕様については、それぞれ » RFC2047 および » RFC2044 を参照ください。

返り値

UTF-8 エンコードされた文字列を返します。

参考

- imap_mime_header_decode()

目次

- imap_headerinfo — メッセージヘッダを読み込む
- imap_headers — メールボックス内のすべてのメッセージのヘッダを返す
- imap_last_error — ページリクエスト時に生じた直近の IMAP エラーを返す
- imap_list — メールボックスのリストを読み込む
- imap_listmailbox — imap_list のエイリアス
- imap_listscan — 指定したテキストにマッチするメールボックスの一覧を返す
- imap_listsubscribed — imap_lsub のエイリアス
- imap_lsub — 購読しているすべてのメールボックスの一覧を得る
- imap_mail_compose — 指定したエンベロープおよびボディセクションに基づいて MIME メッセージを作成する
- imap_mail_copy — 指定されたメッセージをメールボックスにコピーする
- imap_mail_move — 指定されたメッセージをメールボックスに移動する
- imap_mail — e-mail メッセージを送信する
- imap_mailboxmsginfo — 現在のメールボックスに関する情報を得る
- imap_mime_header_decode — MIME ヘッダ要素をデコードする
- imap_msgno — 指定した UID のメッセージ番号を返す
- imap_num_msg — 現在のメールボックスのメッセージ数を取得する
- imap_num_recent — 現在のメールボックスにある新規メッセージ数を取得する
- imap_open — メールボックスへの IMAP ストリームをオープンする
- imap_ping — IMAP ストリームがアクティブかどうかを調べる
- imap_qprint — quoted-printable 文字列を 8 ビット文字列に変換する
- imap_renamemailbox — メールボックスの名前を変更する
- imap_reopen — 新規メールボックスへの IMAP ストリームを再度オープンする
- imap_rfc822_parse_adrlist — アドレス文字列を解釈します
- imap_rfc822_parse_headers — 文字列からメールヘッダを解釈する
- imap_rfc822_write_address — 指定したメールボックス、ホスト、個人情報を、 電子メールアドレスとして適当な形式にして返す
- imap_savebody — 指定した本文部をファイルに保存する
- imap_scanmailbox — imap_listscan のエイリアス
- imap_search — 指定した検索条件にマッチするメッセージを配列として返す
- imap_set_quota — 指定したメールボックスにクォータを設定する
- imap_setacl — 指定したメールボックスの ACL を設定する
- imap_setflag_full — メッセージにフラグをセットする
- imap_sort — メッセージヘッダの配列をソートする
- imap_status — 現在のメールボックス以外のメールボックスのステータス情報を返す
- imap_subscribe — メールボックスを購読する
- imap_thread — スレッド化したメッセージのツリーを返す
- imap_timeout — imap タイムアウトを設定あるいは取得する
- imap_uid — 指定したメッセージシーケンス番号の UID を返す
- imap_undelete — 削除マークがついているメッセージのマークをはずす
- imap_unsubscribe — メールボックスの購読をやめる
- imap_utf7_decode — 修正版 UTF-7 エンコードされた文字列をデコードする
- imap_utf7_encode — ISO-8859-1 文字列を修正版 UTF-7 テキストに変換する
- imap_utf8 — MIME エンコードされたテキストを UTF-8 に変換する

Informix 関数

導入

Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x ,IDS 2000 用の Informix ドライバは、informix 拡張機能用ディレクトリの "ifx.ec" および "php3_ifx.h" に実装されています。 IDS 7.x のサポートは BYTE および TEXT カラムを完全にサポートしており、 完成度はかなり高いです。 IUS 9.x のサポートは部分的に完成しています。つまり、 新しいデータ型はサポートされていますが、SLOB および CLOB のサポートについてはまだ作業中です。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.2.1.

要件

> **注意**: **設定に関する注意** PHP Informix ドライバをコンパイルするには、ESQL/C が必要です。 7.2x 以降のバージョンに付属する ESQL/C は問題なく使用できます。 現在では、ESQL/C は Informix クライアント SDK に含まれています。 "configure" スクリプトを実行する前に、必ず "INFORMIXDIR" 変数を 設定し、PATH に $INFORMIXDIR/bin を設定しておいてください。

インストール手順

このモジュールで定義した関数を使用可能とするには、configure に --with_informix[=DIR] を指定して PHP インタプリタをコンパイルする必要があります。ただし、DIR は Informix のベースインストールディレクトリで、デフォルトはありません。

実行時設定

php.ini の設定により動作が変化します。

注意: Informix 用環境変数 INFORMIXDIR および INFORMIXSERVER が PHP ifx ドライバで利用可能であり、INFORMIX のバイナリがあるディレクトリに パスが通っていることを確認してください。テストを始める前に phpinfo() と書いたスクリプトを実行し、これを確認してください。 phpinfo() があるスクリプトは、これらの環境変数の 一覧を出力します。これは、CGI 版の PHP および Apache mod_php で 共に行われます。これらの環境変数は Apache のスタートアップスクリプトで 設定する必要があります。また、Informix 共有ライブラリがローダーで利用可能である必要があります (LD_LINBRARY_PATH または ld.so.conf/ldconfig を確認してください)。

注意: **BLOB (TEXT および BYTE カラム)の使用に関する注意** 通常、BLOB は BLOB ID により指定されます。 select クエリは、BYTE および TEXT カラム毎に "blob id" を返します。 ("ifx_blobinfile(0);" により) メモリー上で BLOB を得ることを選択した場合、 "string_var = ifx_get_blob($blob_id);" で内容を得ることができます。 BLOB カラムの内容をファイルに取得したい場合、 "ifx_blobinfile(1);" を使用してください。 "ifx_get_blob($blob_id);" によりファイル名を得ることができます。 BLOB の内容を得る際には、通常のファイル入出力を行ってください。
insert/update クエリーに関しては、 "ifx_create_blob();" により自分で "blob id" を作成する必要があります。 その後、blob id を配列に代入し、 クエリー文字列の中の blob カラムを疑問符 (?) で置換します。 updates/inserts の場合、ifx_update_blob() で blob の内容を設定するのが便利でしょう。
BLOB カラムの動作は、設定用変数で変更することができます。 これらの変数は、実行時にも設定可能です。
設定変数 : ifx.textasvarchar
設定変数 : ifx.byteasvarchar
ランタイム関数 :
ifx_textasvarchar(0) : TEXT カラムを有する select クエリーに blob id を使用する
ifx_byteasvarchar(0) : BYTE カラムを有する select クエリーに blob id を使用する
ifx_textasvarchar(1) : TEXT カラムを VARCHAR カラムとして返します。 このため、select クエリにおいて blob id を使用する必要はありません。
ifx_byteasvarchar(1) : BYTE カラムを VARCHAR カラムとして返します。 このため、select クエリにおいて blob id を使用する必要はありません。
設定変数 : ifx.blobinfile
ランタイム関数 :
ifx_blobinfile_mode(0) : メモリに BYTE カラムを返し、blob id によりその内容を取り出す
ifx_blobinfile_mode(1) : メモリに BYTE カラムを返し、blob id によりそのファイル名を取り出す
ifx_text/byteasvarchar を 1 に設定した場合、 通常の(しかしやや長い) VARCHAR フィールドのように select クエリーで TEXT や BYTE カラムを使用することが可能です。 全ての文字列は、PHP で "数えられる" ので、これにより、 "バイナリ・セーフ"が維持されます。 これを正しく処理するのはあなた次第です。 返されるデータには何でも含むことができますが、その内容について 責任を負うことになります。
ifx_blobinfile を 1 に設定した場合、blob の内容を得るために ifx_get_blob(..) により返されたファイル名を使用してください。 この場合、行を取得する際に「Informix により作成された テンポラリファイルを削除する」責任があります。 取得された新規の行は、BYTE カラム毎に新規のテンポラリファイルを 作成します。
テンポラリファイルの位置は、環境変数 "blobdir" により設定することが できます。デフォルトは、"." (カレントディレクトリ)です。 putenv(blobdir="tmpblob"); のようにすることにより、誤って残って しまったテンポラリファイルを削除することが容易になります (テンポラリファイルの名前は "blb" で始まります)。

注意: **自動的に "char" (SQLCHAR および SQLNCHAR) データを取り去る** これは、次の設定変数により設定することが可能です。
ifx.charasvarchar : 最後尾のスペースを何らかの削除処理を行わずに 自動的に取り去る場合に 1 に設定します。

注意: **NULL 値** 設定変数 ifx.nullformat (およびランタイム関数 ifx_nullformat()) を **TRUE** に設定した場合、文字列 "**NULL**" として **NULL** カラムを返します。**FALSE** に設定した場合は 空文字列を返します。これにより、**NULL** カラムと空のカラムを識別することが 可能となります。

Informix 設定オプション

名前	デフォルト	変更の可否	変更履歴
ifx.allow_persistent	"1"	PHP_INI_SYSTEM	PHP 5.2.1 で削除。
ifx.max_persistent	"-1"	PHP_INI_SYSTEM	PHP 5.2.1 で削除。
ifx.max_links	"-1"	PHP_INI_SYSTEM	PHP 5.2.1 で削除。
ifx.default_host	NULL	PHP_INI_SYSTEM	PHP 5.2.1 で削除。
ifx.default_user	NULL	PHP_INI_SYSTEM	PHP 5.2.1 で削除。
ifx.default_password	NULL	PHP_INI_SYSTEM	PHP 5.2.1 で削除。
ifx.blobinfile	"1"	PHP_INI_ALL	PHP 5.2.1 で削除。
ifx.textasvarchar	"0"	PHP_INI_ALL	PHP 5.2.1 で削除。
ifx.byteasvarchar	"0"	PHP_INI_ALL	PHP 5.2.1 で削除。
ifx.charasvarchar	"0"	PHP_INI_ALL	PHP 5.2.1 で削除。
ifx.nullformat	"0"	PHP_INI_ALL	PHP 5.2.1 で削除。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

ifx.allow_persistent boolean

持続的な Informix 接続を可能とするかどうか。

ifx.max_persistent integer

プロセス毎の持続的 Informix 接続の最大数。

ifx.max_links integer

持続的接続を含むプロセス毎の Informix 接続の最大数。

ifx.default_host string

ifx_connect() または ifx_pconnect() において、ホストが指定されない 場合のデフォルトのホスト。 セーフモード では適用されません。

ifx.default_user string

ifx_connect() または ifx_pconnect() において、 ユーザが指定されない場合のデフォルトのユーザ。 セーフモード では適用されません。

ifx.default_password string

ifx_connect() または ifx_pconnect() において、 パスワードが指定されない場合のデフォルトのパスワード。 セーフモード では適用されません。

ifx.blobinfile boolean

blob カラムをファイルに返したい場合には TRUE を指定します。 メモリ内に返したい場合には FALSE を指定します。 ifx_blobinfile_mode() により、実行時にこの設定を 上書きすることができます。

ifx.textasvarchar boolean

select 文において TEXT カラムを通常の文字列として返したい場合は TRUE を指定し、blob id パラメータを使用したい場合は FALSE を指定します。 ifx_textasvarchar() により、実行時にこの設定を 上書きすることができます。

ifx.byteasvarchar boolean

select 文において BYTE カラムを通常の文字列として返したい場合は TRUE を指定し、blob id パラメータを使用したい場合は FALSE を指定します。 ifx_byteasvarchar() により、実行時にこの設定を 上書きすることができます。

ifx.charasvarchar boolean

取得時に CHAR カラムから末尾の空白を取り除きたい場合に TRUE を指定してください。

ifx.nullformat boolean

リテラル文字列 "NULL" として NULL カラムを返したい場合に TRUE、 空の文字列として返したい場合は FALSE を指定してください。
ifx_nullformat() により、実行時にこの設定を 上書きすることができます。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

IFX_SCROLL (integer)
IFX_HOLD (integer)
IFX_LO_RDONLY (integer)
IFX_LO_WRONLY (integer)
IFX_LO_APPEND (integer)
IFX_LO_RDWR (integer)
IFX_LO_BUFFER (integer)
IFX_LO_NOBUFFER (integer)

---

ifx_affected_rows

(No version information available, might be only in CVS)

ifx_affected_rows — クエリで変更された行の数を得る

説明

int **ifx_affected_rows** (resource $result_id)

result_id に関連するクエリにより変更された 行の数を返します。

insert, update ,delete の場合、その数は、実際に作用された行の数 (sqlerrd[2]) です。select の場合、これは推定値 (sqlerrd[0]) です。 この値を信用してはいけません。データベースサーバは、SELECT により 実際に返される行の数を返すことはありません。これは、この段階 (オプティマイザがクエリ手順を定義している場合には、"PREPARE" を 行った直後) では、行の取得を始めてさえもいないためです。

ifx_prepare() の実行後に、クエリ結果を適当な量に 制限するために使用すると便利です。

パラメータ

result_id

ifx_query() または ifx_prepare() により返される有効な結果 ID です。

返り値

行の数を整数値で返します。

例

Example#1 Informix で変更された行

```
<?php
$rid = ifx_prepare("select * from emp
                    where name like " . $name, $connid);
if (! $rid) {
    /* ... error ... */
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)¥n<br />", $rowcount);
    die ("Please restrict your query<br />¥n");
}
?>
```

参考

- ifx_num_rows()

ifx_blobinfile_mode

(No version information available, might be only in CVS)

ifx_blobinfile_mode — 全ての select クエリに関するデフォルトの BLOB モードを設定する

説明

bool **ifx_blobinfile_mode** (int $mode)

全ての select クエリに関するデフォルトの BLOB モードを設定します。

パラメータ

mode

モード "0" は、BLOB をメモリーに保存することを意味し、 モード "1" は、BLOB をファイルに保存することを意味します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ifx_byteasvarchar

(No version information available, might be only in CVS)

ifx_byteasvarchar — デフォルトのバイトモードを設定する

説明

bool **ifx_byteasvarchar** (int $mode)

全ての select クエリに関するデフォルトのバイトモードを設定します。

パラメータ

mode

モード "0" は BLOB ID を返し、モード "1" がテキストの内容を有する varchar を返します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ifx_textasvarchar()

---

ifx_close

(No version information available, might be only in CVS)

ifx_close — Informix 接続を閉じる

説明

bool **ifx_close** ([resource $link_identifier])

ifx_close() は、 指定したリンク ID に関連づけられた Informix データベースへの接続を閉じます。

非持続的接続としてオープンされたリンクはスクリプトの実行終了時に 自動的に閉じられるため、 この関数は通常は必要ではないことに注意してください。

ifx_close() は、ifx_pconnect() により作成された持続的リンクは閉じません。

パラメータ

link_identifier

リンク ID。指定しなかった場合は、最後にオープンされたリンクが仮定されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Informix 接続を閉じる

```
<?php
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
/* ... クエリ等を実行 ... */
ifx_close($conn_id);
?>
```

参考

- ifx_connect()
- ifx_pconnect()

---

ifx_connect

(No version information available, might be only in CVS)

ifx_connect — Informix サーバへの接続をオープンする

説明

resource **ifx_connect** ([string $database [, string $userid [, string $password]]])

ifx_connect() は、Informix サーバへの接続を確立します。

同じ引数で **ifx_connect()** を 2 回目にコールした際には、 新規のリンクは確立されず、代わりに既にオープンされたリンクの リンク ID が返されます。

サーバへのリンクは、ifx_close() のコールに より明示的に事前に閉じない限り、スクリプトの実行終了直後に閉じられます。

パラメータ

全ての引数はオプションであり、指定されない場合には、 設定ファイルにて指定された値 がデフォルト値として設定されます。(ホストに関して ifx.default_host (定義されていない場合、Informix ライブラリは、環境変数INFORMIXSERVERを参照します) , ユーザーに関して ifx.default_user, パスワードに関して ifx.default_password (定義されていない場合は無し) となります。

database

データベース名を表す文字列。

userid

ユーザ名を表す文字列。

password

パスワードを表す文字列。

返り値

成功時に接続 ID を、エラー時に **FALSE** を返します。

例

Example#1 Informix データベースへの接続

```
<?php
$conn_id = ifx_connect ("mydb@ol_srv1", "imyself", "mypassword");
?>
```

参考

- ifx_pconnect()
- ifx_close()

---

ifx_copy_blob

(No version information available, might be only in CVS)

ifx_copy_blob — 指定した BLOB オブジェクトを二重化する

説明

int **ifx_copy_blob** (int $bid)

指定した BLOB オブジェクトを二重化します。

パラメータ

bid

BLOB オブジェクトの ID。

返り値

新規の BLOB オブジェクト ID を返します。 エラーの場合に **FALSE** を返します。

参考

- ifx_create_blob()
- ifx_free_blob()

---

ifx_create_blob

(No version information available, might be only in CVS)

ifx_create_blob — BLOB オブジェクトを作成する

説明

int **ifx_create_blob** (int $type , int $mode , string $param)

BLOB オブジェクトを作成します。

パラメータ

type

1 = TEXT, 0 = BYTE

mode

0 = BLOB オブジェクトはメモリーに内容を保持する, 1 = BLOB オブジェクトはファイルに内容を保持する。

param

mode = 0 の場合: 内容へのポインタ mode = 1 の場合: ファイル文字列へのポインタ

返り値

新規の BLOB オブジェクト ID を返します。 エラーの場合に **FALSE** を返します。

参考

- ifx_copy_blob()
- ifx_free_blob()

---

ifx_create_char

(No version information available, might be only in CVS)

ifx_create_char — 文字オブジェクトを作成する

説明

int **ifx_create_char** (string $param)

文字オブジェクトを作成します。

パラメータ

param

内容となる文字。

返り値

新しい文字オブジェクトの ID、あるいはエラー時に **FALSE** を返します。

参考

- ifx_free_char()

---

ifx_do

(No version information available, might be only in CVS)

ifx_do — 事前に準備された SQL 文を実行する

説明

bool **ifx_do** (resource $result_id)

事前に準備されたクエリを実行し、カーソルをオープンします。

エラーの際に、result_id を解放しないでください。

select 文でない場合に、ifx_affected_rows() に数を設定します。 これは、ifx_affected_rows() で取得可能です。

パラメータ

result_id

result_id は、 ifx_query() あるいは ifx_prepare() が返す有効な結果 ID です (select 型のクエリのみです!)。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

Example#1 ifx_do() の例

```
<?php
$conn = fx_connect( "db", "user", "password" );
$result = ifx_prepare("SELECT customer_num, company FROM customer", $conn);
ifx_do($result);
?>
```

参考

- ifx_prepare()

---

ifx_error

(No version information available, might be only in CVS)

ifx_error — 直近の Informix コールのエラーコードを返す

説明

string **ifx_error** ([resource $link_identifier])

文の結果を表す一文字の文字列、そして、 直近に実行された SQL 文についての SQLSTATE と SQLCODE を返します。

パラメータ

link_identifier

リンク ID。

返り値

Informix エラーコード (SQLSTATE & SQLCODE) は、 x [SQLSTATE = aa bbb SQLCODE=cccc] のような形式になります。

x = 空白 : エラーなし

E : エラー

N : データがもうない

W : 警告

? : 未定義

文字 "x" が空白以外の文字だった場合、SQLSTATE および SQLCODE でエラーの詳細情報を取得します。

SQLSTATE および SQLCODE の詳細については、Informix マニュアルを参照ください。

参考

- ifx_errormsg()

---

ifx_errormsg

(No version information available, might be only in CVS)

ifx_errormsg — 直近の Informix コールのエラーメッセージを返す

説明

string **ifx_errormsg** ([int $errorcode])

直近の Informix エラーに関する Informix エラーメッセージを返します。

パラメータ

errorcode

指定すると、指定したコードに対応するエラーメッセージを返します。

返り値

エラーメッセージを文字列で返します。

例

Example#1 ifx_errormsg() の例

```
printf("%s¥n&lt;br>",  ifx_errormsg(-201));
```

参考

- ifx_error()

---

ifx_fetch_row

(No version information available, might be only in CVS)

ifx_fetch_row — 行を連想配列として取得する

説明

array **ifx_fetch_row** (resource $result_id [, mixed $position])

結果 ID で指定した結果に対応するデータのひとつの行を取得します。

ifx_fetch_row() を続けてコールした場合、結果セットの 次の行が返されるます。 行がもうない場合は、**FALSE** が返されます。

パラメータ

result_id

ifx_query() または ifx_prepare() (select 型のクエリのみ!) により返された有効な結果 ID です。

position

オプションのパラメータで、"スクロール" カーソルで "取得" 操作を行うためのものです。 NEXT, PREVIOUS, CURRENT, FIRST, LAST あるいは番号を指定します。 番号を指定した場合、"絶対" 行の取得が行われます。このパラメータは オプションであり、SCROLL カーソルの場合にのみ有効です。

返り値

取得された行に対応する連想配列を返します。行がもうない場合には **FALSE** を返します。

BLOB カラムは、ifx_get_blob() で使用するために 整数値 BLOB ID として返されます。ただし、ifx_textasvarchar(1) または ifx_byteasvarchar(1) を指定している場合を除きます。この場合、 BLOB は文字列として返されます。エラーの場合は **FALSE** が返されます。

例

Example#1 Informix 行の取得

```
<?php
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    /* ... error ... */
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)¥n<br />", $rowcount);
    die ("Please restrict your query<br />¥n");
}
if (! ifx_do ($rid)) {
   /* ... error ... */
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for (reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("¥n<br />");
    $row = ifx_fetch_row($rid, "NEXT");
}
ifx_free_result ($rid);
?>
```

---

ifx_fieldproperties

(No version information available, might be only in CVS)

ifx_fieldproperties — SQL フィールドプロパティのリスト

説明

array **ifx_fieldproperties** (resource $result_id)

クエリ中の全てのフィールドの Informix SQL フィールドプロパティを 連想配列として返します。プロパティは、以下のような形式となります。"SQLTYPE;length;precision;scale;ISNULLABLE" ただし、SQLTYPE は、 "SQLVCHAR" 等の Informix 型。ISNULLABLE は、"Y" または "N" となります。

パラメータ

result_id

result_id は、 ifx_query() または ifx_prepare() (select 型のクエリのみ!) により返された有効な結果 ID です。

返り値

result_id のクエリについて、フィールド名を キーとし、SQL フィールドプロパティをデータとした連想配列を返します。 エラーの場合に **FALSE** を返します。

例

Example#1 Informix SQL フィールドプロパティ

```
<?php
$properties = ifx_fieldproperties($resultid);
if (!isset($properties)) {
    /* ... エラー ... */
}
foreach ($properties as $fname => $val) {
     echo "$fname:¥t property = $val¥n";
}
?>
```

参考

- ifx_fieldtypes()

---

ifx_fieldtypes

(No version information available, might be only in CVS)

ifx_fieldtypes — Informix SQL フィールドのリスト

説明

array **ifx_fieldtypes** (resource $result_id)

result_id のクエリについて、フィールド名をキーとし、 SQL フィールド型をデータとした連想配列を返します。

パラメータ

result_id

result_id は、 ifx_query() または ifx_prepare() (select 型のクエリのみ!) により返された有効な結果 ID です。

返り値

result_id のクエリについて、フィールド名をキーとし、 SQL フィールド型をデータとした連想配列を返します。 エラーの場合に **FALSE** を返します。

例

Example#1 フィールド名および SQL フィールド型

```
<?php
$types = ifx_fieldtypes($resultid);
if (is_array($types)) {
     foreach ($types as $fname => $val) {
         echo "$fname:¥t type = $val¥n";
     }
}
?>
```

参考

- ifx_fieldproperties()

---

ifx_free_blob

(No version information available, might be only in CVS)

ifx_free_blob — BLOB オブジェクトを削除する

説明

bool **ifx_free_blob** (int $bid)

指定された BLOB オブジェクト ID の BLOB オブジェクトを削除します。

パラメータ

bid

BLOB オブジェクト ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ifx_create_blob()*

---

ifx_free_char

(No version information available, might be only in CVS)

ifx_free_char — 文字オブジェクトを削除する

説明

*bool **ifx_free_char** (int $bid)*

指定したオブジェクト ID の文字オブジェクトを削除します。

パラメータ

bid

文字オブジェクト ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ifx_create_char()*

---

ifx_free_result

(No version information available, might be only in CVS)

ifx_free_result — クエリに関するリソースを解放する

説明

*bool **ifx_free_result** (resource $result_id)*

result_id に間連するクエリの リソースを解放します。

パラメータ

result_id

result_id は、 ifx_query() または ifx_prepare() (select 型のクエリのみ!) により返された有効な結果 ID です。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ifx_do()*

---

ifx_get_blob

(No version information available, might be only in CVS)

ifx_get_blob — BLOB オブジェクトの内容を返す

説明

*string **ifx_get_blob** (int $bid)*

BLOB オブジェクトの内容を返します。

パラメータ

bid

BLOB オブジェクト ID。

返り値

BLOB の内容を表す文字列、あるいはエラー時に FALSE を返します。

参考

- ifx_get_char()

---

ifx_get_char

(No version information available, might be only in CVS)

ifx_get_char — 文字オブジェクトの内容を返す

説明

string **ifx_get_char** (int $bid)

文字オブジェクトの内容を返します。

パラメータ

bid

文字オブジェクト ID。

返り値

文字の内容を表す文字列、あるいはエラー時に FALSE を返します。

参考

- ifx_get_blob()

---

ifx_getsqlca

(No version information available, might be only in CVS)

ifx_getsqlca — クエリ実行後、sqlca.sqlerrd[0..5] の値を得る

説明

array **ifx_getsqlca** (resource $result_id)

result_id に関連するクエリを実行した後の sqlca.sqlerrd[0] から sqlca.sqlerrd[5] に関する擬似レコードを返します。

insert、update、delete の場合、クエリが実行された後、サーバにより 設定される場合と同様にレコードの値が返されます。 これにより、作用を受けた行の数および連番の挿入値にアクセスすることが 可能となります。 SELECT の場合、この値は PREPARE 文の後で保存された値となります。この値から作用を受けた行の数の"推測"値が分かります。 ifx ドライバにより適当な時に保存された値が取得されるため、 この関数を使用することにより、 SELECT dbinfo('sqlca.sqlerrdx') クエリを実行するオーバーヘッドを防止することができます。

パラメータ

result_id

result_id は、 ifx_query() または ifx_prepare() (select 型のクエリのみ!) により返された有効な結果 ID です。

返り値

連想配列を返します。含まれるエントリは sqlerrd0、sqlerrd1、 sqlerrd2、sqlerrd3、 sqlerrd4 および sqlerrd5 です。

例

Example#1 Informix sqlca.sqlerrd[x] の値を取得する

```
<?php
/* 'sometable' の最初のカラムは連番であると仮定する */
$qid = ifx_query("insert into sometable
                  values (0, '2nd column', 'another column') ", $connid);
if (!$qid) {
    /* ... error ... */
}
$sqlca = ifx_getsqlca($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "挿入された行の続き番号は : " . "$serial_value<br>¥n";
?>
```

---

ifx_htmltbl_result

(No version information available, might be only in CVS)

ifx_htmltbl_result — クエリ結果の全行を HTML テーブルにフォーマットする

説明

int **ifx_htmltbl_result** (resource $result_id [, string $html_table_options])

クエリ結果 result_id の全ての行を HTML テーブルにフォーマットします。

パラメータ

result_id

result_id is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

html_table_options

このオプション引数は、<table> タグのオプションとする文字列です。

返り値

取得された行の数、あるいはエラーの場合に **FALSE** を返します。

例

Example#1 Informix 結果を HTML テーブルとして出力する

```
<?php
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    /* ... error ... */
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)¥n<br />", $rowcount);
    die ("Please restrict your query<br />¥n");
}
if (! ifx_do($rid)) {
    /* ... error ... */
}

ifx_htmltbl_result ($rid, "border=¥"2¥"");

ifx_free_result($rid);
?>
```

---

ifx_nullformat

(No version information available, might be only in CVS)

ifx_nullformat — 取得する行のデフォルトの返り値を設定する

説明

bool **ifx_nullformat** (int $mode)

取得する行のデフォルトの返り値を NULL 値に設定します。

パラメータ

mode

モード "0" は "" を返し、モード "1" は "**NULL**" を返します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ifx_num_fields

(No version information available, might be only in CVS)

ifx_num_fields — クエリのカラム数を返す

説明

int **ifx_num_fields** (resource $result_id)

クエリを準備または実行された後、この関数をコールすることにより、 クエリ結果中でカラム数が得られます。

パラメータ

result_id

result_id は、 ifx_query() または ifx_prepare() (select 型のクエリのみ!) により返された有効な結果 ID です。

返り値

result_id に関するクエリ結果のカラム数、 あるいはエラーの場合にFALSEを返します。

例

Example#1 ifx_num_fields() の例

```
<?php
$conn_id = ifx_connect("db", "user", "password");
$res_id = ifx_query("select * from systables", $conn_id);
echo ifx_num_fields($res_id);
?>
```

参考

- ifx_num_rows()

---

ifx_num_rows

(No version information available, might be only in CVS)

ifx_num_rows — クエリから既に取得された行の数を数える

説明

int **ifx_num_rows** (resource $result_id)

ifx_query() または ifx_do() クエリの後、result_id を有する クエリに関してこれまで取得された行の数を返します。

パラメータ

result_id

result_id は、 ifx_query() または ifx_prepare() (select 型のクエリのみ!) により返された有効な結果 ID です。

返り値

取得した行の数、あるいはエラー時に FALSE を返します。

参考

- ifx_num_fields()

---

ifx_pconnect

(No version information available, might be only in CVS)

ifx_pconnect — 持続的 Informix 接続をオープンする

説明

resource **ifx_pconnect** ([string $database [, string $userid [, string $password]]])

ifx_pconnect() は、二つの大きな違いはあるものの、 ifx_connect() と非常に似た動作をします。

まず、接続時に、関数は、既にオープンされている同じホスト、ユーザ名、 パスワードの (持続的) リンクを探します。 そのリンクが見つかった場合、新規に接続をオープンする代わりに その ID が返されます。

第二に、SQL サーバーへの接続は、スクリプトの実行終了時に閉じられません。 代わりにリンクは、後の使用のためにオープンされたままとなります。 (ifx_close() は **ifx_pconnect()** により確立されたリンクを閉じません。)

この型のリンクは、このため、'持続的' であると呼ばれます。

パラメータ

全ての引数はオプションであり、指定されない場合には、 設定ファイルにて指定された値 がデフォルト値として設定されます。 (ホストに関して ifx.default_host (定義されていない場合、Informix ライブラリは、環境変数INFORMIXSERVERを参照します) , ユーザーに関して ifx.default_user, パスワードに関して ifx.default_password (定義されていない場合は無し) となります。

database

データベース名を表す文字列。

userid

ユーザ名を表す文字列。

password

パスワードを表す文字列。

返り値

返り値: 成功時に有効な Informix 持続的リンク ID 、エラー時に **FALSE**

参考

- ifx_connect()

---

ifx_prepare

(No version information available, might be only in CVS)

ifx_prepare — SQL 文を実行用に準備する

説明

resource **ifx_prepare** (string $query , resource $link_identifier [, int $cursor_def], mixed $blobidarray)

あとで ifx_do() で使用するための query を準備します。

"select 型" のクエリの場合はカーソルを宣言してオープンします。 select 以外のクエリは、直接実行されます。

どちらのタイプのクエリにおいても、(予測または実際の数として) 作用された行の数は ifx_affected_rows() により 取得可能です。

TEXT (または BYTE) カラムの内容が許すならば、 "ifx_textasvarchar(1)" または "ifx_byteasvarchar(1)" を使用する ことも可能です。これにより、TEXT (または BYTE) カラムは、select クエリの 通常の (しかし長い) VARCHAR カラムと同様に処理され、 BLOB ID で悩むこともなくなります。

ifx_textasvarchar(0) または ifx_byteasvarchar(0) (デフォルト値) の場合、select クエリは、BLOB ID (整数値) に属するものとして BLOB カラムを返します。 BLOB 関数により文字列またはファイルとして BLOB の値を得ることが 可能です (下記を参照ください)。

パラメータ

query

クエリ文字列。

link_identifier

リンク ID。

cursor_def

オプションのパラメータで、そのカーソルを "スクロール" または "ホールド" カーソルとすることができます。 このオプションはビットマスクであり、 **IFX_SCROLL**、 **IFX_HOLD** あるいは 両方とも指定することができます。

blobidarray

もし、クエリにおいて BLOB (BYTE または TEXT) カラムがある場合、 対応する "BLOB ID" を有する blobidarray パラメータを追加することが可能です。この場合、クエリテキストの これらのカラムを "?" で置換することが必要です。

返り値

後で ifx_do() で使用するための結果 ID、 あるいはエラー時に **FALSE** を返します。

参考

- ifx_do()

---

ifx_query

(No version information available, might be only in CVS)

ifx_query — Informix クエリを送信する

説明

resource **ifx_query** (string $query , resource $link_identifier [, int $cursor_type [, mixed $blobidarray]])

指定したリンク ID が指す現在アクティブなデータベースに クエリ query を送信します。

"select 型" のクエリの場合はカーソルを宣言してオープンします。 select 以外のクエリは、直接実行されます。

どちらのタイプのクエリにおいても、(予測または実際の数として) 作用された行の数は ifx_affected_rows() により 取得可能です。

TEXT (または BYTE) カラムの内容が許すならば、 "ifx_textasvarchar(1)" または "ifx_byteasvarchar(1)" を使用する ことも可能です。これにより、TEXT (または BYTE) カラムは、select クエリの 通常の (しかし長い) VARCHAR カラムと同様に処理され、 BLOB ID で悩むこともなくなります。

ifx_textasvarchar(0) または ifx_byteasvarchar(0) (デフォルト値) の場合、select クエリは、BLOB ID (整数値) に属するものとして BLOB カラムを返します。 BLOB 関数により文字列またはファイルとして BLOB の値を得ることが 可能です (下記を参照ください)。

パラメータ

query

クエリ文字列。

link_identifier

リンク ID。

cursor_def

オプションのパラメータで、そのカーソルを "スクロール" または "ホールド" カーソルとすることができます。 このオプションはビットマスクであり、 IFX_SCROLL、 IFX_HOLD あるいは 両方とも指定することができます。

blobidarray

もし、クエリにおいて BLOB (BYTE または TEXT) カラムがある場合、 対応する "BLOB ID" を有する blobidarray パラメータを追加することが可能です。この場合、クエリテキストの これらのカラムを "?" で置換することが必要です。

返り値

成功した場合に Informix 結果 ID、エラー時に FALSE を返します。

例

Example#1 "orders" テーブルの全行を HTML テーブルとして表示する

```
<?php
ifx_textasvarchar(1);      // BLOB 用に "text mode" を使用する
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s¥n<br />%s<br />¥n", ifx_error(), ifx_errormsg());
    die;
}
ifx_htmltbl_result($res_id, "border=¥"1¥"");
ifx_free_result($res_id);
?>
```

Example#2 値を "catalog" テーブルに挿入する

```
<?php

// バイトおよびテキストカラムに関する BLOB ID を作成する。
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");

// blob id を blobid 配列に保存
$blobidarray[] = $textid;
$blobidarray[] = $byteid;

// クエリを実行
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
    /* ... エラー ... */
}

// 結果 id を解放
ifx_free_result($res_id);
?>
```

参考

- ifx_connect()

---

ifx_textasvarchar

(No version information available, might be only in CVS)

ifx_textasvarchar — デフォルトのテキストモードを設定する

説明

bool **ifx_textasvarchar** (int $mode)

全ての select クエリに関するデフォルトのテキストモードを設定します。

パラメータ

mode

モード "0" は、BLOB を返し、"1" は、テキストの内容を有する varchar を 返します。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- ifx_bytesasvarchar()

---

ifx_update_blob

(No version information available, might be only in CVS)

ifx_update_blob — BLOB オブジェクトの内容を更新する

説明

bool **ifx_update_blob** (int $bid , string $content)

指定したBLOB オブジェクト ID bid に関する BLOB オブジェクトの内容を更新します。

パラメータ

bid

BLOB オブジェクト ID。

content

新規データの文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ifx_update_char()

---

ifx_update_char

(No version information available, might be only in CVS)

ifx_update_char — 文字オブジェクトの内容を更新する

説明

bool **ifx_update_char** (int $bid , string $content)

指定した文字オブジェクト bid に関して 文字オブジェクトの内容を更新します。

パラメータ

bid

文字オブジェクト ID。

content

新規データの文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ifx_update_blob()

---

ifxus_close_slob

(No version information available, might be only in CVS)

ifxus_close_slob — SLOB オブジェクトを削除する

説明

bool **ifxus_close_slob** (int $bid)

指定した SLOB オブジェクト ID bid の SLOB オブジェクトを削除します。

パラメータ

bid

既存の SLOB の ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ifxus_open_slob()

---

ifxus_create_slob

(No version information available, might be only in CVS)

ifxus_create_slob — SLOB オブジェクトを作成し、オープンする

説明

int **ifxus_create_slob** (int $mode)

SLOB オブジェクトを作成し、オープンします。

パラメータ

mode

IFX_LO_RDONLY、 **IFX_LO_WRONLY**、**IFX_LO_APPEND**、 **IFX_LO_RDWR**、**IFX_LO_BUFFER**、 **IFX_LO_NOBUFFER** の組み合わせ。

返り値

新しい SLOB オブジェクトの ID、あるいはエラー時に **FALSE** を返します。

参考

- ifxus_close_slob()
- ifxus_free_slob()

---

ifxus_free_slob

(No version information available, might be only in CVS)

ifxus_free_slob — SLOB オブジェクトを削除する

説明

bool **ifxus_free_slob** (int $bid)

SLOB オブジェクトを削除します。

パラメータ

bid

既存の SLOB の ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ifxus_close_slob()

---

ifxus_open_slob

(No version information available, might be only in CVS)

ifxus_open_slob — SLOB オブジェクトをオープンする

説明

int **ifxus_open_slob** (int $bid , int $mode)

SLOB オブジェクトをオープンします。 bid は存在する SLOB ID である必要があります。

パラメータ

bid

既存の SLOB の ID。

mode

IFX_LO_RDONLY、 **IFX_LO_WRONLY**、**IFX_LO_APPEND**、 **IFX_LO_RDWR**、**IFX_LO_BUFFER**、 **IFX_LO_NOBUFFER** の組み合わせ。

返り値

新しい SLOB オブジェクトの ID、あるいはエラー時に **FALSE** を返します。

参考

- ifxus_close_slob()
- ifxus_free_slob()

---

ifxus_read_slob

(No version information available, might be only in CVS)

ifxus_read_slob — SLOB オブジェクトから n バイト読みこむ

説明

string **ifxus_read_slob** (int $bid , int $nbytes)

SLOB オブジェクトから nbytes バイトを読みこみます。

パラメータ

bid

既存の SLOB の ID。

nbytes

読み込むバイト数。

返り値

SLOB の内容を表す文字列、あるいはエラー時に **FALSE** を返します。

参考

- ifxus_write_slob()

---

ifxus_seek_slob

(No version information available, might be only in CVS)

ifxus_seek_slob — 現在のファイル位置またはシーク位置を返す

説明

int **ifxus_seek_slob** (int $bid , int $mode , int $offset)

オープンされた SLOB オブジェクトに関する 現在のファイルまたはシーク位置を設定します。

パラメータ

bid

既存の SLOB の ID。

mode

0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END.

offset

バイトオフセット。

返り値

シーク位置を表す整数値、あるいはエラー時に **FALSE** を返します。

参考

- ifxus_tell_slob()

---

ifxus_tell_slob

(No version information available, might be only in CVS)

ifxus_tell_slob — 現在のファイルまたはシーク位置を返す

説明

int ***ifxus_tell_slob*** *(int $bid)*

オープンされた SLOB オブジェクトに関して 現在のファイルまたはシーク位置を返します。

パラメータ

bid

既存の SLOB の ID。

返り値

シーク位置を表す整数値、あるいはエラー時に ***FALSE*** *を返します。*

参考

- *ifxus_seek_slob()*

---

ifxus_write_slob

(No version information available, might be only in CVS)

ifxus_write_slob — SLOB オブジェクトに文字列を書きこむ

説明

int ***ifxus_write_slob*** *(int $bid , string $content)*

SLOB オブジェクトを文字列に書きこみます。

パラメータ

bid

既存の SLOB の ID。

content

書き込む内容を表す文字列。

返り値

書き込んだバイト数を表す整数値、あるいはエラー時に ***FALSE*** *を返します。*

参考

- *ifxus_read_slob()*

目次

- ifx_num_rows — クエリから既に取得された行の数を数える
- ifx_pconnect — 持続的 Informix 接続をオープンする
- ifx_prepare — SQL 文を実行用に準備する
- ifx_query — Informix クエリを送信する
- ifx_textasvarchar — デフォルトのテキストモードを設定する
- ifx_update_blob — BLOB オブジェクトの内容を更新する
- ifx_update_char — 文字オブジェクトの内容を更新する
- ifxus_close_slob — SLOB オブジェクトを削除する
- ifxus_create_slob — SLOB オブジェクトを作成し、オープンする
- ifxus_free_slob — SLOB オブジェクトを削除する
- ifxus_open_slob — SLOB オブジェクトをオープンする
- ifxus_read_slob — SLOB オブジェクトから n バイト読みこむ
- ifxus_seek_slob — 現在のファイル位置またはシーク位置を返す
- ifxus_tell_slob — 現在のファイルまたはシーク位置を返す
- ifxus_write_slob — SLOB オブジェクトに文字列を書きこむ

Informix 関数 (PDO_INFORMIX)

導入

PDO_INFORMIX は、PHP から Informix データベースへのアクセスを可能にするための PHP Data Objects (PDO) インターフェース を実装したドライバです。

インストール手順

PDO_INFORMIX 拡張モジュールをビルドするには、PHP と同じシステムに Informix Client SDK 2.81 UC1 以降がインストールされている必要があります。 Informix Client SDK は、» IBM Informix Support Site で入手できます。

PDO_INFORMIX は » PECL 拡張モジュールです。 そのため、PDO_INFORMIX 拡張モジュールをインストールするには PECL 拡張モジュールのインストール の手順に従います。 Informix Client SDK ヘッダファイルおよびライブラリの場所を指定するため、 次のように **configure** コマンドを実行します。

```
bash$  ./configure  --with-pdo-informix=/path/to/SDK[,shared]
```

configure コマンドのデフォルトは、 環境変数 INFORMIXDIR の値となります。

スクロール可能なカーソル

PDO_INFORMIX は、スクロール可能なカーソルをサポートしています。 しかし、デフォルトでは有効になっていません。 スクロール可能なカーソルのサポートを有効にするには、 odbc.ini での対応する ODBC 接続設定で **ENABLESCROLLABLECURSORS=1** を設定するか、 DSN 接続文字列に **EnableScrollableCursors=1** 句を指定しなければなりません。

PDO_INFORMIX DSN

(No version information available, might be only in CVS)

PDO_INFORMIX DSN — Informix データベースに接続する

説明

PDO_INFORMIX データソース名 (DSN) は、Informix ODBC DSN 文字列を基にしています。 Informix ODBC DSN の設定についての詳細は、 » Informix Dynamic Server Information Center にあります。PDO_INFORMIX DSN の主要な部分は以下のようになります。

DSN 接頭辞

DSN 接頭辞は **informix:** です。

DSN

DSN は、odbc.ini を使用したデータソース設定、あるいは完全な » 接続文字列 のいずれかです。

例

Example#1 odbc.ini を使用した PDO_INFORMIX DSN の例

以下の例は、odbc.ini で Infdrv33 として登録されている Informix データベースに接続するための PDO_INFORMIX DSN を表します。

```
$db = new PDO("informix:DSN=Infdrv33", "", "");

[ODBC Data Sources]
Infdrv33=INFORMIX 3.3 32-BIT

[Infdrv33]
Driver=/opt/informix/csdk_2.81.UC1G2/lib/cli/iclis09b.so
Description=INFORMIX 3.3 32-BIT
Database=common_db
LogonID=testuser
```

```
pwd=testpass
Servername=ids_server
DB_LOCALE=en_US.819
OPTIMIZEAUTOCOMMIT=1
ENABLESCROLLABLECURSORS=1
```

Example#2 接続文字列を使用した PDO_INFORMIX DSN の例

以下の例は、**common_db** という名前の Informix データベースに接続文字列を使用して接続するための PDO_INFORMIX DSN を表します。

```
$db = new PDO("informix:host=host.domain.com; service=9800;
      database=common_db; server=ids_server; protocol=onsoctcp;
      EnableScrollableCursors=1", "testuser", "tespass");
```

Ingres II 関数

導入

以下の関数により、Ingres II データベースサーバにアクセスできるように なります。

> **注意**: 既に他のデータベースサーバにアクセスするための PHP 拡張モジュールを 使用している場合、Ingres は同一のコネクションについて複数のクエリや トランザクションの並列実行ができない、つまり、この拡張モジュールに おいて結果およびトランザクションのハンドルを得ることができないことに 注意する必要があります。クエリの結果は、別のクエリを送信する前に 処理する必要があり、トランザクションは別のトランザクションを オープンする前にコミットまたはロールバックする必要があります （これは、最初のクエリを送信する際に自動的に行われます）。

要件

Ingres サポートを有効にして PHP をコンパイルするには、 Ingres OpenAPI ライブラリとヘッダファイルが必要です。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/ingres.

これらの関数を利用可能とするには、オプション --with-ingres[=DIR] を付け、 Ingres サポートを有効にして PHP をコンパイルする必要があります。ただし、DIR は Ingres ベースディレクトリで、/II/ingres が デフォルトです。環境変数 II_SYSTEM が正しく設定されていない場合、Ingres をインストールしたディレクトリを指定するために --with-ingres=DIR を使用することが 必要となる可能性があります。

この拡張モジュールを Apache で使用する際に Apache が起動せず、 "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0" というエラーが発生する場合、 環境変数 II_SYSTEM が正確に設定されているかどうかを確認してください。 "export II_SYSTEM="/home/ingres/II" を Apache を開始するスクリプトに 追加してください。この後に httpd を起動すればうまくいくはずです。

実行時設定

php.ini の設定により動作が変化します。

Ingres 設定オプション

名前	デフォルト	変更の可否	変更履歴
ingres.allow_persistent	"1"	PHP_INI_SYSTEM	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
ingres.max_persistent	"-1"	PHP_INI_SYSTEM	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
ingres.max_links	"-1"	PHP_INI_SYSTEM	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
ingres.default_database	NULL	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
ingres.default_user	NULL	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
ingres.default_password	NULL	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
ingres.report_db_warnings	"1"	PHP_INI_ALL	ingres 1.1 以降で使用可能です。
ingres.cursor_mode	"0"	PHP_INI_ALL	ingres 1.1 以降で使用可能です。
ingres.blob_segment_length	"4096"	PHP_INI_ALL	ingres 1.2.0 以降で使用可能です。
ingres.trace_connect	"0"	PHP_INI_ALL	ingres 1.2.1 以降で使用可能です。
ingres.timeout	"-1"	PHP_INI_ALL	ingres 1.4.0 以降で使用可能です。
ingres.array_index_start	"1"	PHP_INI_ALL	ingres 1.4.0 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

ingres_connect() および ingres_pconnect() は Ingres II リンク ID を返します。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

INGRES_ASSOC (integer)
フィールド名をキーとする配列でカラムデータを返します。

INGRES_NUM (integer)
数値添字の配列でカラムデータを返します。 結果の最初のフィールドの添字は 1 となります。
INGRES_BOTH (integer)
数値添字・フィールド名キーの両方の形式でカラムデータを返します。
INGRES_EXT_VERSION (string)
Ingres 拡張モジュールのバージョンを表します。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_API_VERSION (integer)
この拡張モジュールがビルドされた際の Ingres OpenAPI のバージョンを表します。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_CURSOR_READONLY (integer)
Ingres カーソルが 'readonly' モードでオープンされていることを表します。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_CURSOR_UPDATE (integer)
Ingres カーソルが 'update' モードでオープンされていることを表します。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_MULTINATIONAL (integer)
II_DATE_FORMAT の設定 MULTINATIONAL と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_MULTINATIONAL4 (integer)
II_DATE_FORMAT の設定 MULTINATIONAL4 と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_FINNISH (integer)
II_DATE_FORMAT の設定 FINNISH と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_ISO (integer)
II_DATE_FORMAT の設定 ISO と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_ISO4 (integer)
II_DATE_FORMAT の設定 ISO4 と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_GERMAN (integer)
II_DATE_FORMAT の設定 GERMAN と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_MDY (integer)
II_DATE_FORMAT の設定 MDY と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_DMY (integer)
II_DATE_FORMAT の設定 DMY と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_DATE_YMD (integer)
II_DATE_FORMAT の設定 YMD と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_MONEY_LEADING (integer)
金額の値の先頭に表示する文字を表します。 II_MONEY_FORMAT の設定 'L:' と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_MONEY_TRAILING (integer)
金額の値の最後に表示する文字を表します。 II_MONEY_FORMAT の設定 'T:' と等価です。 PECL 拡張モジュールのバージョン 1.2.0 以降で使用可能です。
INGRES_STRUCTURE_BTREE (integer)
デフォルトのテーブル構造あるいはインデックス構造を BTREE に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。
INGRES_STRUCTURE_CBTREE (integer)
デフォルトのテーブル構造あるいはインデックス構造を COMPRESSED BTREE に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。
INGRES_STRUCTURE_HASH (integer)
デフォルトのテーブル構造あるいはインデックス構造を HASH に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。
INGRES_STRUCTURE_CHASH (integer)
デフォルトのテーブル構造あるいはインデックス構造を COMPRESSED HASH に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。
INGRES_STRUCTURE_HEAP (integer)
デフォルトのテーブル構造あるいはインデックス構造を HEAP に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。
INGRES_STRUCTURE_CHEAP (integer)
デフォルトのテーブル構造あるいはインデックス構造を COMPRESSED HEAP に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。
INGRES_STRUCTURE_ISAM (integer)
デフォルトのテーブル構造あるいはインデックス構造を ISAM に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。
INGRES_STRUCTURE_CISAM (integer)
デフォルトのテーブル構造あるいはインデックス構造を COMPRESSED ISAM に設定します。 これは、接続時のオプション table_structure あるいは index_structure と組み合わせて使用します。 PECL 拡張モジュールのバージョン 1.4.0 以降で使用可能です。

例

この例では、Ingres データベースに対する接続、クエリの実行、 結果の表示および接続解除の方法を説明します。

Example#1 シンプルな Ingres の例

```
<?php
// 接続し、データベースを選択します
$link = ingres_connect('database', 'user', 'password')
    or die('接続できなせんでした: ' . ingres_error($link));
echo '接続に成功しました';

// すべての Ingres データベース内に存在するテーブルから select します
$query = 'SELECT * FROM iirelation';
$returncode = ingres_query($query,$link) or die('問い合わせに失敗しました: ' .
ingres_error($link));

// 結果を HTML で出力します
// relid - テーブル名
// relowner - テーブルの所有者
echo "<table>¥n";
while ($iirelation = ingres_fetch_object(INGRES_BOTH, $link)) {
    echo "¥t<tr>¥n";
    echo "¥t¥t<td>" . $iirelation->relid . "</td>¥n";
    echo "¥t¥t<td>" . $iirelation->relowner . "</td>¥n";
    echo "¥t</tr>¥n";
}
echo "</table>¥n";

// トランザクションをコミットします
ingres_commit($link);
// 接続を閉じます
ingres_close($link);
?>
```

---

ingres_autocommit

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_autocommit — autocommit をオンまたはオフに切替える

説明

bool **ingres_autocommit** ([resource $link])

ingres_autocommit() は、サーバの "autocommit" モードをオンまたはオフに切替えるために、トランザクションをオープンする前（最初に ingres_query() をコールする前、または ingres_rollback() あるいは ingres_commit() をコールした直後）にコールされます（スクリプトの実行開始には、autocommit モードはオフです）。

autocommit モードがオンの場合、各クエリはサーバにより自動的にコミットされます。 これは、ingres_query() をコールした後に常に ingres_commit() をコールすることと等価です。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ingres_query()
- ingres_rollback()
- ingres_commit()

---

ingres_close

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_close — Ingres II データベース接続を閉じる

説明

bool **ingres_close** ([resource $link])

ingres_close() は、指定したリンクが指す Ingres サーバへの接続をクローズします。パラメータ

持続的接続をクローズすることはできず、 持続的でない接続はスクリプトの終了時に自動的にクローズされるため、 **ingres_close()** は通常は不要です。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ingres_connect()
- ingres_pconnect()

---

ingres_commit

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_commit — トランザクションをコミットする

説明

bool **ingres_commit** ([resource $link])

ingres_commit() は、 現在オープンしているトランザクションをコミットし、 全ての変更をデータベースに保存します。

この関数はトランザクションをクローズします。 ingres_query() によりクエリを送信することで、 新規のトランザクションをオープンすることが可能です。

トランザクションをオープンする前に ingres_autocommit() をコールすることにより、 各クエリの後に自動的にサーバ側でコミットを行うことも可能です。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ingres_query()
- ingres_rollback()
- ingres_autocommit()

---

ingres_connect

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_connect — Ingres データベースへの接続をオープンする

説明

resource **ingres_connect** ([string $database [, string $username [, string $password [, array $options]]]])

ingres_connect() は、指定した Ingres database との接続をオープンします。

この接続は、スクリプトの実行終了時または、このリンクについて ingres_close() がコールされた場合に クローズされます。

他の全ての Ingres 関数は、デフォルトで直近にオープンされたリンクを使用します。 このため、複数のリンクを同時に使用したい場合には、 返された値を保存しておく必要があります。

パラメータ

いくつかのパラメータが欠けている場合、 **ingres_connect()** は、 ingres.default_database 、 ingres.default_user 、 ingres.default_password に関して php.ini の値を使用します。

database

データベース名。 [node_id::]dbname[/svr_class] 形式でなければなりません。

username

Ingres のユーザ名。

password

username のパスワード。

options

ingres_connect() のオプション

オプション名	説明
date_century_boundary	2桁で指定した年を、今世紀と判断するか来世紀と判断するかの閾値。 II_DATE_CENTURY_BOUNDARY と同等です。
group	ユーザのグループ ID を指定します。'-G' フラグと同等です。
role	アプリケーションのロール ID。ロールのパスワードが必要な場合は、 このパラメータは "role/password" 形式で指定する必要があります。
effective_user	使用する ingres ユーザアカウント。'-u' フラグと同等です。
dbms_password	ユーザが Ingres に接続する際に使用する、内部データベースのパスワード。
table_structure	新規テーブルのデフォルトの構造。以下のいずれかが指定可能です。 • INGRES_STRUCTURE_BTREE • INGRES_STRUCTURE_HASH • INGRES_STRUCTURE_HEAP • INGRES_STRUCTURE_ISAM • INGRES_STRUCTURE_CBTREE • INGRES_STRUCTURE_CISAM • INGRES_STRUCTURE_CHASH • INGRES_STRUCTURE_CHEAP
index_structure	新規セカンダリインデックスのデフォルトの構造。以下のいずれかが指定可能です。 • INGRES_STRUCTURE_CBTREE • INGRES_STRUCTURE_CISAM • INGRES_STRUCTURE_CHASH • INGRES_STRUCTURE_BTREE • INGRES_STRUCTURE_HASH • INGRES_STRUCTURE_ISAM
login_local	対象データベースの文字列に VNODE が含まれていた場合に ユーザ ID とパスワードをどのように使用して接続するかを指定します。 TRUE に設定すると、ユーザ ID とパスワードで VNODE にローカルアクセスし、 VNODE のログイン情報を使用してデータベースとの接続を確立します。 FALSE に設定すると、プロセスのユーザ ID を使用して VNODE にアクセスし、 接続用のユーザ ID とパスワードを VNODE のログイン情報の代わりに使用して データベースとの接続を確立します。対象データベースの文字列に VNODE が含まれていない場合は、このパラメータは無視されます。 デフォルトは FALSE です。
timezone	このセッションのタイムゾーンを設定します。設定しなかった場合は II_TIMEZONE_NAME の値が使用されます。II_TIMEZONE_NAME も設定されていない場合は、NA-PACIFIC (GMT-8 に夏時間を適用) が用いられます。

オプション名	説明
date_format	Ingres の日付として指定可能な入出力書式を設定します。 デフォルトは II_DATE_FORMAT で指定された値です。 II_DATE_FORMAT が設定されていない場合は、デフォルトは US の日付フォーマット、すなわち mm/dd/yy となります。 以下のいずれかが使用可能です。 • INGRES_DATE_DMY • INGRES_DATE_FINISH • INGRES_DATE_GERMAN • INGRES_DATE_ISO • INGRES_DATE_ISO4 • INGRES_DATE_MDY • INGRES_DATE_MULTINATIONAL • INGRES_DATE_MULTINATIONAL4 • INGRES_DATE_YMD • INGRES_DATE_US
decimal_separator	小数点を表す文字。
money_lort	通貨記号を先頭あるいは末尾のどちらにつけるか。以下のいずれかが使用可能です。 • INGRES_MONEY_LEADING • INGRES_MONEY_TRAILING
money_sign	MONEY データ型で使用する通貨記号。
money_precision	MONEY データ型の精度。
float4_precision	FLOAT4 データ型の精度。
float8_precision	FLOAT8 データの精度。
blob_segment_length	BLOB/CLOB データを読み込む際に、一度に取得するデータのバイト数。 明示的に指定しなかった場合、デフォルトは 4096 バイトです。

返り値

成功時に Ingres リンクリソース、失敗した際に **FALSE** を返します。

例

Example#1 ingres_connect() の例

```
<?php
$link = ingres_connect("mydb", "user", "pass")
    or die("接続できませんでした");
echo "接続に成功しました";
ingres_close($link);
?>
```

Example#2 デフォルトリンクを使用する ingres_connect() の例

```
<?php
ingres_connect("mydb", "user", "pass")
    or die("接続できませんでした");
echo "接続に成功しました";
ingres_close();
?>
```

参考

- ingres_pconnect()
- ingres_close()

---

ingres_cursor

(PECL ingres:1.1-1.4.3)

ingres_cursor — 指定したリンクリソースのカーソル名を取得する

説明

string **ingres_cursor** ([resource $link])

アクティブなカーソルの名前を含む文字列を返します。 アクティブなカーソルがない場合は NULL が返されます。

link リソースが **ingres_cursor()** に渡された場合、そのリンクに記録されたカーソル名を返します。 リンクが渡されなかった場合、**ingres_cursor()** はデフォルトのリンクに関連付けられたカーソル名を返します。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

アクティブなカーソルの名前を含む文字列を返します。 アクティブなカーソルがない場合は NULL が返されます。

例

Example#1 ingres_cursor() の例

```
<?php
$link = ingres_connect($database, $user, $password);
```

```
ingres_prepare("select * from table", $link);

$cursor_name = ingres_cursor($link);

echo $cursor_name;

?>
```

参考

- **ingres_prepare()**
- **ingres_execute()**

---

ingres_errno

(PECL ingres:1.1-1.4.3)

ingres_errno — 直近に発生した ingres エラー番号を取得する

説明

int **ingres_errno** ([resource $link])

直近のエラー番号を整数値で返します。エラーが発生していない場合は 0 を返します。

link リソースが **ingres_errno()** に渡された場合、そのリンクに記録された直近のエラーを返します。 リンクが渡されなかった場合、**ingres_errno()** はデフォルトのリンクを使用して直近のエラーを返します。

この関数 **ingres_errno()** は、 データベースクエリを実行した直後にコールしなければなりません。 **ingres_errno()** の前に他の関数がコールされると、 直近の Ingres 関数のコールで発生したエラーコードが書き換えられてしまいます。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

直近のエラー番号を整数値で返します。エラーが発生していない場合は 0 を返します。

例

Example#1 ingres_errno() の例

```
<?php
$link = ingres_connect($database, $user, $password);

ingres_query("select * from table", $link);

$error_code = ingres_errno($link);

if ( $error_code != 0 ) {
   echo "エラーが発生しました - " . $error_code;
}
?>
```

参考

- ingres_error()
- ingres_errsqlstate()

---

ingres_error

(PECL ingres:1.1-1.4.3)

ingres_error — 直近に発生したエラーのエラーメッセージを取得する

説明

string **ingres_error** ([resource $link])

直近のエラーの内容を文字列で返します。エラーが発生していない場合は NULL を返します。

link リソースが **ingres_error()** に渡された場合、そのリンクに記録された直近のエラーを返します。 リンクが渡されなかった場合、**ingres_error()** はデフォルトのリンクを使用して直近のエラーを返します。

この関数 **ingres_error()** は、 データベースクエリを実行した直後にコールしなければなりません。 **ingres_error()** の前に他の関数がコールされると、 直近の Ingres 関数のコールで発生したエラーメッセージが書き換えられてしまいます。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

直近のエラーの内容を文字列で返します。エラーが発生していない場合は NULL を返します。

例

Example#1 ingres_error() の例

```
<?php
ingres_connect($database, $user, $password);

ingres_query("select * from table");

$error_text = ingres_error();

if (!is_null($error_text)) {
   echo "エラーが発生しました - " . $error_text;
}
?>
```

参考

- ingres_errno()
- ingres_errsqlstate()

---

ingres_errsqlstate

(PECL ingres:1.1-1.4.3)

ingres_errsqlstate — 直近に発生した SQLSTATE エラーコードを取得する

説明

string **ingres_errsqlstate** ([resource $link])

直近の SQLSTATE を文字列で返します。エラーが発生していない場合は NULL を返します。

link リソースが **ingres_errsqlstate()** に渡された場合、そのリンクに記録された直近のエラーを返します。 リンクが渡されなかった場合、**ingres_errsqlstate()** はデフォルトのリンクを使用して直近のエラーを返します。

この関数 **ingres_errsqlstate()** は、 データベースクエリを実行した直後にコールしなければなりません。 **ingres_errsqlstate()** の前に他の関数がコールされると、 直近の Ingres 関数のコールで発生したエラーメッセージが書き換えられてしまいます。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

直近の SQLSTATE を文字列で返します。エラーが発生していない場合は NULL を返します。

例

Example#1 ingres_errsqlstate() の例

```
<?php
ingres_connect($database, $user, $password);

ingres_query("select * from table");

$error_sqlstate = ingres_errsqlstate();

if (!is_null($error_sqlstate)) {
   echo "エラーが発生しました - " . $error_sqlstate;
}
?>
```

参考

- ingres_errno()
- ingres_error()

---

ingres_fetch_array

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_fetch_array — 1 行分の結果を配列に取得する

説明

```
array ingres_fetch_array ([ int $result_type [, resource $link ]] )
```

この関数は、ingres_fetch_row() の拡張版です。 結果として返される配列の数値添字にデータを保存するだけでなく、 フィールド名をキーとして連想配列にもデータが保存されます。

結果において複数のカラムが同じフィールド名を有している場合、後の カラムが優先されます。同名の他のカラムにアクセスするには、カラムの 添字番号を使用するか、カラムのエイリアスを作成する必要があります。

```
<?php

ingres_query("select t1.f1 as foo t2.f1 as bar from t1, t2");
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];

?>
```

速度面では、この関数は ingres_fetch_object() と同じで、ingres_fetch_row() とほぼ同等です（差は僅かです）。

パラメータ

result_type

result_type には、数値添字の場合に INGRES_NUM、連想配列の場合にINGRES_ASSOC、両方の場合に INGRES_BOTH(デフォルト)を指定可能です。

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

取得したレコード(行)に対応する配列を返します。 レコードがもうない場合は **FALSE** を返します。

例

Example#1 ingres_fetch_array() の例

```
<?php
ingres_connect($database, $user, $password);

ingres_query("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  //  連想配列を使用する
    echo $row["fullname"];
    echo $row[1];          //  数値添字配列を使用する
    echo $row[2];
}
?>
```

参考

- ingres_query()
- ingres_num_fields()
- ingres_field_name()
- ingres_fetch_object()
- ingres_fetch_row()

---

ingres_fetch_object

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_fetch_object — 1 行分の結果をオブジェクトとして取得する

説明

```
object ingres_fetch_object ([ int $result_type [, resource $link ]] )
```

この関数は ingres_fetch_array() に似ていますが、 配列の代わりにオブジェクトが返されるという違いが一つあります。 間接的なアクセス、つまり、オフセットではなくフィールド名によりデータを アクセスすることのみが可能です(数値はプロパティ名としては使用 できません)。

速度の面では、この関数は ingres_fetch_array() と等価であり、ingres_fetch_row() とほぼ同等です（違いは僅かです）。

パラメータ

result_type

オプションの引数 result_type は定数であり、 次の値のどれかとなります: INGRES_ASSOC, INGRES_NUM, INGRES_BOTH

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

取得したレコード(行)をオブジェクトとして返します。 レコードがもうない場合は **FALSE** を返します。

例

Example#1 ingres_fetch_object() の例

```
<?php
ingres_connect($database, $user, $password);
ingres_query("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

参考

- ingres_query()
- ingres_num_fields()
- ingres_field_name()
- ingres_fetch_array()
- ingres_fetch_row()

---

ingres_fetch_row

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_fetch_row — 1 行分の結果を数値添字配列として取得する

説明

array **ingres_fetch_row** ([resource $link])

ingres_fetch_row() は取得したレコード(行)を有 する配列を返します。レコードがもうない場合は **FALSE** を返します。各 カラムは、オフセット1から始まる配列オフセットに保存されます。

ingres_fetch_row() を連続的にコールした場合、 結果集合の中の次のレコードが返され、もうレコードがない場合は**FALSE** を返します。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

取得したレコード(行)を有する配列を返します。 レコードがもうない場合は **FALSE** を返します。

例

Example#1 ingres_fetch_row() の例

```
<?php
ingres_connect($database, $user, $password);

ingres_query("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

参考

- ingres_num_fields()
- ingres_query()
- ingres_fetch_array()
- ingres_fetch_object()

---

ingres_field_length

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_field_length — フィールド長を得る

説明

int **ingres_field_length** (int $index [, resource $link])

ingres_field_length() はフィールド長を返します。 フィールド長は、フィールドをサーバに保存する際に使用されるバイト数です。 詳細な情報については、Ingres/OpenAPI User Guide の Appendix C を参照ください。

パラメータ

index

index はフィールド番号であり、*1* と *ingres_num_fields()* で指定した値の間である必要があります。

link

接続リンク *ID*。省略した場合は、最後にオープンしたリンクを使用します。

返り値

フィールドの長さを返します。

参考

- *ingres_query()*
- *ingres_fetch_array()*
- *ingres_fetch_object()*
- *ingres_fetch_row()*

---

ingres_field_name

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_field_name — クエリ結果においてフィールド名を得る

説明

string ***ingres_field_name*** *(int $index [, resource $link])*

ingres_field_name() は、クエリ結果のフィールド名を返します。

パラメータ

index

index はフィールド番号であり、*1* と *ingres_num_fields()* で指定した値の間である必要があります。

link

接続リンク *ID*。省略した場合は、最後にオープンしたリンクを使用します。

返り値

クエリ結果のフィールド名を返します。 失敗した場合は ***FALSE*** を返します。

参考

- *ingres_query()*
- *ingres_fetch_array()*
- *ingres_fetch_object()*
- *ingres_fetch_row()*

---

ingres_field_nullable

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_field_nullable — フィールドに ***NULL*** 値を設定可能かどうか調べる

説明

bool ***ingres_field_nullable*** *(int $index [, resource $link])*

フィールドに *null* を設定可能かどうかを調べます。

パラメータ

index

index はフィールド番号であり、*1* と *ingres_num_fields()* で指定した値の間である必要があります。

link

接続リンク *ID*。省略した場合は、最後にオープンしたリンクを使用します。

返り値

ingres_field_nullable() は、フィールドに ***NULL*** が設定可能な場合に ***TRUE***、設定できない場合に ***FALSE*** を返します。

参考

- *ingres_query()*
- *ingres_fetch_array()*

- ingres_fetch_object()
- ingres_fetch_row()

---

ingres_field_precision

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_field_precision — フィールドの精度を得る

説明

int **ingres_field_precision** (int $index [, resource $link])

ingres_field_precision() はフィールドの精度を返します。 この値は、decimal、float、SQLデータ money 型でのみ使用されます。 詳細な情報については、Ingres/OpenAPI User Guide の Appendix C を参照ください。

パラメータ

index

index はフィールド番号であり、1 と ingres_num_fields() で指定した値の間である必要があります。

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

フィールドの精度を返します。

参考

- ingres_query()
- ingres_fetch_array()
- ingres_fetch_object()
- ingres_fetch_row()

---

ingres_field_scale

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_field_scale — フィールドのスケールを得る

説明

int **ingres_field_scale** (int $index [, resource $link])

ingres_field_scale() はフィールドのスケールを返します。 この値は、SQLデータ decimal 型でのみ使用されます。 詳細な情報については、Ingres/OpenAPI User Guide の Appendix C を参照ください。

パラメータ

index

index はフィールド番号であり、1 と ingres_num_fields() で指定した値の間である必要があります。

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

フィールドのスケールを返します。

参考

- ingres_query()
- ingres_fetch_array()
- ingres_fetch_object()
- ingres_fetch_row()

---

ingres_field_type

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_field_type — クエリ結果においてフィールドの型を得る

説明

string **ingres_field_type** (int $index [, resource $link])

クエリ結果においてフィールドの型を取得します。

パラメータ

index

index はフィールド番号であり、1 と ingres_num_fields() で指定した値の間である必要があります。

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

ingres_field_type() はクエリ結果のフィールド型、 失敗した場合は、**FALSE** を返します。返される型は、例えば、 "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE" になります。 これらの型のいくつかは、フィールド長に応じて複数の SQL 型を マップすることが可能です(ingres_field_length() を参照ください)。例えば、"IIAPI_FLT_TYPE" は、float4 または float8 とすることが可能です。詳細な情報については、Ingres/OpenAPI User Guide の Appendix C を参照ください。

参考

- ingres_query()
- ingres_fetch_array()
- ingres_fetch_object()
- ingres_fetch_row()

---

ingres_num_fields

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_num_fields — 直近のクエリにより返されたフィールドの数を得る

説明

int **ingres_num_fields** ([resource $link])

ingres_num_fields() は、 ingres_query() をコールした後で、Ingres サーバにより返された結果のフィールド数を返します。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

フィールドの数を返します。

参考

- ingres_query()
- ingres_fetch_array()
- ingres_fetch_object()
- ingres_fetch_row()

---

ingres_num_rows

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_num_rows — 直近のクエリで作用されたレコードの数を取得し、返す

説明

int **ingres_num_rows** ([resource $link])

この関数は、主にデータベースで修正されたレコードの数を得る際に有用です。 この関数が ingres_fetch_array()、 ingres_fetch_object()、 ingres_fetch_row() を使用する前にコールされた場合、 サーバは結果のデータを削除し、スクリプトは結果を得ることができなくなります。

代わりにこれらの取得用関数のどれかをもう結果が残っていないという意味で **FALSE** を返すまでループ処理を行い、結果のデータを取得する必要があります。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

delete, insert, update クエリの場合、 **ingres_num_rows()** は、そのクエリにより作用された行 (レコード)の数を返します。その他のクエリの場合、 **ingres_num_rows()** はクエリ結果のレコード数を返します。

参考

- ingres_query()
- ingres_fetch_array()
- ingres_fetch_object()
- ingres_fetch_row()

---

ingres_pconnect

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_pconnect — Ingres II データベースへの持続的接続をオープンする

説明

resource **ingres_pconnect** ([string $database [, string $username [, string $password]]])

Ingres II データベースへの持続的接続をオープンします。

この関数と ingres_connect() の違いは次の 2 つだけです。 まず、接続時にこの関数は、まず同じパラメータ既にオープンされている (持続的な) リンクを探すことです。第2に、Ingres サーバへの接続は スクリプトの実行終了時にもクローズされないところです。代わりに、 リンクは、後で使用するためにオープンされたままとなります (ingres_close() は、 **ingres_pconnect()** により確立されたリンクを クローズしません)。このため、この型のリンクは「持続的(persistent)」 であると呼ばれます。

パラメータ

database

データベース名。 [node_id::]dbname[/svr_class] 形式でなければなりません。

username

Ingres のユーザ名。

password

username のパスワード。

返り値

成功した際に Ingres II リンクリソース、失敗した際に **FALSE** を返します。

参考

- ingres_connect()
- ingres_close()

---

ingres_query

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_query — Ingres II に SQL クエリを送信する

説明

bool **ingres_query** (string $query [, resource $link])

ingres_query() は、指定した query を Ingres サーバに送信します。

クエリは、現在オープンされているトランザクションの一部となります。 オープンされているトランザクションがない場合、 **ingres_query()** は新規のトランザクションをオープンします。 トランザクションをクローズするには、データベースへの変更をコミットする場合に ingres_commit() を、 これらの変更をキャンセルする場合に ingres_rollback() のどちらかを使用することが可能です。 スクリプト終了時に、全てのオープンされたトランザクションは (ingres_rollback() をコールすることにより) ロールバックされます。新規トランザクションをオープンする前に ingres_autocommit() を使用することも可能です。 この場合、各 SQL クエリは直ちにコミットされます。

パラメータ

query

有効な SQL クエリ (Ingres SQL リファレンスガイドを参照ください)。

次の型の SQL クエリは、この関数で送信できません。

- close (ingres_close() を参照)
- commit (ingres_commit() を参照)
- connect (ingres_connect() を参照)
- disconnect (ingres_close() を参照)
- get dbevent
- prepare to commit

- rollback (ingres_rollback() を参照)
- savepoint
- set autocommit (ingres_autocommit() を参照)
- カーソルに関係するクエリはサポートされていません

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 ingres_query() の例

```
<?php
ingres_connect($database, $user, $password);

ingres_query("select * from table");
while ($row = ingres_fetch_row()) {
     echo $row[1];
     echo $row[2];
}
?>
```

参考

- ingres_fetch_array()
- ingres_fetch_object()
- ingres_fetch_row()
- ingres_commit()
- ingres_rollback()
- ingres_autocommit()

---

ingres_rollback

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5, PECL ingres:1.0-1.4.3)

ingres_rollback — トランザクションをロールバックする

説明

bool **ingres_rollback** ([resource $link])

ingres_rollback() は現在オープンされているトランザクションをロールバックし、 トランザクションの間にデータベースに行われた全ての変更をキャンセルします。

この関数は、トランザクションをクローズします。 ingres_query() によりクエリを送信することによって、 新規のトランザクションをオープンすることが可能です。

パラメータ

link

接続リンク ID。省略した場合は、最後にオープンしたリンクを使用します。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- ingres_query()
- ingres_commit()
- ingres_autocommit()

目次

- ingres_fetch_object — 1 行分の結果をオブジェクトとして取得する
- ingres_fetch_row — 1 行分の結果を数値添字配列として取得する
- ingres_field_length — フィールド長を得る
- ingres_field_name — クエリ結果においてフィールド名を得る
- ingres_field_nullable — フィールドに NULL 値を設定可能かどうか調べる
- ingres_field_precision — フィールドの精度を得る
- ingres_field_scale — フィールドのスケールを得る
- ingres_field_type — クエリ結果においてフィールドの型を得る
- ingres_num_fields — 直近のクエリにより返されたフィールドの数を得る
- ingres_num_rows — 直近のクエリで作用されたレコードの数を取得し、返す
- ingres_pconnect — Ingres II データベースへの持続的接続をオープンする
- ingres_query — Ingres II に SQL クエリを送信する
- ingres_rollback — トランザクションをロールバックする

IRC Gateway 関数

導入

IRCG により、同時に接続した数千人のユーザに簡単に XML データを 放送することができます。IRCG は、オンラインゲームや Web チャットのような 強力、拡張可能で対話的なプラットフォームを構築する際に使用することが できます。 IRCG は、放送以外のモードもサポートしており、ヘルパーアプリケーションが 入力されたデータを再編集し、静的なファイルを cHTML (i-mode) ま たは WML (WAP) のような特定の形式で出力することができます。 これらの静的なファイルは、高性能な Web サーバにより配信されます。

v4 の時点では、IRCG は以下のプラットフォームで実行できます。

- AIX
- FreeBSD
- HP-UX
- Irix
- Linux
- Solaris
- Tru64
- Windows

注意: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。
PHP 5.1.0.

インストール手順

詳細な手順は、» http://www.schumann.cx/ircg/ にあります。 提供されているインストールスクリプトをただちに使用するとよいでしょう。

お勧めはしませんが、IRCG サポートをあなた自身で設定することも可能です。 ircg-config スクリプトへのパスを
--with-ircg-config=path/to/irc-config のように設定し、さらに --with-ircg を configure 行に追加します。

実行時設定

設定ディレクティブは定義されていません。

リソース型

定義済み定数

定数は定義されていません。

---

ircg_channel_mode

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_channel_mode — ユーザ用にチャンネルモードフラグを設定する

説明

bool **ircg_channel_mode** (resource $connection , string $channel , string $mode_spec , string $nick)

connection で接続されているサーバ上のチャンネル channel のモードフラグを設定します。 モードフラグは mode_spec で渡され、 nick で指定したユーザに対して適用されます。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

mode_spec

nick に適用するモード。

モードフラグは、モード文字の前にプラスあるいはマイナスの文字を つけることで、それぞれ設定あるいは解除します。たとえば オペレータ権限を与えるには '+o'、破棄するには '-o' を mode_spec として指定します。

nick

そのモードを適用するユーザ。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

---

ircg_disconnect

(PHP 4 >= 4.0.4, PHP 5 <= 5.0.5)

ircg_disconnect — サーバへの接続を閉じる

説明

*bool **ircg_disconnect** (resource $connection , string $reason)*

***ircg_disconnect()** は、事前に ircg_pconnect() で確立しているサーバへの接続 connection を閉じます。*

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

reason

切断時のメッセージ。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ircg_pconnect()*

---

ircg_eval_ecmascript_params

(PHP 4 >= 4.3.0, PHP 5 <= 5.0.5)

ircg_eval_ecmascript_params — JS エンコードされたパラメータの一覧をデコードする

説明

*array **ircg_eval_ecmascript_params** (string $params)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ircg_eval_ecmascript_params のソースファイルより。

```
/*
  * State 0: Looking for ' or digit
  * State 1: Assembling parameter inside '..'
  * State 2: After escape sign: Copies single char verbatim, go to 1
  * State 3: Assembling numeric para, no quotation
  * State 4: Looking for ",", skipping whitespace
 */
```

参考

- *ircg_lookup_format_messages()*

---

ircg_fetch_error_msg

(PHP 4 >= 4.0.7, PHP 5 <= 5.0.5)

ircg_fetch_error_msg — 直前の IRCG 処理からエラーを返す

説明

array **ircg_fetch_error_msg** (resource $connection)

ircg_fetch_error_msg() は、 接続が失敗した際のエラーを返します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

返り値

配列の最初の要素がエラーコード、2 番目の要素がエラー文字列となります。

エラーコードの内容は、RFC 2812 で定義されている IRC 応答コードと同じです。

例

Example#1 ircg_fetch_error_msg() の例

```
<?php
if (!ircg_join ($id, "#php")) {
    $error = ircg_fetch_error_msg($id);
    echo "チャンネル #php に参加できません。エラーコード:
            $error[0] Description: $error[1]";
}
?>
```

---

ircg_get_username

(PHP 4 >= 4.0.7, PHP 5 <= 5.0.5)

ircg_get_username — 接続用にユーザ名を取得する

説明

string **ircg_get_username** (resource $connection)

関数 **ircg_get_username()** は、 指定した接続 connection のユーザ名を返します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

返り値

ユーザ名、あるいはエラー時に **FALSE** を返します。

---

ircg_html_encode

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_html_encode — HTML で保存された文字列をエンコードする

説明

string **ircg_html_encode** (string $html_string [, bool $auto_links [, bool $conv_br]])

HTML 文字列 html_string を、 出力用にエンコードします。これは、 IRC リンクからやってくるデータを再フォーマットするために IRCG 拡張モジュールで内部的に使用されているインターフェースを公開したものです。 この関数は IRC の色/フォントコードを HTML 形式にエンコードし、 適切なエンティティにエスケープします。

パラメータ

html_string

エンコードしたい HTML 文字列。

auto_links

conv_br

返り値

エンコードした HTML 文字列を返します。

---

ircg_ignore_add

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_ignore_add — サーバ上の除外リストにユーザを追加する

説明

void **ircg_ignore_add** (resource $connection , string $nick)

この関数は、接続 connection の除外リストにユーザ nick を追加します。 接続中にこのユーザから送信されたあらゆるメッセージは抑制されます。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

nick

除外するユーザ。

返り値

値を返しません。

参考

- ircg_ignore_del()

---

ircg_ignore_del

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_ignore_del — サーバ上の除外リストからユーザを削除する

説明

bool **ircg_ignore_del** (resource $connection , string $nick)

この関数は、接続 connection の IRCG 除外リストからユーザ nick を削除します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

nick

ユーザのニックネーム。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ircg_ignore_add()

---

ircg_invite

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5)

ircg_invite — ユーザをチャンネルに招待する

説明

bool **ircg_invite** (resource $connection , string $channel , string $nickname)

ircg_invite() は、ユーザ nickname に対してチャンネル channel に参加する招待状を送ります。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

nickname

ユーザのニックネーム。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ircg_is_conn_alive

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_is_conn_alive — 接続ステータスを確認する

説明

bool **ircg_is_conn_alive** (resource $connection)

ircg_is_conn_alive() は、 connection の状態を調べます。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

返り値

connection が有効で正常に動作している場合に **TRUE**、 何らかの理由で接続が死んでいる場合に **FALSE** を返します。

---

ircg_join

(PHP 4 >= 4.0.4, PHP 5 <= 5.0.5)

ircg_join — 接続中のサーバ上でチャンネルに接続する

説明

bool **ircg_join** (resource $connection , string $channel [, string $key])

connection で接続したサーバ上のチャンネル channel に参加します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

key

オプションで指定する、ルームのパスワード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ircg_kick

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_kick — サーバ上のチャンネルからユーザを排除する

説明

bool **ircg_kick** (resource $connection , string $channel , string $nick , string $reason)

connection で接続したサーバ上のチャンネル channel から、ユーザ nick を追い出します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

nick

追い出したいユーザ。

reason

なぜこの操作を実行したのか、簡単なメッセージを reason で指定しなければなりません。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

---

ircg_list

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5)

ircg_list — チャンネル内のトピック/ユーザの数を取得する

説明

*bool **ircg_list** (resource $connection , string $channel)*

***ircg_list()** は、channel の中のユーザ数を要求します。結果は、 ircg_set_file() あるいは ircg_set_current() で定義した出力に返されます。*

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 ircg_list() の例

```php
<?php

// サーバに接続します
$id = ircg_pconnect($nickname, $ip, $port);

// 出力をファイルに設定します
ircg_set_file($id, 'irc_output.html');

// チャンネルに参加します
if (!ircg_join($id, $channel)) {
    echo "Cannot /join $channel<br />";
}

// list コマンドを送信します
ircg_list($id, $channel);

// 出力が返されるのを待ちます
sleep(5);

// 切断します
ircg_disconnect($id,'Bye World');

// すべてを出力します
readfile('irc_output.html');

?>
```

上の例の出力は、たとえば以下のようになります。

```
...
Channel #channel has n users and the topic is 'Topic'
End of LIST
...
```

参考

- ircg_set_file()
- ircg_set_current()
- ircg_who()

---

ircg_lookup_format_messages

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_lookup_format_messages — フォーマットメッセージセットの存在を調べる

説明

bool ***ircg_lookup_format_messages*** *(string $name)*

フォーマットメッセージセット name が存在するかどうかを確かめます。フォーマットメッセージセットは ircg_register_format_messages() で登録し、 デフォルトのセット名 ircg が常に使用可能です。

パラメータ

name

フォーマットメッセージセットの名前。

返り値

セットが存在する場合に ***TRUE****、それ以外の場合に* ***FALSE*** *を返します。*

参考

- *ircg_register_format_messages()*

---

ircg_lusers

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5)

ircg_lusers — IRC ネットワーク統計

説明

bool ***ircg_lusers*** *(resource $connection)*

ircg_lusers() *は、connection で接続したネットワーク接続上のユーザの統計情報を要求します。 結果は、 ircg_set_file() あるいは ircg_set_current() で定義した出力に返されます。*

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *ircg_set_file()*
- *ircg_set_current()*
- *ircg_who()*

---

ircg_msg

(PHP 4 >= 4.0.4, PHP 5 <= 5.0.5)

ircg_msg — サーバ上のチャンネルまたはユーザにメッセージを送信する

説明

bool ***ircg_msg*** *(resource $connection , string $recipient , string $message [, bool $suppress])*

ircg_msg() *は、 サーバ上のチャンネルあるいはユーザにメッセージを送信します。*

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

recipient

recipient の先頭に # あるいは & をつけると message をチャンネルに送信し、 それ以外の場合はユーザ名として解釈されます。

message

送信するメッセージ。

suppress

オプションのパラメータ suppress を ***TRUE*** *にすると、あなた自身が送信したメッセージの connection 上での表示を抑制します。 この loopback といわれている機能が必要になるのは、IRC サーバが PRIVMSG コマンドの結果を返さないからです。*

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ircg_names

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5)

ircg_names — ユーザ名を問い合わせる

説明

bool **ircg_names** (int $connection , string $channel [, string $target])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

target

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ircg_get_username()
- ircg_lusers()

---

ircg_nick

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_nick — サーバ上のニックネームを変更する

説明

bool **ircg_nick** (resource $connection , string $nick)

指定した connection 上でのニックネームを変更します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

nick

現在のユーザの新しいニックネーム。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ircg_nickname_escape

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

ircg_nickname_escape — ニックネームの中の特別な文字が IRC 互換となるようにエンコードする

説明

string **ircg_nickname_escape** (string $nick)

関数 **ircg_nickname_escape()** は、 nick で指定したニックネームを IRC 互換形式にエンコードした結果を返します。

パラメータ

nick

返り値

IRC 互換形式のニックネームを返します。

参考

- ircg_nickname_unescape()

---

ircg_nickname_unescape

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

ircg_nickname_unescape — エンコードされたニックネームをデコードする

説明

string **ircg_nickname_unescape** (string $nick)

関数 **ircg_nickname_unescape()** は、 nick で指定したニックネームをデコードした結果を返します。

パラメータ

nick

エンコードされたニックネーム。

返り値

デコードしたニックネームを返します。

参考

- ircg_nickname_escape()

---

ircg_notice

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_notice — サーバ上のユーザに通知を送信する

説明

bool **ircg_notice** (resource $connection , string $nick , string $message)

この関数は、connection で接続したサーバ上のユーザ nick に message を送信します。他のメッセージ型と異なり、 IRC サーバやその他のソフトウェアは NOTICE については返信を自動生成しません。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

nick

対象となるユーザ。

message

メッセージの内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ircg_oper

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5)

ircg_oper — 権限を IRC OPER に昇格させる

説明

bool **ircg_oper** (resource $connection , string $name , string $password)

ircg_oper() は、connection 上にログインしているユーザに IRC オペレータの権限を与えます。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

name

オペレータのアカウント名。登録されている IRC アカウントと一致する必要があります。

password

name のパスワード。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

---

ircg_part

(PHP 4 >= 4.0.4, PHP 5 <= 5.0.5)

ircg_part — サーバ上のチャンネルから離脱する

説明

*bool **ircg_part** (resource $connection , string $channel)*

connection で接続したサーバ上のチャンネル channel から離脱します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

---

ircg_pconnect

(PHP 4 >= 4.0.4, PHP 5 <= 5.0.5)

ircg_pconnect — IRC サーバに接続する

説明

*resource **ircg_pconnect** (string $username [, string $server_ip [, int $server_port [, string $msg_format [, array $ctcp_messages [, array $user_settings [, bool $bailout_on_trivial]]]]])*

***ircg_pconnect()** は、 IRC サーバへの接続を確立しようと試みます。*

パラメータ

username

サーバ上での初期ニックネーム。

server_ip

IRC サーバのアドレス。

server_ip は数値表現の IP アドレスしか受け付けません。 DNS 検索はコストがかかる処理で、IRCG で行うべきです。 デフォルトは 127.0.0.1 です。

server_port

サーバのポート番号。デフォルトは 6667 です。

msg_format

事前に ircg_register_format_messages() で作成したフォーマットメッセージセットの名前を msg_format に指定することで、 IRC のメッセージやイベントの出力をカスタマイズすることが可能です。

ctcp_messages

ACTION (/me) のような CTCP メッセージを扱いたい場合は、 CTCP 型 (例: ACTION) からカスタムフォーマット文字列への マッピングを定義する必要があります。そうするには、 ctcp_messages に連想配列を渡します。 配列のキーが CTCP 型で、対応する値がフォーマットメッセージとなります。

user_settings

IRC サーバに送信する "ident"、"password" および "realname" トークンを 連想配列として指定することが可能です。この連想配列を設定します。

bailout_on_trivial

返り値

後で使うための接続リソースハンドル、あるいはエラー時に **FALSE** を返します。

参考

- ircg_disconnect()
- ircg_is_conn_alive()
- ircg_register_format_messages()

---

ircg_register_format_messages

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_register_format_messages — フォーマットメッセージセットを登録する

説明

bool **ircg_register_format_messages** (string $name , array $messages)

ircg_register_format_messages() を使用することで、 IRC 出力の見た目やクライアント側でどのスクリプトを起動させるかなどを カスタマイズすることが可能です。

パラメータ

name

messages

- Plain channel message
- Private message received
- Private message sent
- Some user leaves channel
- Some user enters channel
- Some user was kicked from the channel
- Topic has been changed
- Error
- Fatal error
- Join list end(?)
- Self part(?)
- Some user changes his nick
- Some user quits his connection
- Mass join begin
- Mass join element
- Mass join end
- Whois user
- Whois server
- Whois idle
- Whois channel
- Whois end
- Voice status change on user
- Operator status change on user
- Banlist
- Banlist end
- %f - from
- %t - to
- %c - channel
- %r - plain message
- %m - encoded message
- %j - js encoded message
- 1 - mod encode
- 2 - nickname decode

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ircg_lookup_format_messages()

---

ircg_set_current

(PHP 4 >= 4.0.4, PHP 5 <= 5.0.5)

ircg_set_current — 出力用に現在の接続を設定する

説明

bool **ircg_set_current** (resource $connection)

この実行コンテキストを出力するための HTTP 接続を選択します。 connection で接続したサーバから送信されたすべての出力は、 デフォルトのフォーマットあるいは ircg_register_format_messages() で指定したフォーマットメッセージセットを使用して標準出力に送られます。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- ircg_register_format_messages()

---

ircg_set_file

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

ircg_set_file — 接続用にログファイルを設定する

説明

bool **ircg_set_file** (resource $connection , string $path)

関数 **ircg_set_file()** は、接続 connection からの全ての出力を記録するログファイルを指定します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

path

ログファイルへのパス。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

---

ircg_set_on_die

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5)

ircg_set_on_die — 接続が終了する際にホスト側で実行されるアクションを設定する

説明

bool **ircg_set_on_die** (resource $connection , string $host , int $port , string $data)

接続 connection が終了する際に、 IRCG はホスト host のポート port に接続し、そのホストに data を送信した後に接続が閉じられるまで待ちます。例えば、 PHP スクリプトを起動させるトリガーとして利用されます。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

host

これは IPv4 アドレスでなければなりません。 IRCG はブロックの問題によりホスト名の解決を行いません。

port

ポート番号。

data

接続が閉じる前に送信されるデータ。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

注意

注意: この機能は IRCG 3 を必要とします。

---

ircg_topic

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_topic — サーバ上のチャンネル用にトピックを設定する

説明

bool **ircg_topic** (*resource $connection* , *string $channel* , *string $new_topic*)

channel のトピックを変更します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

channel

#name 形式のチャンネル名。

new_topic

新しく設定するトピック。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ircg_who

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5)

ircg_who — サーバに *WHO* 情報を問い合わせる

説明

bool **ircg_who** (*resource $connection* , *string $mask* [, *bool $ops_only*])

ircg_who() は、接続しているネットワーク *connection* 上でニックネームが *mask* に一致するユーザの一覧を要求します。

結果は、 *ircg_set_file()* あるいは *ircg_set_current()* で定義した出力に返されます。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

mask

ニックネームのマスク。

ops_only

オプションのパラメータ *ops_only* は、 一覧をサーバオペレータのみに絞り込みます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ircg_set_file()*
- *ircg_set_current()*

---

ircg_whois

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5)

ircg_whois — ユーザ情報をサーバに問い合わせる

説明

bool **ircg_whois** (*resource $connection* , *string $nick*)

指定したユーザの情報を問い合わせるクエリを送信します。

パラメータ

connection

ircg_pconnect() が返す接続リソースハンドル。

nick

ユーザのニックネーム。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

目次

PHP / Java の連携

導入

PHP と Java の連携をとして考えられる手段は 2 種類あります。 PHP を Java サーブレット環境に統合する方法 と Java サポートを PHP に統合する方法です。前者のほうが より安定で効率的な手法です。前者は、サーブレットサーバへのインターフェイス として SAPI モジュールにより提供され、後者は Java 拡張モジュール として提供されます。

Java 拡張モジュールは、PHP から Java オブジェクトのメソッドを生成し、 コールする簡単で効率的な手段を提供します。この JVM は JNI を用いて 作成され、全てはこのプロセスで動作します。

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

要件

この拡張モジュールを使用するには、使用するマシンに Java VM が インストールされていることが必要です。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

PHP 4 の場合、この PECL 拡張モジュールのソースは、PHP のソースの ext/ ディレクトリ、または上の PECL リンクで入手可能です。 これらの関数を使用するには --with-java[=DIR] を使用して Java サポートつきで PHP をコンパイルする必要があります。 DIR は JDK のインストールディレクトリを指します。この拡張モジュールは 共有モジュールとしてのみビルド可能です。詳細な情報は php-src/ext/java/README にあります。

Windows ユーザがこれらの関数を使用するには、php.ini 内で php_java.dll を有効にします。 PHP 4 の場合、この DLL は PHP の Windows ダウンロードバイナリの extensions/ ディレクトリにあります。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

> **注意**: Windows 環境において PHP <= 4.0.6 を使用してこのモジュールを有効に するには、jvm.dll をシステムの PATH が通った場所に おく必要があります。PHP > 4.0.6 では、追加の DLL は必要ありません。

実行時設定

php.ini の設定により動作が変化します。

Java 設定オプション

名前	デフォルト	変更の可否	変更履歴
java.class.path	NULL	PHP_INI_ALL	
java.home	NULL	PHP_INI_ALL	
java.library.path	NULL	PHP_INI_ALL	
java.library	JAVALIB	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

例

Example#1 Java の例

```
<?php
// Java クラス java.lang.System のインスタンスをPHPに作成する
$system = new Java('java.lang.System');

// プロパティへのアクセスのデモ
echo 'Java version=' . $system->getProperty('java.version') . '<br />';
echo 'Java vendor=' . $system->getProperty('java.vendor') . '<br />';
echo 'OS=' . $system->getProperty('os.name') . ' ' .
             $system->getProperty('os.version') . ' on ' .
             $system->getProperty('os.arch') . ' <br />';

// java.util.Dateの例
$formatter = new Java('java.text.SimpleDateFormat',
                      "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

echo $formatter->format(new Java('java.util.Date'));
?>
```

Example#2 AWT の例

```
<?php
// この例は、CGI として実行されることのみを考慮しています。

$frame  = new Java('java.awt.Frame', 'PHP');
$button = new Java('java.awt.Button', 'Hello Java World!');

$frame->add('North', $button);
$frame->validate();
$frame->pack();
$frame->visible = True;

$thread = new Java('java.lang.Thread');
$thread->sleep(10000);

$frame->dispose();
?>
```

注意:

- new Java() は、有効なコンストラクタがある場合、 クラスのインスタンスを生成します。引数が指定されない場合には、 デフォルトのコンストラクタにより java.lang.Systemのようなクラスにアクセスすると 良いでしょう。このクラスは、ほとんどの機能を静的なメソッドとして公開しています。
- あるインスタンスのメンバーにアクセスする際には、まず bean プロパティ が探されてから、次に public フィールドが探されます。言い換えると、 print $date.time はまず $date.getTime() と解釈され、続いて $date.time と解釈されます。
- 静的メンバおよびインスタンスメンバは共に同じ構文でアクセス可能です。 さらに、java オブジェクトが java.lang.Class 型の 場合、このクラスの静的メンバ(フィールドとメソッド)にアクセス可能です。
- 例外が発生すると PHP の警告が出力され、結果は NULL となります。警告は "@" 記号を付けてメソッドをコールすることに抑圧できます。直近のエラーを取得し、リセットするために以下の API を使用することができます。
  - java_last_exception_get()
  - java_last_exception_clear()
- オーバーロードの解決は、二つの言語の間で型の違いがあるため一般には 困難な問題です。PHP の Java 拡張機能は、どのオーバーロードが最も一致するかを定義するための方法として、簡単ですがかなり 効率的なものを使用しています。 加えて、PHP のメソッド名は大文字小文字を区別しないため、選択される オーバーロードの数は増加する傾向があります。 メソッドが一度選択されると、パラメータの値は必要に応じて調整されます。 このため、(倍精度実数が論理値に変換されるといった)データの劣化が 発生する可能性があります。
- PHP では伝統的に配列とハッシュテーブルは相互に完全に可換でした。PHP の ハッシュテーブルは整数または文字列の添字のみを使用できることに注意して ください。また、Java の primitive 型の配列は疎とすることができないことに も注意してください。これらの構造は値で渡されるため、メモリと時間の 消費量が大きくする可能性があります。

Java Servlet SAPI

Java Servlet SAPI は、PHP プロセッサ全体をサーブレットとして実行する ために、Java 拡張モジュールにより定義された機構の上に構築されています。 PHP の側からみてこの実装が基本的に優れている点は、サーブレットを サポートする Web サーバが通常 JVM をプールし、再利用することに注力している ことです。このサーブレット SAPI モジュールの構築手順は、 php4/sapi/README にあります。 注意:

- このコードは、全てのサーブレットエンジンで実行可能であるように作成 されていますが、現在 Apache の Jakarta/tomcat でしかテストされていません。 他のエンジンでこのコードを実行する際に必要なパッチ、バグレポート、 成功事例等をお知らせください。
- PHP は動作ディレクトリを変更する特徴があります。sapi/サーブレット はもとに戻そうとしますが、PHP が実行されている間、サーブレットエンジンは CLASSPATH に相対ディレクトリにより指定されている全てのクラスを ロードできないか、管理用および JSP コンパイル用に使用されている 作業ディレクトリを見つけることができなくなる可能性があります。

---

java_last_exception_clear

(PHP 4 >= 4.0.2)

java_last_exception_clear — 直近の例外をクリアする

説明

void **java_last_exception_clear** (void)

直近の Java の例外をクリアします。

返り値

値を返しません。

例

使用例は java_last_exception_get() を参照ください。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

java_last_exception_get

(PHP 4 >= 4.0.2)

java_last_exception_get — 直近の Java 例外を取得する

説明

object **java_last_exception_get** (void)

直近の Java の例外を取得します。

返り値

例外オブジェクトを返します。

例

以下の例は、Java 例外ハンドラをPHPから使用する方法を示すものです。

Example#1 Java 例外ハンドラ

```
<?php
$stack = new Java('java.util.Stack');
$stack->push(1);

// 以下のコードは実行に成功します
$result = $stack->pop();
$ex = java_last_exception_get();
if (!$ex) {
   echo "$result¥n";
}

// 以下のコードは失敗します (エラー出力は、@により抑制されています)
$result = @$stack->pop();
$ex = java_last_exception_get();
if ($ex) {
   echo $ex->toString();
}

// 直近の例外をクリア
java_last_exception_clear();
?>
```

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

目次

JSON 関数

導入

この拡張モジュールは、» JavaScript Object Notation (JSON) というデータ交換形式を実装したものです。デコード処理は、 Douglas Crockford による JSON_checker を基にしています。

要件

PHP 5.2.0 以降、JSON 拡張モジュールはデフォルトで PHP に組み込まれます。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 »
http://pecl.php.net/package/json

---

json_decode

(PHP 5 >= 5.2.0, PECL json:1.2.0-1.2.1)

json_decode — JSON 文字列をデコードする

説明

mixed **json_decode** (string $json [, bool $assoc])

JSON エンコードされた文字列を受け取り、それを PHP の変数に変換します。

パラメータ

json

デコード対象となる json 文字列。

assoc

TRUE の場合は、返されるオブジェクトが連想配列形式になります。

返り値

オブジェクトを返します。あるいは、オプションのパラメータ assoc が **TRUE** の場合には、 連想配列を返します。

例

Example#1 json_decode() の例

```
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var_dump(json_decode($json));
var_dump(json_decode($json, true));

?>
```

上の例の出力は以下となります。

```
object(stdClass)#1 (5) {
     ["a"] => int(1)
     ["b"] => int(2)
     ["c"] => int(3)
     ["d"] => int(4)
     ["e"] => int(5)
}

array(5) {
     ["a"] => int(1)
     ["b"] => int(2)
     ["c"] => int(3)
     ["d"] => int(4)
     ["e"] => int(5)
}
```

注意

警告

JSON エンコードされたデータのネストの深さが 127 を超えると、 この関数は false を返します。

変更履歴

バージョン	説明
5.2.3	ネストの制限が 20 から 128 に拡張されました。

参考

- json_encode()

---

json_encode

(PHP 5 >= 5.2.0, PECL json:1.2.0-1.2.1)

json_encode — 値を JSON 形式にして返す

説明

string **json_encode** (mixed $value)

value を JSON 形式にした文字列を返します。

パラメータ

value

エンコードする値。 resource 型以外の任意の型を指定できます。

この関数は、UTF-8 エンコードされたデータでのみ動作します。

返り値

成功した場合に、JSON エンコードされた文字列を返します。

変更履歴

バージョン	説明
5.2.1	JSON エンコードの基本型のサポートを追加しました。

例

Example#1 json_encode() の例

```
<?php
$arr = array ('a'=>1,'b'=>2,'c'=>3,'d'=>4,'e'=>5);

echo json_encode($arr);
?>
```

上の例の出力は以下となります。

```
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

参考

- json_decode()

目次

KADM5

導入

このパッケージは、Kerberos V (ケルベロス バージョン 5) 管理サーバへのアクセス機能を提供します。 これにより、Kerberos V のプリンシパルやポリシーの 作成・変更・削除が可能となります。

Kerberos についての詳細な情報は、» http://web.mit.edu/kerberos/www/ で得られます。

Kerberos および KADM5 についてのドキュメントは、» http://web.mit.edu/kerberos/www/krb5-1.2/krb5-1.2.8/doc/admin_toc.html にあります。

リソース型

この拡張モジュールでは KADM5 ハンドルを定義しています。これは kadm5_init_with_password() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

属性フラグ用の定数

関数 kadm5_create_principal()、 kadm5_modify_principal() および kadm5_modify_principal() では、ビットを立てることで 特別な属性を指定することができます。以下のような定数が定義されています。

KDC が使用する属性定数

定数
KRB5_KDB_DISALLOW_POSTDATED
KRB5_KDB_DISALLOW_FORWARDABLE
KRB5_KDB_DISALLOW_TGT_BASED
KRB5_KDB_DISALLOW_RENEWABLE
KRB5_KDB_DISALLOW_PROXIABLE
KRB5_KDB_DISALLOW_DUP_SKEY
KRB5_KDB_DISALLOW_ALL_TIX
KRB5_KDB_REQUIRES_PRE_AUTH
KRB5_KDB_REQUIRES_HW_AUTH
KRB5_KDB_REQUIRES_PWCHANGE
KRB5_KDB_DISALLOW_SVR
KRB5_KDB_PWCHANGE_SERVER
KRB5_KDB_SUPPORT_DESMD5
KRB5_KDB_NEW_PRINC

オプション用の定数

関数 kadm5_create_principal()、 kadm5_modify_principal() および kadm5_get_principal() では、プリンシパルの オプションを連想配列形式で指定したり関数の返り値として 受け取ったりできます。連想配列のキーは、以下で定義されている 文字列定数となります。

プリンシパルを作成/変更/取得する際のオプション

定数	型	説明
KADM5_PRINCIPAL	long	プリンシパルの有効期限を Kerberos タイムスタンプで指定します。
KADM5_PRINC_EXPIRE_TIME	long	プリンシパルの有効期限を Kerberos タイムスタンプで指定します。
KADM5_LAST_PW_CHANGE	long	プリンシパルのパスワードが最後に変更された時刻。
KADM5_PW_EXPIRATION	long	プリンシパルの現在のパスワードの有効期限を Kerberos タイムスタンプで指定します。
KADM5_MAX_LIFE	long	このプリンシパルに発行された Kerberos チケットの最大の生存時間。
KADM5_MAX_RLIFE	long	このプリンシパルに発行された Kerberos チケットの最大の 更新可能生存時間。
KADM5_MOD_NAME	string	最後にこのプリンシパルを変更した Kerberos プリンシパルの名前。
KADM5_MOD_TIME	long	このプリンシパルの最終更新時刻を Kerberos タイムスタンプ形式で指定します。
KADM5_KVNO	long	プリンシパルの現在のキーのバージョン。
KADM5_POLICY	string	このプリンシパルを制御するポリシーの名前。
KADM5_CLEARPOLICY	long	新しいプリンシパルの「デフォルトの」ポリシーには 標準手続きが関連付けられます。 KADM5_CLEARPOLICY は、この挙動を抑制します。
KADM5_LAST_SUCCESS	long	最後に AS_REQ が成功したときの KDC 時刻。
KADM5_LAST_FAILED	long	最後に AS_REQ が失敗したときの KDC 時刻。
KADM5_FAIL_AUTH_COUNT	long	AS_REQ が連続して失敗した数。
KADM5_RANDKEY	long	プリンシパルに対してランダムなパスワードを生成します。 パラメータ *password* は無視されます。
KADM5_ATTRIBUTES	long	KDC が使用する属性のビットフィールド。

例

この例では、KADM5 データベースに対して 接続・問い合わせ・結果 principal の表示・切断 を行います。

Example#1 KADM5 拡張モジュールの概要

```
<?php

   $handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

  print "<h1>get_principals</h1>¥n";
  $principals  = kadm5_get_principals($handle);
  for( $i=0; $i<count($principals); $i++)
       print "$principals[$i]<br>¥n";

  print "<h1>get_policies</h1>¥n";
  $policies  = kadm5_get_policies($handle);
  for( $i=0; $i<count($policies); $i++)
       print "$policies[$i]<br>¥n";

  print "<h1>get_principal  burbach@GONICUS.LOCAL</h1>¥n";

  $options  = kadm5_get_principal($handle, "burbach@GONICUS.LOCAL"  );
  $keys  = array_keys($options);
  for( $i=0; $i<count($keys); $i++) {
    $value  = $options[$keys[$i]];
     print "$keys[$i]:  $value<br>¥n";
  }

  $options  =  array(KADM5_PRINC_EXPIRE_TIME  => 0);
  kadm5_modify_principal($handle, "burbach@GONICUS.LOCAL", $options);

  kadm5_destroy($handle);
?>
```

問合せ先

この拡張モジュールに対するコメントがあったり、バグフィックスや 機能拡張のパッチを作成された場合、あるいは開発に参加したいなどの 場合は » holger.burbach@gonicus.de までメールをください。このプロジェクトのホームページは » http://oss.gonicus.de/project/?group_id=7 です。

---

kadm5_chpass_principal

(PECL kadm5:0.2.3)

kadm5_chpass_principal — プリンシパルのパスワードを変更する

説明

bool **kadm5_chpass_principal** (resource $handle , string $principal , string $password)

kadm5_chpass_principal() は、 principal の新しいパスワードを password に設定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 プリンシパルのパスワードを変更する例

```
<?php

$handle  = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

kadm5_chpass_principal($handle, "burbach@GONICUS.LOCAL", "newpassword");

kadm5_destroy($handle);
?>
```

---

kadm5_create_principal

(PECL kadm5:0.2.3)

kadm5_create_principal — 指定したパラメータで、kerberos のプリンシパルを作成する

説明

bool **kadm5_create_principal** (resource $handle , string $principal [, string $password [, array $options]])

kadm5_create_principal() は、 指定した password で principal を作成します。 password を指定しなかったり **NULL** を指定した場合は、ランダムなキーが生成されます。

配列 options にオプションのパラメータを 指定することができます。指定可能なオプションは以下のとおりです。 KADM5_PRINC_EXPIRE_TIME、KADM5_PW_EXPIRATION、KADM5_ATTRIBUTES、 KADM5_MAX_LIFE、KADM5_KVNO、KADM5_POLICY、KADM5_CLEARPOLICY、 KADM5_MAX_RLIFE

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 プリンシパルを作成する例

```
<?php

$handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

$attributes = KRB5_KDB_REQUIRES_PRE_AUTH | KRB5_KDB_DISALLOW_PROXIABLE;
$options = array(KADM5_PRINC_EXPIRE_TIME => 0,
                 KADM5_POLICY => "default",
                 KADM5_ATTRIBUTES => $attributes);

kadm5_create_principal($handle, "burbach@GONICUS.LOCAL", "password", $options);

kadm5_destroy($handle);
?>
```

---

kadm5_delete_principal

(PECL kadm5:0.2.3)

kadm5_delete_principal — kerberos プリンシパルを削除する

説明

bool **kadm5_delete_principal** (resource $handle , string $principal)

kadm5_delete_principal() は、 Kerberos データベースから principal を削除します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 kadm5_delete_principal() の例

```
<?php

$handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

kadm5_delete_principal($handle, "burbach@GONICUS.LOCAL");

kadm5_destroy($handle);
?>
```

---

kadm5_destroy

(PECL kadm5:0.2.3)

kadm5_destroy — 管理サーバへの接続を閉じ、関連するすべてのリソースを開放する

説明

bool **kadm5_destroy** (resource $handle)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

kadm5_flush

(PECL kadm5:0.2.3)

kadm5_flush — Kerberos データベースに対するすべての変更を取り消し、Kerberos 管理サーバとの接続はオープンしたままにする

説明

bool **kadm5_flush** (resource $handle)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

kadm5_get_policies

(PECL kadm5:0.2.3)

kadm5_get_policies — Kerberos データベースから、すべてのポリシーを取得する

説明

array **kadm5_get_policies** (resource $handle)

kadm5_get_policies() は、ポリシーの名前を含む 配列を返します。

返り値

成功した場合はポリシーの配列、失敗した場合は **FALSE** を返します。

例

Example#1 kadm5_get_policies() の例

```
<?php
$handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

print "<h1>get_policies</h1>¥n";
foreach (kadm5_get_policies($handle) as $policy) {
    echo "$policy<br />¥n";
}

kadm5_destroy($handle);
?>
```

---

kadm5_get_principal

(PECL kadm5:0.2.3)

kadm5_get_principal — Kerberos データベースから、プリンシパルのエントリを取得する

説明

array **kadm5_get_principal** (resource $handle , string $principal)

kadm5_get_principal() は、以下のキーを含む連想配列を返します。 KADM5_PRINCIPAL、KADM5_PRINC_EXPIRE_TIME、KADM5_PW_EXPIRATION、KADM5_ATTRIBUTES、KADM5_MAX_LIFE、KADM5_MOD_NAME、 KADM5_MOD_TIME、KADM5_KVNO、KADM5_POLICY、KADM5_MAX_RLIFE、 KADM5_LAST_SUCCESS、KADM5_LAST_FAILED、KADM5_FAIL_AUTH_COUNT

返り値

成功した場合はオプションの配列、失敗した場合は **FALSE** を返します。

例

Example#1 kadm5_get_principal() の例

```
<?php
$handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

print "<h1>get_principal burbach@GONICUS.LOCAL</h1>¥n";

$options = kadm5_get_principal($handle, "burbach@GONICUS.LOCAL" );

foreach ($options as $key => $value) {
    echo "$key: $value<br />¥n";
}

kadm5_destroy($handle);
?>
```

---

kadm5_get_principals

(PECL kadm5:0.2.3)

kadm5_get_principals — Kerberos データベースから、すべてのプリンシパルを取得する

説明

array **kadm5_get_principals** (resource $handle)

kadm5_get_principals() は、プリンシパルの名前を含む 配列を返します。

返り値

成功した場合はプリンシパルの配列、失敗した場合は **FALSE** を返します。

例

Example#1 kadm5_get_principals() の例

```
<?php
```

```
$handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

print "<h1>get_principals</h1>¥n";
foreach (kadm5_get_principals($handle) as $principal) {
    echo "$principal<br />¥n";
}

kadm5_destroy($handle);
?>
```

---

kadm5_init_with_password

(PECL kadm5:0.2.3)

kadm5_init_with_password — KADM5 ライブラリへの接続をオープンし、必要なステータス情報を初期化する

説明

resource **kadm5_init_with_password** (string $admin_server , string $realm , string $principal , string $password)

kadm5_init_with_password() は、 principal および指定した password を使用して KADM5 ライブラリとの接続をオープンし、admin_server から証明書を取得します。 パラメータ realm は、この接続の認証領域を定義します。

返り値

成功した場合は KADM5 ハンドル、失敗した場合は **FALSE** を返します。

例

Example#1 KADM5 の初期化の例

```
<?php

$handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

$attributes = KRB5_KDB_REQUIRES_PRE_AUTH | KRB5_KDB_DISALLOW_PROXIABLE;
$options = array(KADM5_PRINC_EXPIRE_TIME => 0,
                 KADM5_POLICY => "default",
                 KADM5_ATTRIBUTES => $attributes);

kadm5_create_principal($handle, "burbach@GONICUS.LOCAL", "password", $options);

kadm5_destroy($handle);
?>
```

注意

注意: kadm5_destroy() を使用すると、接続は閉じられます。

参考

- kadm5_destroy()

---

kadm5_modify_principal

(PECL kadm5:0.2.3)

kadm5_modify_principal — kerberos プリンシパルを、指定したパラメータで変更する

説明

bool **kadm5_modify_principal** (resource $handle , string $principal , array $options)

kadm5_modify_principal() は、 指定した options に基づいて principal を変更します。変更可能なオプションは 以下のとおりです。KADM5_PRINC_EXPIRE_TIME、KADM5_PW_EXPIRATION、KADM5_ATTRIBUTES、 KADM5_MAX_LIFE、KADM5_KVNO、KADM5_POLICY、KADM5_CLEARPOLICY、 KADM5_MAX_RLIFE、KADM5_FAIL_AUTH_COUNT

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 プリンシパルの変更の例

```
<?php

$handle = kadm5_init_with_password("afs-1", "GONICUS.LOCAL", "admin/admin", "password");

$attributes = KRB5_KDB_REQUIRES_PRE_AUTH;
$options = array(KADM5_PRINC_EXPIRE_TIME => 3451234,
                 KADM5_POLICY => "gonicus",
                 KADM5_ATTRIBUTES => $attributes);

kadm5_modify_principal($handle, "burbach@GONICUS.LOCAL", $options);

kadm5_destroy($handle);
```

```
?>
```

目次

LDAP 関数

導入

LDAP とは Lightweight Directory Access Protocol を意味し、 "ディレクトリサーバ" にアクセスするために使用されるプロトコルです。 ディレクトリとは、 ツリー構造に情報を保持している特殊なデータベースのことです。

この概念は、ハードディスクのディレクトリ構造に似ています。 ただし、その内容は異なっており、ルートディレクトリは "世界" であり、 最初のレベルのサブディレクトリは "国" となります。ディレクトリ構造の 下位には会社や機関、場所のエントリがあります。さらに下位には、 人やおそらく道具や文書に関するディレクトリエントリもあります。

ハードディスク内のサブディレクトリにあるファイルを参照するには、 次のようにすることでしょう。

```
/usr/local/myapp/docs
```

スラッシュが参照の各部分を区分し、左から右に解釈されます。

LDAP においてこの完全に正しいファイル参照に等価なものは "区分された名前(distinguished name)" であり、単に "dn" と表されます。 例として dn が次のようになっているとしましょう。

```
cn=John Smith,ou=Accounts,o=My Company,c=US
```

カンマは各部分を区分し、右から左に解釈されます。この dn は 次のように解釈されます。

```
country = US
organization = My Company
organizationalUnit = Accounts
commonName = John Smith
```

ハードディスクのディレクトリ構造を管理する手法について明確な規約が ないのと同様に、ディレクトリサーバーマネージャーは、目的に適した あらゆる構造を設定することが可能です。しかし、実用的にはいくつかの 慣習があります。 利用可能なものに関する情報を持たずにデータベースを使用することが できないのと同様、ディレクトリの構造に関する情報なくして ディレクトリサーバーにアクセスするコードを書くことはできない ということが言えます。

LDAP に関する多くの情報が以下の場所にあります。

- » Mozilla
- » OpenLDAP Project

Netscape SDK では、有用な » プログラマーズガイド が HTML 形式で公開されています。

要件

LDAP サポートを有効にして PHP をコンパイルするには、 » OpenLDAP あるいは » Bind9.net から LDAP クライアント ライブラリを入手し、コンパイルしておく必要があります。

インストール手順

PHP の LDAP サポートはデフォルトで有効になっていません。 LDAP サポートを有効にして PHP をコンパイルするには、設定オプション --with-ldap[=DIR] を指定して PHP をコンパイルする必要があります。 DIR は LDAP をインストールしたディレクトリです。 SASL サポートを有効にするためには、システム上に sasl.h を用意し、 --with-ldap-sasl[=DIR] を指定する必要があります。

注意: **Win32 ユーザへの注意** この拡張モジュールを動作させるには、 Windows システムの PATH が通った場所に DLL ファイルが存在する必要があります。 FAQ の "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" で、その方法を説明しています。 DLL ファイルを PHP のフォルダから Windows のシステムディレクトリにコピーしても動作します (システムディレクトリは、デフォルトで PATH に含まれるからです) が、これは推奨しません。 この拡張モジュールを使用するには、以下のファイルが PATH の通った場所にある必要があります。 libeay32.dll および ssleay32.dll
PHP 4.3.0 より前のバージョンでは、さらに libsasl.dll も必要です。

Oracle LDAP ライブラリを使用するには、Oracle 環境 が適切に設定されている 必要があります。

実行時設定

php.ini の設定により動作が変化します。

LDAP 設定オプション

名前	デフォルト	変更の可否	変更履歴
ldap.max_links	"-1"	PHP_INI_SYSTEM	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

ほとんどの LDAP 関数は、リソースを操作するか、あるいはリソースを返します（たとえば ldap_connect() は正の LDAP リンク ID を返し、ほとんどの LDAP 関数はそれを使用します）。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

LDAP_DEREF_NEVER (integer)
LDAP_DEREF_SEARCHING (integer)
LDAP_DEREF_FINDING (integer)
LDAP_DEREF_ALWAYS (integer)
LDAP_OPT_DEREF (integer)
LDAP_OPT_SIZELIMIT (integer)
LDAP_OPT_TIMELIMIT (integer)
LDAP_OPT_NETWORK_TIMEOUT (integer)
ldap_set_option() 用のオプションで、ネットワークのタイムアウトを設定します（PHP 5.3.0 以降で利用可能です）。
LDAP_OPT_PROTOCOL_VERSION (integer)
LDAP_OPT_ERROR_NUMBER (integer)
LDAP_OPT_REFERRALS (integer)
LDAP_OPT_RESTART (integer)
LDAP_OPT_HOST_NAME (integer)
LDAP_OPT_ERROR_STRING (integer)
LDAP_OPT_MATCHED_DN (integer)
LDAP_OPT_SERVER_CONTROLS (integer)
LDAP_OPT_CLIENT_CONTROLS (integer)
LDAP_OPT_DEBUG_LEVEL (integer)
GSLC_SSL_NO_AUTH (integer)
GSLC_SSL_ONEWAY_AUTH (integer)
GSLC_SSL_TWOWAY_AUTH (integer)

例

あるディレクトリサーバーから姓が "S" から始まる全てのエントリに 関する情報を検索し、名前と電子メールアドレスで検索結果を表示します。

Example#1 LDAP 検索の例

```
<?php
// LDAP の基本シーケンスは、接続、バインド、検索、検索結果の解釈、
// 接続のクローズです。

echo "<h3>LDAP query test</h3>";
echo "Connecting ...";
$ds=ldap_connect("localhost");  // 有効な LDAP サーバーに違いない!
echo "connect result is " . $ds . "<br />";

if ($ds) {
    echo "Binding ...";
    $r=ldap_bind($ds);     // これは "匿名" バインドで、通常は
                           // 読みこみのみのアクセスとなります。
    echo "Bind result is " . $r . "<br />";

    echo "Searching for (sn=S*) ...";
    // 名前(surname)エントリを検索
    $sr=ldap_search($ds, "o=My Company, c=US", "sn=S*");
    echo "Search result is " . $sr . "<br />";

    echo "Number of entires returned is " . ldap_count_entries($ds, $sr) . "<br />";

    echo "Getting entries ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Data for " . $info["count"] . " items returned:<p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn is: " . $info[$i]["dn"] . "<br />";
        echo "first cn entry is: " . $info[$i]["cn"][0] . "<br />";
        echo "first email entry is: " . $info[$i]["mail"][0] . "<br /><hr />";
    }

    echo "Closing connection";
    ldap_close($ds);

} else {
    echo "<h4>Unable to connect to LDAP server</h4>";
}
?>
```

PHP LDAP コールの使用法

LDAP コールを使用する前に、以下のことを知っておいてください。

- *使用するディレクトリサーバの名前またはアドレス*
- *サーバの "base dn"（このサーバーがあるワールドディレクトリ の部分で、"o=My Company,c=US" のようにすることができます）*
- *サーバーへのアクセスにパスワードを必要とするかどうか（多くのサーバーは "匿名バインド" に関して読みこみを 許可するが、他の処理につ*

いてはパスワードを要求します)。

アプリケーションとして作成する LDAP コールのシーケンスは、 通常、次のようなパターンに沿っています。

```
ldap_connect()      // サーバへの接続を確立
    |
ldap_bind()         // 匿名または認証された "ログイン"
    |
ディレクトリの検索または更新等を行い、結果を表示する
    |
ldap_close()        // "ログアウト"
```

---

ldap_8859_to_t61

(PHP 4 >= 4.0.2, PHP 5)

ldap_8859_to_t61 — 8859 文字を t61 文字に変換する

説明

string **ldap_8859_to_t61** (string $value)

ISO-8859 文字を t61 文字に変換します。

この関数は、 LDAPv2 サーバとやり取りしなければならない場合に有用です。

パラメータ

value

変換するテキスト。

返り値

value を t61 に変換したものを返します。

参考

- ldap_t61_to_8859()

---

ldap_add

(PHP 4, PHP 5)

ldap_add — LDAP ディレクトリにエントリを付加する

説明

bool **ldap_add** (resource $link_identifier , string $dn , array $entry)

エントリを LDAP ディレクトリに追加します。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

entry

そのエントリに関する情報を表す配列。 エントリの値は、個々の属性によりインデックスが作成されています。 ある属性に関して複数の値がある場合は、0 から始まる整数で 添字が作成されます。

```
<?php
$entree["attribut1"] = "value";
$entree["attribut2"][0] = "value1";
$entree["attribut2"][1] = "value2";
?>
```

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 認証型バインドの例

```
<?php
$ds = ldap_connect("localhost");  // LDAP サーバーはこのホストであると仮定

if ($ds) {
    // 更新アクセスを行うために適当な dn でバインドする
    $r = ldap_bind($ds, "cn=root, o=My Company, c=US", "secret");
```

```
    // データを準備する
    $info["cn"] = "John Jones";
    $info["sn"] = "Jones";
    $info["mail"] = "jonj@example.com";
    $info["objectclass"] = "person";

    // データをディレクトリに追加
    $r = ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);

    ldap_close($ds);
} else {
    echo "LDAP サーバーに接続できません";
}
?>
```

注意

注意: この関数はバイナリデータに対応しています。

参考

- ldap_delete()

---

ldap_bind

(PHP 4, PHP 5)

ldap_bind — LDAP ディレクトリにバインドする

説明

bool **ldap_bind** (resource $link_identifier [, string $bind_rdn [, string $bind_password]])

指定した RDN およびパスワードを用いて LDAP ディレクトリにバインドします。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

bind_rdn

bind_password

bind_rdn および bind_password を省略した場合は匿名バインドを試みます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 LDAP バインドの使用

```
<?php

// ldap バインドを使用する
$ldaprdn  = 'uname';     // ldap rdn あるいは dn
$ldappass = 'password';  // パスワード

// ldap サーバに接続する
$ldapconn = ldap_connect("ldap.example.com")
    or die("Could not connect to LDAP server.");

if ($ldapconn) {

    // ldap サーバにバインドする
    $ldapbind = ldap_bind($ldapconn, $ldaprdn, $ldappass);

    // バインド結果を検証する
    if ($ldapbind) {
        echo "LDAP bind successful...";
    } else {
        echo "LDAP bind failed...";
    }

}

?>
```

Example#2 LDAP 匿名バインドの使用

```
<?php

// ldap 匿名バインドを使用する

// ldap サーバに接続する
$ldapconn = ldap_connect("ldap.example.com")
    or die("Could not connect to LDAP server.");
```

```
if ($ldapconn) {

    // 匿名でバインドする
    $ldapbind = ldap_bind($ldapconn);

    if ($ldapbind) {
        echo "LDAP bind anonymous successful...";
    } else {
        echo "LDAP bind anonymous failed...";
    }

}

?>
```

参考

- ldap_unbind()

---

ldap_close

(PHP 4, PHP 5)

ldap_close — のエイリアス ldap_unbind()

説明

この関数は次の関数のエイリアスです。 ldap_unbind().

---

ldap_compare

(PHP 4 >= 4.0.2, PHP 5)

ldap_compare — 指定した DN のエントリで見付かった属性の値を比較する

説明

mixed **ldap_compare** (resource $link_identifier , string $dn , string $attribute , string $value)

属性 attribute の値 value を、指定した LDAP ディレクトリエントリの同じ属性の値と比較します。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

attribute

属性の名前。

value

比較する値。

返り値

value がマッチする場合に **TRUE**、その他の場合に **FALSE**、エラーの場合に -1 を返します。

例

次の例は、指定したパスワードが DN の指定したエントリで定義された ものと一致するかどうかをチェックする方法を示しています。

Example#1 パスワード確認の例

```
<?php

$ds=ldap_connect("localhost");  // LDAP サーバが同一ホストであると仮定

if ($ds) {

    // バインド
    if (ldap_bind($ds)) {

        // データを準備
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // 値を比較
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: " . ldap_error($ds);
```

```
        } elseif ($r === true) {
            echo "Password correct.";
        } elseif ($r === false) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

注意

警告

ldap_compare() では BINARY 値を比較することはできません!

---

ldap_connect

(PHP 4, PHP 5)

ldap_connect — LDAP サーバへ接続する

説明

resource **ldap_connect** ([string $hostname [, int $port]])

指定した hostname および port 上の LDAP サーバとの接続を確立します。

パラメータ

hostname

OpenLDAP 2.x.x を使用している場合、ホスト名の代わりに URL を指定する ことが可能です。SSL と組み合わせて LDAP を使用するには、SSL サポートを指定して OpenLDAP 2.x.x をコンパイルし、PHP の configure で SSL を指定し、このパラメータを ldaps://hostname/ とします。

port

接続するポート。URL を用いる場合は使用しません。デフォルトは 389 です。

返り値

成功した場合に正の LDAP リンク ID 、エラーの場合に **FALSE** を返します。OpenLDAP 2.x.x を使用している場合は、 **ldap_connect()** は常に resource を返しますが、実際には接続せずにパラメータの初期化だけを行います。 実際に接続するのは次に ldap_* 関数がコールされた際で、これは 通常は ldap_bind() です。

引数が指定されない場合、既に開かれているリンクのリンク ID を返します。

変更履歴

バージョン	説明
4.0.4	URL および SSL のサポートが追加されました。

例

Example#1 LDAP サーバに接続する例

```
<?php

// LDAP 変数
$ldaphost = "ldap.example.com";  // ldap サーバ
$ldapport = 389;                 // ldap サーバのポート番号

// LDAP に接続します
$ldapconn = ldap_connect($ldaphost, $ldapport)
          or die("Could not connect to $ldaphost");

?>
```

Example#2 LDAP サーバへのセキュアな接続の例

```
<?php

// サーバ証明書が証明するホストであることを
// 確認する
$ldaphost = "ldaps://ldap.example.com/";

// LDAP に接続します
$ldapconn = ldap_connect($ldaphost)
          or die("Could not connect to {$ldaphost}");

?>
```

参考

- ldap_bind()

---

ldap_count_entries

(PHP 4, PHP 5)

ldap_count_entries — 検索結果のエントリ数を数える

説明

int **ldap_count_entries** (resource $link_identifier , resource $result_identifier)

直前の検索結果として保存されたエントリの数を返します。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_identifier

内部の LDAP 結果。

返り値

結果のエントリ数を返します。エラーの場合は **FALSE** を返します。

---

ldap_delete

(PHP 4, PHP 5)

ldap_delete — ディレクトリからエントリを削除する

説明

bool **ldap_delete** (resource $link_identifier , string $dn)

指定したエントリを LDAP ディレクトリから削除します。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ldap_add()

---

ldap_dn2ufn

(PHP 4, PHP 5)

ldap_dn2ufn — DN をユーザに分かりやすい名前のフォーマットに変換する

説明

string **ldap_dn2ufn** (string $dn)

指定した dn をよりユーザにわかりやすい形式にするため、 型名を取り除きます。

パラメータ

dn

LDAP エンティティの識別名。

返り値

ユーザにわかりやすい形式の名前を返します。

---

ldap_err2str

(PHP 4, PHP 5)

ldap_err2str — LDAP のエラー番号をエラーメッセージ文字列に変換する

説明

string **ldap_err2str** (int $errno)

この関数は、エラー番号 errno が意味する エラーメッセージ文字列を返します。LDAP errno 番号は標準化されていますが、 異なったライブラリでは、異なる（あるいはローカライズされた）エラーメッセージ が返されます。エラーメッセージの内容をチェックするのではなく、 必ずエラー番号をチェックするようにしてください。

パラメータ

errno

エラー番号。

返り値

エラーメッセージを表す文字列を返します。

例

Example#1 全ての LDAP エラーメッセージに番号をふる

```
<?php
  for ($i=0; $i<100; $i++) {
    printf("Error $i: %s<br />¥n", ldap_err2str($i));
  }
?>
```

参考

- ldap_errno()
- ldap_error()

---

ldap_errno

(PHP 4, PHP 5)

ldap_errno — 直近の LDAP コマンドの LDAP エラー番号を返す

説明

int **ldap_errno** (resource $link_identifier)

直近の LDAP コマンドにより返された、標準化されたエラー番号を返します。 この番号は、ldap_err2str() を用いてエラーメッセージ 文字列に変換することができます。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

返り値

このリンクに関する直近の LDAP コマンドの LDAP エラー番号を返します。

例

php.ini で警告レベルを十分に下げるか、警告出力を抑制するために LDAP コマンドに @ 文字をつけない限り、発生したエラーは、HTML 出力 にも表示されます。

Example#1 エラーを生成し、取得する

```
<?php
// この例には、エラーがあり、これを取得します。
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
// フィルタ式に構文エラーがあります (errno 87)。
// 動作させるには、"objectclass=*" とする必要があります。
$res = @ldap_search($ld, "o=Myorg, c=DE", "objectclass");
if (!$res) {
    echo "LDAP-Errno: " . ldap_errno($ld) . "<br />¥n";
    echo "LDAP-Error: " . ldap_error($ld) . "<br />¥n";
    die("Argh!<br />¥n");
}
$info = ldap_get_entries($ld, $res);
echo $info["count"] . " matching entries.<br />¥n";
?>
```

参考

- ldap_err2str()
- ldap_error()

---

ldap_error

(PHP 4, PHP 5)

ldap_error — 直近の LDAP コマンドの LDAP エラーメッセージを返す

説明

string **ldap_error** (resource $link_identifier)

指定した link_identifier に関して、 直近の LDAP コマンドにより生成されたエラーを表すエラーメッセージ文字列を返します。 LDAP errno 番号は標準化されていますが、ライブラリによって異なる (あるいはローカライズされた) エラーメッセージを返します。 エラーメッセージの内容をチェックするのではなく、 必ずエラー番号をチェックするようにしてください。

php.ini で警告レベルを十分に下げるか、 警告出力を抑制するために LDAP コマンドの前に @ 文字をつけない限り、発生したエラーは HTML 出力にも表示されます。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

返り値

エラーメッセージ文字列を返します。

参考

- ldap_err2str()
- ldap_errno()

---

ldap_explode_dn

(PHP 4, PHP 5)

ldap_explode_dn — DN を構成要素ごとに分割する

説明

array **ldap_explode_dn** (string $dn , int $with_attrib)

ldap_get_dn() により返された DN を分割し、 複数の要素に分けます。各部分は、相対区分名 (Relative Distinguished Name または RDN) と呼ばれます。

パラメータ

dn

LDAP エンティティの識別名。

with_attrib

RDN が値のみを返すのか、 あるいは、属性を同時に返すのかを指定するために使用します。 属性を有する RDN (属性=値 フォーマットで) を得るためには with_attrib を 0 とし、 値のみを得るためには 1 に設定します。

返り値

すべての DN 要素を配列で返します。

---

ldap_first_attribute

(PHP 4, PHP 5)

ldap_first_attribute — 最初の属性を返す

説明

string **ldap_first_attribute** (resource $link_identifier , resource $result_entry_identifier)

指定したエントリの最初の属性を取得します。残りの属性は、 ldap_next_attribute() を逐次コールして取得します。

エントリの読み込みと同様、 エントリからの属性の読み込みもひとつづつ行われます。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_entry_identifier

ber_identifier

ber_identifier は内部メモリにおける位置ポインタの ID です。この ID は 参照渡しされます。同じ ber_identifier が ldap_next_attribute() 関数に渡された場合、 そのポインタは修正されます。

> ***注意**: このパラメータは使用されなくなりました。現在は PHP が自動的にこれを処理します。 後方互換性を保持するため、もしこのパラメータが渡されても PHP はエラーを発生させません。*

返り値

*成功した場合にエントリの最初の属性、エラーの場合に **FALSE** を返します。*

変更履歴

バージョン	説明
5.2.4	*ber_identifier* が削除されました。 これは PHP が自動的に処理するようになりました。

参考

- *ldap_next_attribute()*
- *ldap_get_attributes()*

---

ldap_first_entry

(PHP 4, PHP 5)

ldap_first_entry — 最初の結果 ID を返す

説明

*resource **ldap_first_entry** (resource $link_identifier , resource $result_identifier)*

結果内の最初のエントリのエントリ ID を返します。このエントリ ID を ldap_next_entry() に渡し、 結果からそれ以降のエントリを取得します。

*LDAP 結果におけるエントリは、**ldap_first_entry()** および ldap_next_entry() 関数を用いて連続的に 読み込まれます。*

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_identifier

返り値

*成功時に最初のエントリに関する結果エントリ ID、エラーの場合に **FALSE** を返します。*

参考

- *ldap_get_entries()*

---

ldap_first_reference

(PHP 4 >= 4.0.5, PHP 5)

ldap_first_reference — 最初のリファレンスを返す

説明

*resource **ldap_first_reference** (resource $link , resource $result)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_free_result

(PHP 4, PHP 5)

ldap_free_result — 結果メモリを開放する

説明

bool **ldap_free_result** (resource $result_identifier)

内部での結果保持用に割り当てられていたメモリを開放します。 結果メモリは、スクリプトの終了時に自動的に開放されます。

通常、LDAP の結果用に確保された全てのメモリはスクリプトの実行終了時 に開放されます。連続的な検索を行うスクリプトのように大きな 結果セットを返す場合、スクリプトにより使用される実行用メモリを 小さく保つために **ldap_free_result()** をコールすることが可能です。

パラメータ

result_identifier

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ldap_get_attributes

(PHP 4, PHP 5)

ldap_get_attributes — 検索結果エントリから属性を得る

説明

array **ldap_get_attributes** (resource $link_identifier , resource $result_entry_identifier)

検索結果エントリから属性と値を読み込みます。

ディレクトリに特定のエントリを置いている場合、この関数をコールする ことにより、そのエントリに関して保持されている情報得ることができます。 ディレクトリエントリを "ブラウズ" するか、ディレクトリエントリの 構造が未知であるアプリケーションにおいてこの関数を使用します。 多くのアプリケーションにおいては、電子メールアドレスや姓のような 特定の属性を検索するため、保持されている他のデータにどんなものがあるか ということは問題にならないと思われます。

```
return_value["count"] = そのエントリの属性の数
return_value[0] = 最初の属性
return_value[n] = n 番目の属性

return_value["attribute"]["count"] = その属性に関する値の数
return_value["attribute"][0] = その属性に関する最初の値
return_value["attribute"][i] = その属性に関する (i+1) 番目の値
```

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_entry_identifier

返り値

成功時に完全なエントリ情報を多次元配列で返します。エラーの場合、 **FALSE** を返します。

例

Example#1 特定のディレクトリエントリに関して保持されている属性のリストを表示

```
<?php
// $ds はディレクトリのリンク ID

// $sr は事前の LDAP のディレクトリ検索コールの有効な結果

$entry = ldap_first_entry($ds, $sr);

$attrs = ldap_get_attributes($ds, $entry);

echo $attrs["count"] . " attributes held for this entry:<p>";

for ($i=0; $i < $attrs["count"]; $i++) {
    echo $attrs[$i] . "<br />";
}
?>
```

参考

- ldap_first_attribute()
- ldap_next_attribute()

---

ldap_get_dn

(PHP 4, PHP 5)

ldap_get_dn — 結果エントリから DN を得る

説明

string **ldap_get_dn** (resource $link_identifier , resource $result_entry_identifier)

結果における、あるエントリの DN を見つけます。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_entry_identifier

返り値

結果エントリの DN を返します。エラーの場合、**FALSE**を返します。

---

ldap_get_entries

(PHP 4, PHP 5)

ldap_get_entries — 全ての結果エントリを得る

説明

array **ldap_get_entries** (resource $link_identifier , resource $result_identifier)

指定した結果から複数のエントリを読み込み、 その属性および複数の値を読み込みます。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_identifier

返り値

成功時に完全な結果情報を多次元配列で返します。エラーの場合に **FALSE** を返します。

配列の構造は次のようになります。 属性インデックスは、小文字に変換されます (属性は大文字小文字を区別しませんが、配列インデックスとして使用する時は別です)。

return_value["count"] = 結果におけるエントリの数
return_value[0] : 最初のエントリの詳細情報を参照します

return_value[i]["dn"] = 結果における i 番目のエントリ DN

return_value[i]["count"] = i 番目のエントリにおける属性の数
return_value[i][j] = 結果における i 番目のエントリにおける j 番目の属性

return_value[i]["attribute"]["count"] = i 番目のエントリにおける属性に関する値の数
return_value[i]["attribute"][j] = i 番目のエントリにおける j 番目の値

参考

- ldap_first_entry()
- ldap_next_entry()

---

ldap_get_option

(PHP 4 >= 4.0.4, PHP 5)

ldap_get_option — 指定したオプションの現在の値を得る

説明

bool **ldap_get_option** (resource $link_identifier , int $option , mixed &$retval)

retval を、指定したオプションの値として設定します。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

option

パラメータ option は以下のいずれかとなります。

オプション	型
LDAP_OPT_DEREF	integer
LDAP_OPT_SIZELIMIT	integer
LDAP_OPT_TIMELIMIT	integer
LDAP_OPT_NETWORK_TIMEOUT	integer
LDAP_OPT_PROTOCOL_VERSION	integer

オプション	型
LDAP_OPT_ERROR_NUMBER	integer
LDAP_OPT_REFERRALS	bool
LDAP_OPT_RESTART	bool
LDAP_OPT_HOST_NAME	string
LDAP_OPT_ERROR_STRING	string
LDAP_OPT_MATCHED_DN	string
LDAP_OPT_SERVER_CONTROLS	array
LDAP_OPT_CLIENT_CONTROLS	array

retval

これが、オプションの値として設定されます。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 プロトコルのバージョンを調べる

```
<?php
// $ds はディレクトリサーバへの有効なリンクIDです
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version)) {
    echo "使用中のプロトコルのバージョン: $version\n";
} else {
    echo "プロトコルのバージョンを決定することができません\n";
}
?>
```

注意

> **注意**: この関数は、OpenLDAP 2.x.x または Netscape Directory SDK x.x を使用した場合にのみ利用可能です。

参考

- ldap_set_option()

---

ldap_get_values_len

(PHP 4, PHP 5)

ldap_get_values_len — 結果エントリから全てのバイナリ値を得る

説明

*array **ldap_get_values_len** (resource $link_identifier , resource $result_entry_identifier , string $attribute)*

結果のエントリから、属性のすべての値を読み込みます。

この関数は、文字列データではなくバイナリデータを処理すること以外は、ldap_get_values() と全く同じです。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_entry_identifier

attribute

返り値

*成功時に属性の値を配列で返し、エラー時に **FALSE** を返します。 個々の値は、配列インデックスによりアクセスします。最初のインデックスは、0 です。 値の数は、結果の配列の "count" という要素で知ることができます。*

参考

- ldap_get_values()

---

ldap_get_values

(PHP 4, PHP 5)

ldap_get_values — 結果エントリから全ての値を得る

説明

*array **ldap_get_values** (resource $link_identifier , resource $result_entry_identifier , string $attribute)*

結果内のエントリ属性の、すべての値を読み込みます。

この関数のコールは、result_entry_identifier を必要とします。このため、事前に LDAP の検索用関数のコールと個々の エントリ取得用関数のコールを行っておく必要があります。

アプリケーションでは、("surname" または "mail" のような)特定の属性 を探すためにその属性をコードに埋め込んで置くか、さもなくば、指定した エントリに関して存在する属性を調べるために ldap_get_attributes() をコールする必要があります。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_entry_identifier

attribute

返り値

成功した場合、指定した属性に関する値を配列を返します。エラーの場合は **FALSE** を返します。値の数を調べるには、結果の配列の "count" という要素を確認します。個々の値にアクセスするには、 整数値のインデックスを指定して配列にアクセスします。 最初のインデックスは 0 となります。

LDAP では、ある属性に関して複数のエントリを持つことが可能です。 このため、たとえば一人の人間のディレクトリエントリに多数の 電子メールアドレスがあったとしても、それらをすべて "mail" という属性で管理することができます。

```
return_value["count"] = 属性の値の数
return_value[0] = 属性の最初の値
return_value[i] = 属性の i 番目の値
```

例

Example#1 あるディレクトリエントリの全ての "mail" 属性の一覧を表示する

```
<?php
// $ds はディレクトリサーバーの有効なリンク ID

// $sr は事前の LDAP 検索コールのどれかから返された有効な検索結果

// $entry はディレクトリエントリを返すコールのどれかから返された有効な
//         エントリ ID

$values = ldap_get_values($ds, $entry, "mail");

echo $values["count"] . " email addresses for this entry.<br />";

for ($i=0; $i < $values["count"]; $i++) {
    echo $values[$i] . "<br />";
}
?>
```

参考

- ldap_get_values_len()

---

ldap_list

(PHP 4, PHP 5)

ldap_list — 単一階層の検索を行う

説明

resource **ldap_list** (resource $link_identifier , string $base_dn , string $filter [, array $attributes [, int $attrsonly [, int $sizelimit [, int $timelimit [, int $deref]]]]])

指定したフィルタ filter を使用し、スコープ **LDAP_SCOPE_ONELEVEL** でディレクトリを検索します。

LDAP_SCOPE_ONELEVEL は、コール時に指定した base_dn の直下の階層から検索した結果のみを返すことを表します ("ls" と入力して、現在の作業ディレクトリのファイル/フォルダ 一覧を取得するのと同じようなものです)。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

base_dn

ディレクトリのベース DN。

filter

attributes

必要な属性を、 array("mail", "sn", "cn") のような通常の PHP 文字列配列で保持します。 "dn" は要求された属性の型によらず常に返されることに注意してください。

このパラメータを使用すると、デフォルトの動作よりもかなり効率的になります (デフォルトでは、すべての属性とその値を返します)。 したがって、これを使用することを推奨します。

attrsonly

属性の型のみを取得したい場合は 1 を設定します。 属性の型および値の両方を取得したい場合は 0 を設定します (これがデフォルトの挙動です)。

sizelimit

取得するエントリ数の制限を設定します。 0 は無制限であることを表します。

> ***注意**: このパラメータは、サーバ側で事前に設定されている sizelimit を上書きすることはできません。それ以下の値を指定することはできます。*
> *ディレクトリサーバのホストによっては、 事前に設定された数以上のエントリを返さないようになっているものもあります。 この場合、サーバでは、それが結果セットのすべてではないことを通知します。 このパラメータでエントリ数を制限した場合にも、同じことが起こります。*

timelimit

検索に要する最大秒数を設定します。 これを 0 にすると無制限であることを表します。

> ***注意**: このパラメータは、サーバ側で事前に設定されている timelimit を上書きすることはできません。それ以下の値を指定することはできます。*

deref

検索時のエイリアスの扱いについて指定します。 以下のいずれかとなります。

- ***LDAP_DEREF_NEVER** - (デフォルト) エイリアスは参照されません。*
- ***LDAP_DEREF_SEARCHING** - エイリアスを参照しますが、検索のベースオブジェクト上にいるときは参照しません。*
- ***LDAP_DEREF_FINDING** - エイリアスの参照は、ベースオブジェクト上にいて検索中でない場合に行われます。*
- ***LDAP_DEREF_ALWAYS** - エイリアスを常に参照します。*

返り値

*検索結果 ID を返します。エラーの場合は、**FALSE** を返します。*

変更履歴

バージョン	説明
4.0.5	並列検索のサポートが追加されました。詳細は ldap_search() を参照ください。
4.0.2	*attrsonly* 、*sizelimit* 、 *timelimit* および *deref* が追加されました。

例

Example#1 ある組織の全ての組織単位を一覧表示する

```
// $ds はディレクトリサーバーの有効なリンク ID

$basedn = "o=My Company, c=US";
$justthese = array("ou");

$sr=ldap_list($ds, $basedn, "ou=*", $justthese);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++) {
     echo $info[$i]["ou"][0] ;
}
```

参考

- *ldap_search()*

---

ldap_mod_add

(PHP 4, PHP 5)

ldap_mod_add — 現在の属性に属性を追加する

説明

*bool **ldap_mod_add** (resource $link_identifier , string $dn , array $entry)*

指定した dn に属性を追加します。 この関数は、オブジェクトレベルではなく属性レベルで修正を行います。 オブジェクトレベルの追加は、ldap_add() 関数により行います。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

entry

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数はバイナリデータに対応しています。

参考

- ldap_mod_del()
- ldap_mod_replace()

---

ldap_mod_del

(PHP 4, PHP 5)

ldap_mod_del — 現在の属性から属性を削除する

説明

bool **ldap_mod_del** (resource $link_identifier , string $dn , array $entry)

ひとつあるいは複数の属性を、指定した dn から削除します。 この関数は、オブジェクトレベルではなく属性レベルで修正を行います。 オブジェクトレベルの削除は、ldap_delete() 関数により行います。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

entry

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ldap_mod_add()
- ldap_mod_replace()

---

ldap_mod_replace

(PHP 4, PHP 5)

ldap_mod_replace — 属性を新規の値に置換する

説明

bool **ldap_mod_replace** (resource $link_identifier , string $dn , array $entry)

指定した dn でひとつあるいは複数の属性の置換を行います。 この関数は、オブジェクトレベルではなく属性レベルで修正を行います。 オブジェクトレベルの修正は、ldap_modify() 関数により行います。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

entry

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この関数はバイナリデータに対応しています。

参考

- ldap_mod_del()

- ldap_mod_add()

---

ldap_modify

(PHP 4, PHP 5)

ldap_modify — LDAP エントリを修正する

説明

bool **ldap_modify** (resource $link_identifier , string $dn , array $entry)

LDAP ディレクトリ内のエントリを修正します。エントリの構造は ldap_add() と同じです。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

entry

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数はバイナリデータに対応しています。

参考

- ldap_rename()

---

ldap_next_attribute

(PHP 4, PHP 5)

ldap_next_attribute — 結果における次の属性を得る

説明

string **ldap_next_attribute** (resource $link_identifier , resource $result_entry_identifier)

エントリ内の属性を取得します。 最初に **ldap_next_attribute()** をコールした際には ldap_first_attribute() から返される result_entry_identifier を使用します。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_entry_identifier

ber_identifier

ポインタの内部状態を、このパラメータで管理します。

> **注意**: このパラメータは使用されなくなりました。現在は PHP が自動的にこれを処理します。 後方互換性を保持するため、もしこのパラメータが渡されても PHP はエラーを発生させません。

返り値

成功した場合、エントリにおける次の属性を返します。エラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.2.4	ber_identifier が削除されました。 これは PHP が自動的に処理するようになりました。

参考

- ldap_get_attributes()

---

ldap_next_entry

(PHP 4, PHP 5)

ldap_next_entry — 次の結果エントリを得る

説明

resource **ldap_next_entry** (resource $link_identifier , resource $result_entry_identifier)

結果に保持されたエントリを取得します。 連続的に **ldap_next_entry()** をコールした場合、 エントリがなくなるまでエントリを一つずつ返します。 **ldap_next_entry()** への最初のコールは、 ldap_first_entry() に result_entry_identifier を指定してコールした後に、その結果を用いて行います。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

result_entry_identifier

返り値

ldap_first_entry() によりエントリが読み始められた 結果において次のエントリに関するエントリID を返します。 これ以上結果のエントリがない場合は、**FALSE**を返します。

参考

- ldap_get_entries()

---

ldap_next_reference

(PHP 4 >= 4.0.5, PHP 5)

ldap_next_reference — 次のリファレンスを得る

説明

resource **ldap_next_reference** (resource $link , resource $entry)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_parse_reference

(PHP 4 >= 4.0.5, PHP 5)

ldap_parse_reference — 参照エントリから情報を展開する

説明

bool **ldap_parse_reference** (resource $link , resource $entry , array &$referrals)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_parse_result

(PHP 4 >= 4.0.5, PHP 5)

ldap_parse_result — 結果から情報を展開する

説明

bool **ldap_parse_result** (resource $link , resource $result , int &$errcode [, string &$matcheddn [, string &$errmsg [, array &$referrals]]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_read

(PHP 4, PHP 5)

ldap_read — エントリを読み込む

説明

resource **ldap_read** (resource $link_identifier , string $base_dn , string $filter [, array $attributes [, int $attrsonly [, int $sizelimit [, int $timelimit [, int $deref]]]]])

指定したフィルタ filter を使用し、スコープ **LDAP_SCOPE_BASE** でディレクトリを検索します。これは、 ディレクトリからエントリを読み込むことと同じ意味です。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

base_dn

ディレクトリのベース DN。

filter

空のフィルタは指定できません。 このエントリに関する全ての情報を完全に取得したい場合は、 objectClass=* というフィルタを使用してください。 ディレクトリサーバーで使用されるエントリの型が分かっている場合、 objectClass=inetOrgPerson のように適切なフィルタを使用することができます。

attributes

必要な属性を、 array("mail", "sn", "cn") のような通常の PHP 文字列配列で保持します。 "dn" は要求された属性の型によらず常に返されることに注意してください。

このパラメータを使用すると、デフォルトの動作よりもかなり効率的になります（デフォルトでは、すべての属性とその値を返します)。 したがって、これを使用することを推奨します。

attrsonly

属性の型のみを取得したい場合は 1 を設定します。 属性の型および値の両方を取得したい場合は 0 を設定します（これがデフォルトの挙動です)。

sizelimit

取得するエントリ数の制限を設定します。 0 は無制限であることを表します。

> **注意**: このパラメータは、サーバ側で事前に設定されている sizelimit を上書きすることはできません。それ以下の値を指定することはできます。
> ディレクトリサーバのホストによっては、 事前に設定された数以上のエントリを返さないようになっているものもあります。 この場合、サーバでは、それが結果セットのすべてではないことを通知します。 このパラメータでエントリ数を制限した場合にも、同じことが起こります。

timelimit

検索に要する最大秒数を設定します。 これを 0 にすると無制限であることを表します。

> **注意**: このパラメータは、サーバ側で事前に設定されている timelimit を上書きすることはできません。それ以下の値を指定することはできます。

deref

検索時のエイリアスの扱いについて指定します。 以下のいずれかとなります。

- **LDAP_DEREF_NEVER** - （デフォルト）エイリアスは参照されません。
- **LDAP_DEREF_SEARCHING** - エイリアスを参照しますが、検索のベースオブジェクト上にいるときは参照しません。
- **LDAP_DEREF_FINDING** - エイリアスの参照は、ベースオブジェクト上にいて検索中でない場合に行われます。
- **LDAP_DEREF_ALWAYS** - エイリアスを常に参照します。 always.

返り値

検索結果 ID を返します。エラーの場合は、**FALSE** を返します。

変更履歴

バージョン	説明
4.0.5	並列検索のサポートが追加されました。詳細は ldap_search() を参照ください。
4.0.2	attrsonly 、sizelimit 、 timelimit および deref が追加されました。

---

ldap_rename

(PHP 4 >= 4.0.5, PHP 5)

ldap_rename — エントリ名を修正する

説明

bool **ldap_rename** (resource $link_identifier , string $dn , string $newrdn , string $newparent , bool $deleteoldrdn)

dn で指定したエントリについて、 名前の変更または移動を行います。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

dn

LDAP エンティティの識別名。

newrdn

新しい RDN。

newparent

新しい親エントリ。

deleteoldrdn

***TRUE** の場合は古い RDN 値を削除します。それ以外の場合は古い RDN 値がそのエントリの non-distinguished 値として残されます。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

注意

> ***注意**: この関数は、現在、LDAPv3 でのみ動作します。LDAPv3 でバインドする前に **ldap_set_option()()** を使用する必要があるかもしれません。 この関数は、OpenLDAP 2.x.x または Netscape Directory SDK x.x を使用している場合にのみ使用可能です。*

参考

- *ldap_modify()*

---

ldap_sasl_bind

(PHP 5)

ldap_sasl_bind — SASL を使用して LDAP ディレクトリにバインドする

説明

*bool **ldap_sasl_bind** (resource $link [, string $binddn [, string $password [, string $sasl_mech [, string $sasl_realm [, string $sasl_authz_id [, string $props]]]]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

注意

> ***注意**: **前提条件** **ldap_sasl_bind()** は SASL サポート (sasl.h) を必要とします。PHP の configure 時に --with-ldap-sasl が指定されている ことを確認してください。さもないとこの関数は未定義となります。*

---

ldap_search

(PHP 4, PHP 5)

ldap_search — LDAP ツリーを探索する

説明

*resource **ldap_search** (resource $link_identifier , string $base_dn , string $filter [, array $attributes [, int $attrsonly [, int $sizelimit [, int $timelimit [, int $deref]]]]])*

*指定したフィルタを使用し、 スコープ **LDAP_SCOPE_SUBTREE** でディレクトリを検索します。これは、ディレクトリ全体を検索するのと同じ意味です。*

*4.0.5 以降、並列検索も可能となっています。並列検索を行うには、単一の ID を使うのではなく、リンク ID の配列を使用します。 同じベース DN を使用したくない場合や全ての検索について同じフィルタを 使用したくない場合、ベース DN の配列またはフィルタの配列を使用する ことが可能です。これらの配列は、リンク ID の配列と同じ大きさである 必要があります。これは、その配列の最初が一回の検索で使用され、2 番目の エントリが他の検索で使用されるといったようになるからです。並列検索を 実行する際、エラーの場合を除き、検索結果 ID の配列が返されます。 エラーの場合は対応する検索のエントリは **FALSE** となります。これは 通常返される値とよく似ていますが、検索が行われた際に結果 ID が常に 返されます。まれに、並列検索は ID を返すにもかかわらず通常の検索は **FALSE** を返すということがあります。*

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

base_dn

ディレクトリのベース DN。

filter

検索フィルタは、LDAP ドキュメントに記述されたフォーマットの論理 演算子を用いて、簡単なものまたは高度なものとすることができます (フィル

タに関する詳細な情報については、 » Netscape Directory SDK を参照ください)。

attributes

必要な属性を、 array("mail", "sn", "cn") のような通常の PHP 文字列配列で保持します。 "dn" は要求された属性の型によらず常に返されることに注意してください。

このパラメータを使用すると、デフォルトの動作よりもかなり効率的になります (デフォルトでは、すべての属性とその値を返します)。 したがって、これを使用することを推奨します。

attrsonly

属性の型のみを取得したい場合は 1 を設定します。 属性の型および値の両方を取得したい場合は 0 を設定します (これがデフォルトの挙動です)。

sizelimit

取得するエントリ数の制限を設定します。 0 は無制限であることを表します。

> **注意**: このパラメータは、サーバ側で事前に設定されている sizelimit を上書きすることはできません。それ以下の値を指定することはできます。
> ディレクトリサーバのホストによっては、 事前に設定された数以上のエントリを返さないようになっているものもあります。 この場合、サーバでは、それが結果セットのすべてではないことを通知します。 このパラメータでエントリ数を制限した場合にも、同じことが起こります。

timelimit

検索に要する最大秒数を設定します。 これを 0 にすると無制限であることを表します。

> **注意**: このパラメータは、サーバ側で事前に設定されている timelimit を上書きすることはできません。それ以下の値を指定することはできます。

deref

検索時のエイリアスの扱いについて指定します。 以下のいずれかとなります。

- **LDAP_DEREF_NEVER** - (デフォルト) エイリアスは参照されません。
- **LDAP_DEREF_SEARCHING** - エイリアスを参照しますが、検索のベースオブジェクト上にいるときは参照しません。
- **LDAP_DEREF_FINDING** - エイリアスの参照は、ベースオブジェクト上にいて検索中でない場合に行われます。
- **LDAP_DEREF_ALWAYS** - エイリアスを常に参照します。

返り値

検索結果 ID を返します。エラーの場合は、**FALSE** を返します。

変更履歴

バージョン	説明
4.0.5	並列検索のサポートが追加されました。
4.0.2	attrsonly 、 sizelimit 、 timelimit および deref が追加されました。

例

以下の例は、"My Company" の全員について姓または名の一部に文字列 $person を含む人の組織単位、姓、名、電子メールアドレスを取得します。この例は、複数の属性に関する情報についてサーバに検索をかける論理 フィルタを使用します。

Example#1 LDAP 検索

```
<?php
// $ds は、ディレクトリサーバの有効なリンク ID

// $person は、人名またはその一部。例 "Jo"

$dn = "o=My Company, c=US";
$filter="(|(sn=$person*)(givenname=$person*))";
$justthese = array("ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filter, $justthese);

$info = ldap_get_entries($ds, $sr);

echo $info["count"]." 個のエントリが返されました\n";
?>
```

---

ldap_set_option

(PHP 4 >= 4.0.4, PHP 5)

ldap_set_option — 指定したオプションの値を設定する

説明

bool **ldap_set_option** (resource $link_identifier , int $option , mixed $newval)

指定したオプションの値を newval に設定します。

パラメータ

link_identifier

ldap_connect() が返す LDAP リンク ID。

option

パラメータ option は以下のいずれかとなります。

オプション	型
LDAP_OPT_DEREF	integer
LDAP_OPT_SIZELIMIT	integer
LDAP_OPT_TIMELIMIT	integer
LDAP_OPT_NETWORK_TIMEOUT	integer
LDAP_OPT_PROTOCOL_VERSION	integer
LDAP_OPT_ERROR_NUMBER	integer
LDAP_OPT_REFERRALS	bool
LDAP_OPT_RESTART	bool
LDAP_OPT_HOST_NAME	string
LDAP_OPT_ERROR_STRING	string
LDAP_OPT_MATCHED_DN	string
LDAP_OPT_SERVER_CONTROLS	array
LDAP_OPT_CLIENT_CONTROLS	array

***LDAP_OPT_SERVER_CONTROLS** および **LDAP_OPT_CLIENT_CONTROLS** はコントロールのリストを必要とします。 これは、値がコントロールの配列である必要があることです。 コントロールは、そのコントロールの ID である oid、オプションの value、オプションのフラグ criticality からなります。PHP において、 コントロールはキーが oid で値が文字列、 二つのオプションの要素からなる配列で指定されます。オプションの要素は、キーが value で値が文字列、そしてキーが iscritical で値が論理値となります。 iscritical を省略した場合のデフォルトは **FALSE** です。詳細は » draft-ietf-ldapext-ldap-c-api-xx.txt を参照ください。また以下の二番目の例も参照ください。*

newval

指定したオプション option の新しい値。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 プロトコルバージョンの設定

```
<?php
// $ds はディレクトリサーバーへの有効なリンクIDです
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3)) {
    echo "LDAPv3 を使用します";
} else {
    echo "プロトコルバージョンを 3 に設定できませんでした";
}
?>
```

Example#2 サーバコントロールの設定

```
<?php
// $ds は値を持たないディレクトリサーバコントロールへの有効なリンクID
// です。
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => true);
// iscritical のデフォルトは、FALSE です。
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// 両方のコントロールを試します
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2))) {
    echo "サーバコントロールの設定に失敗しました。";
}
?>
```

注意

> ***注意**: この関数は、OpenLDAP 2.x.x または Netscape Directory SDK x.x を使用している場合のみ利用可能です。*

参考

- ldap_get_option()

---

ldap_set_rebind_proc

(PHP 4 >= 4.2.0, PHP 5)

ldap_set_rebind_proc — 参照先を再バインドするためのコールバック関数を設定する

説明

*bool **ldap_set_rebind_proc** (resource $link , callback $callback)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_sort

(PHP 4 >= 4.2.0, PHP 5)

ldap_sort — LDAP 結果エントリをソートする

説明

bool **ldap_sort** (resource $link , resource $result , string $sortfilter)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_start_tls

(PHP 4 >= 4.2.0, PHP 5)

ldap_start_tls — TLS を開始する

説明

bool **ldap_start_tls** (resource $link)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_t61_to_8859

(PHP 4 >= 4.0.2, PHP 5)

ldap_t61_to_8859 — t61 文字を 8859 文字に変換する

説明

string **ldap_t61_to_8859** (string $value)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ldap_unbind

(PHP 4, PHP 5)

ldap_unbind — LDAP ディレクトリへのバインドを解除する

説明

bool **ldap_unbind** (resource $link_identifier)

LDAP ディレクトリへのバインドを解除します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ldap_bind()

目次

- ldap_free_result — 結果メモリを開放する
- ldap_get_attributes — 検索結果エントリから属性を得る
- ldap_get_dn — 結果エントリから DN を得る
- ldap_get_entries — 全ての結果エントリを得る
- ldap_get_option — 指定したオプションの現在の値を得る
- ldap_get_values_len — 結果エントリから全てのバイナリ値を得る
- ldap_get_values — 結果エントリから全ての値を得る
- ldap_list — 単一階層の検索を行う
- ldap_mod_add — 現在の属性に属性を追加する
- ldap_mod_del — 現在の属性から属性を削除する
- ldap_mod_replace — 属性を新規の値に置換する
- ldap_modify — LDAP エントリを修正する
- ldap_next_attribute — 結果における次の属性を得る
- ldap_next_entry — 次の結果エントリを得る
- ldap_next_reference — 次のリファレンスを得る
- ldap_parse_reference — 参照エントリから情報を展開する
- ldap_parse_result — 結果から情報を展開する
- ldap_read — エントリを読み込む
- ldap_rename — エントリ名を修正する
- ldap_sasl_bind — SASL を使用して LDAP ディレクトリにバインドする
- ldap_search — LDAP ツリーを探索する
- ldap_set_option — 指定したオプションの値を設定する
- ldap_set_rebind_proc — 参照先を再バインドするためのコールバック関数を設定する
- ldap_sort — LDAP 結果エントリをソートする
- ldap_start_tls — TLS を開始する
- ldap_t61_to_8859 — t61 文字を 8859 文字に変換する
- ldap_unbind — LDAP ディレクトリへのバインドを解除する

---

libxml 関数

導入

以下の関数/定数は、PHP 5.1.0 以降、 DOM, SimpleXML および XSLT のような libxml に基く エクステンションのどれかをコンパイルしている場合に利用可能です。

要件

このエクステンションは、» libxml >= 2.6.0を必要とします。

定義済みクラス

LibXMLError

プロパティ

- code - エラーのコード。
- column - エラーが発生した列。このプロパティは libxml では完全には 実装されておらず、しばしば 0 を返すことがある ということに注意しましょう。
- file - ファイル名。もし XML が文字列から読み込まれた場合は空文字列。
- level - エラーの深刻度（以下の定数 **LIBXML_ERR_WARNING**、 **LIBXML_ERR_ERROR** あるいは **LIBXML_ERR_FATAL** のいずれか）。
- line - エラーが発生した行数。
- message - エラーメッセージ。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

LIBXML_COMPACT (integer)
小さなノードを割り当てるように最適化する。アプリケーションの コードを変更することなしにスピードアップさせることができる

> **注意**: Libxml >= 2.6.21 でのみ有効

LIBXML_DTDATTR (integer)
デフォルトのDTD属性
LIBXML_DTDLOAD (integer)
外部サブセットをロードする
LIBXML_DTDVALID (integer)
DTDで検証する
LIBXML_NOBLANKS (integer)

空白のノードを削除
LIBXML_NOCDATA (integer)
CDATA をテキストノードとしてマージ
LIBXML_NOEMPTYTAG (integer)
空タグを拡張する (例
 を
</br> にする)

注意: このオプションは、現在 DOMDocument->save() および DOMDocument->saveXML() 関数でのみ有効です。

LIBXML_NOENT (integer)
エンティティを置換
LIBXML_NOERROR (integer)
エラー出力を抑制
LIBXML_NONET (integer)
ドキュメントロード時にネットワークアクセスを無効にする
LIBXML_NOWARNING (integer)
警告出力を抑制する
LIBXML_NOXMLDECL (integer)
ドキュメントの保存時に XML 宣言を削除する

注意: Libxml >= 2.6.21 でのみ有効

LIBXML_NSCLEAN (integer)
冗長な名前空間宣言を削除する
LIBXML_XINCLUDE (integer)
XInclude 置換を実装する
LIBXML_ERR_ERROR (integer)
復帰可能なエラー
LIBXML_ERR_FATAL (integer)
致命的なエラー
LIBXML_ERR_NONE (integer)
エラーなし
LIBXML_ERR_WARNING (integer)
単純な警告
LIBXML_VERSION (integer)
20605 や 20617 のような libxml のバージョン
LIBXML_DOTTED_VERSION (string)
2.6.5または2.6.17のようなlibxmlのバージョン

---

libxml_clear_errors

(PHP 5 >= 5.1.0)

libxml_clear_errors — libxmlエラーハンドラをクリアする

説明

void **libxml_clear_errors** (void)

libxml_clear_errors()は、libxmlエラーバッファをクリアします。

返り値

値を返しません。

参考

- libxml_get_errors()
- libxml_get_last_error()

---

libxml_get_errors

(PHP 5 >= 5.1.0)

libxml_get_errors — エラー配列を取得する

説明

array **libxml_get_errors** (void)

エラー配列を取得します。

返り値

エラーがバッファにある場合に LibXMLError オブジェクトの配列、 それ以外の場合に空の配列を返します。

例

Example#1 libxml_get_errors() の例

この例は、簡単な libxml エラーハンドラを構築する方法を示すものです。

```
<?php

libxml_use_internal_errors(true);

$xmlstr = <<< XML
<?xml version='1.0' standalone='yes'?>
<movies>
```

```
  <movie>
    <titles>PHP: Behind the Parser</title>
  </movie>
</movies>
XML;

$doc = simplexml_load_string($xmlstr);
$xml = explode("¥n", $xmlstr);

if (!$doc) {
    $errors = libxml_get_errors();

     foreach ($errors as $error) {
         echo display_xml_error($error, $xml);
    }

    libxml_clear_errors();
}


function display_xml_error($error, $xml)
{
    $return  = $xml[$error->line - 1] . "¥n";
    $return .= str_repeat('-', $error->column) . "^¥n";

     switch ($error->level) {
         case LIBXML_ERR_WARNING:
             $return .= "Warning $error->code: ";
              break;
          case LIBXML_ERR_ERROR:
             $return .= "Error $error->code: ";
              break;
         case LIBXML_ERR_FATAL:
             $return .= "Fatal Error $error->code: ";
              break;
    }

    $return .= trim($error->message) .
               "¥n  Line: $error->line" .
               "¥n   Column: $error->column";

    if ($error->file) {
        $return .= "¥n   File: $error->file";
    }

     return "$return¥n¥n--------------------------------------------¥n¥n";
}

?>
```

上の例の出力は以下となります。

```
   <titles>PHP: Behind the Parser</title>
^
Fatal Error 76: Opening and ending tag mismatch: titles line 4 and title
  Line: 4
  Column: 0

--------------------------------------------
```

参考

- libxml_get_last_error()
- libxml_clear_errors()

---

libxml_get_last_error

(PHP 5 >= 5.1.0)

libxml_get_last_error — libxmlから直近のエラーを取得する

説明

LibXMLError **libxml_get_last_error** (void)

libxmlから直近のエラーを取得します。

返り値

エラーがバッファにある場合にLibXMLErrorオブジェクト、 それ以外の場合に **FALSE** を返します。

参考

- libxml_get_errors()
- libxml_clear_errors()

---

libxml_set_streams_context

(PHP 5)

libxml_set_streams_context — 次のlibxmlドキュメントの読込／書きこみのためにストリームコンテキストを設定する

説明

void **libxml_set_streams_context** (resource $streams_context)

次のlibxmlドキュメントの読込／書きこみのためにストリームコンテキストを設定します。

パラメータ

streams_context

ストリームコンテキストリソース(stream_context_create()で作成)

返り値

値を返しません。

例

Example#1 libxml_set_streams_context() の例

```
<?php

$opts = array(
    'http' => array(
        'user_agent' => 'PHP libxml agent',
    )
);

$context = stream_context_create($opts);
libxml_set_streams_context($context);

// HTTPによりファイルをリクエスト
$doc = DOMDocument::load('http://www.example.com/file.xml');

?>
```

参考

- stream_context_create()

---

libxml_use_internal_errors

(PHP 5 >= 5.1.0)

libxml_use_internal_errors — libxmlエラーを無効にし、ユーザが必要に応じてエラー情報を取得できるようにする

説明

bool **libxml_use_internal_errors** ([bool $use_errors])

libxml_use_internal_errors() により、 標準のlibxmlエラーを無効にし、ユーザによるエラー処理を有効にすることができます。

パラメータ

use_errors

ユーザによるエラー処理を有効または無効にする。デフォルトは、**FALSE**です。

返り値

この関数は、 use_errors の前の値を返します。

例

Example#1 libxml_use_internal_errors()の例

この例は、libxmlエラーとこの関数により返された値の 貴本的な使用方法に示すものです。

```
<?php

// ユーザによるエラー処理を有効にする
var_dump(libxml_use_internal_errors(true));

$doc = DOMDocument::load('file.xml');

if (!$doc) {
    $errors = libxml_get_errors();
    foreach ($errors as $error) {
        // handle errors here
    }

    libxml_clear_errors();
```

```
}

?>
```

上の例の出力は以下となります。

```
bool(false)
```

参考

- libxml_clear_errors()
- libxml_get_errors()

目次

Lotus Notes 関数

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 5.0.0. この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/notes.

警告

この拡張モジュールは現在サポートされていません。メンテナを募集中です。

---

notes_body

(PHP 4 >= 4.0.5)

notes_body — 指定したサーバにある指定したメールボックスのメッセージ msg_number をオープンする

説明

```
array notes_body ( string $server , string $mailbox , int $msg_number )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_copy_db

(PHP 4 >= 4.0.5)

notes_copy_db — Lotus Notes データベースをコピーする

説明

```
bool notes_copy_db ( string $from_database_name , string $to_database_name )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_create_db

(PHP 4 >= 4.0.5)

notes_create_db — Lotus Notes データベースを作成する

説明

bool **notes_create_db** (string $database_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_create_note

(PHP 4 >= 4.0.5)

notes_create_note — フォーム form_name を用いてノートを作成する

説明

bool **notes_create_note** (string $database_name , string $form_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_drop_db

(PHP 4 >= 4.0.5)

notes_drop_db — Lotus Notes データベースを破棄する

説明

bool **notes_drop_db** (string $database_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_find_note

(PHP 4 >= 4.0.5)

notes_find_note — database_name で見つかったノートの ID を返す

説明

int **notes_find_note** (string $database_name , string $name [, string $type])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_header_info

(PHP 4 >= 4.0.5)

notes_header_info — 指定したサーバ上の指定したメールボックスにあるメッセージ msg_number をオープンする

説明

object ***notes_header_info*** *(string $server , string $mailbox , int $msg_number)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_list_msgs

(PHP 4 >= 4.0.5)

notes_list_msgs — 選択した database_name からノートを返す

説明

bool ***notes_list_msgs*** *(string $db)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_mark_read

(PHP 4 >= 4.0.5)

notes_mark_read — ユーザ user_name 用に note_id に既読マークを付ける

説明

bool ***notes_mark_read*** *(string $database_name , string $user_name , string $note_id)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_mark_unread

(PHP 4 >= 4.0.5)

notes_mark_unread — ユーザ user_name 用に note_id に未読マークを付ける

説明

bool ***notes_mark_unread*** *(string $database_name , string $user_name , string $note_id)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_nav_create

(PHP 4 >= 4.0.5)

notes_nav_create — database_name にナビゲータ名を作成する

説明

bool ***notes_nav_create*** *(string $database_name , string $name)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_search

(PHP 4 >= 4.0.5)

notes_search — database_name のキーワードにマッチするノートを見つける

説明

array **notes_search** (string $database_name , string $keywords)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_unread

(PHP 4 >= 4.0.5)

notes_unread — 現在のユーザ user_name に関して未読のノート ID を返す

説明

array **notes_unread** (string $database_name , string $user_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

notes_version

(PHP 4 >= 4.0.5)

notes_version — Lotus Notes のバージョンを取得する

説明

float **notes_version** (string $database_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

LZF 関数

導入

LZF は非常に高速な圧縮アルゴリズムで、わずかな速度の低下と引き換えに スペースを節約したい場合に適しています。 コンパイル時に速度もしくはスペースのどちらを優先して最適化を行うかを 選択することができます。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/lzf.

これらの関数を使用するには、--with-lzf[=DIR] オプションを指定して PHP を lzf サポートつきでコンパイルする必要があります。 --enable-lzf-better-compression を指定して、 圧縮速度より圧縮率を優先するよう LZF を設定することも可能です。

Windows ユーザは、php.ini で php_lzf.dll を有効にすることでこれらの関数を使用可能です。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

---

lzf_compress

(PECL lzf:0.1-1.4)

lzf_compress — LZF 圧縮を行う

説明

string **lzf_compress** (string $data)

lzf_compress() は、指定された arg を圧縮します。

パラメータ

data

圧縮するデータ。

返り値

圧縮されたデータを返します。エラー時には **FALSE** を返します。

参考

- lzf_decompress()

---

lzf_decompress

(PECL lzf:0.1-1.4)

lzf_decompress — LZF 圧縮を解凍する

説明

string **lzf_decompress** (string $data)

lzf_decompress() は、指定された data を解凍します。

パラメータ

data

圧縮されたデータ。

返り値

解凍されたデータを返します。エラー時には **FALSE** を返します。

参考

- lzf_compress()

---

lzf_optimized_for

(PECL lzf:1.0-1.4)

lzf_optimized_for — LZF 拡張モジュールの最適化指定を取得する

説明

int **lzf_optimized_for** (void)

LZF 拡張モジュールの最適化指定を取得します。

返り値

LZF が速度に最適化されている場合に 1、圧縮率に最適化されている場合に 0 を返します。

目次

メール関数(Mail)

導入

mail() 関数によりメールを送信できるようになります。

要件

メール関数を使用可能にするには、PHP のコンパイル時点でシステム上の sendmail バイナリにアクセスできなければなりません。 postfix や qmail など他のメールプログラムを使用してる場合には、 それらのソフトに付随する適当な sendmail ラッパを使用するように 気をつけてください。 PHP は sendmail を探す際にまず PATH を見ます。次に /usr/bin:/usr/sbin:/usr/etc:/etc:/usr/ucblib:/usr/lib の順で探します。PATH を通して sendmail を使用可能な 状態にしておくことが強く推奨されます。 また、コンパイルされた PHP が sendmail バイナリにアクセスできる権限を 持っていなければなりません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

Mail 設定オプション

名前	デフォルト	変更の可否	変更履歴
SMTP	"localhost"	PHP_INI_ALL	
smtp_port	"25"	PHP_INI_ALL	
sendmail_from	NULL	PHP_INI_ALL	
sendmail_path	"/usr/sbin/sendmail -t -i"	PHP_INI_SYSTEM	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

SMTP string

Windows 環境のみで使用されます: mail() 関数でメールを送信する際に使用する SMTP サーバのホスト名または IP アドレス。

smtp_port int

Windows 環境のみで使用されます: SMTP 設定で指定したサーバに接続するポート番号で、 デフォルトは 25 です。PHP 4.3.0 以降でのみ利用可能です。

sendmail_from string

Windows 環境で、PHP から送信されるメールにおいて "From:" に使用するメールアドレスを指定します。 このディレクティブは、"Return-Path:" ヘッダも設定します。

sendmail_path string

sendmail プログラムを探すパスを指定します。 通常、/usr/sbin/sendmail または /usr/lib/sendmail です。 **configure** は、このパスを探し、 デフォルト値として設定しますが、 これで上手くいかない場合にはこのオプションで設定する必要があります。

sendmail を使用していないシステムは、 使用するメールシステムが提供する sendmail のラッパ/代替品を、 必要に応じてこのディレクティブに設定する必要があります。例えば、» Qmail ユーザは、通常 /var/qmail/bin/sendmail または /var/qmail/bin/qmail-inject に設定します。

qmail-inject では、 メールを正しく処理するためのオプション設定は不要です。

このディレクティブは Windows 環境でも動作します。指定された場合は smtp および smtp_port・ sendmail_from の値は無視され、 ここで指定したコマンドが実行されます。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

ezmlm_hash

(PHP 4 >= 4.0.2, PHP 5)

ezmlm_hash — EZMLM で必要なハッシュ値を計算する

説明

int **ezmlm_hash** (string $addr)

ezmlm_hash() は、EZMLM メイリングリストを MySQL データベースで管理する際に必要なハッシュ値を計算します。

パラメータ

addr

ハッシュ値を計算する email アドレス。

返り値

addr のハッシュ値を返します。

例

Example#1 ハッシュ値を計算し、ユーザを登録する

```
<?php

$user = "joecool@example.com";
$hash = ezmlm_hash($user);
$query = sprintf("INSERT INTO sample VALUES (%s, '%s')", $hash, $user);
$db->query($query); // PHPLIB dbインターフェースを使用します

?>
```

---

mail

(PHP 4, PHP 5)

mail — メールを送信する

説明

bool **mail** (string $to , string $subject , string $message [, string $additional_headers [, string $additional_parameters]])

メールを送信します。

パラメータ

to

メールの受信者。

» RFC 2822 を満たす書式でなければ なりません。例えば以下のようなものです。

- user@example.com
- user@example.com, anotheruser@example.com
- User <user@example.com>
- User <user@example.com>, Another User <anotheruser@example.com>

subject

送信するメールの表題。

警告

改行を含んではいけません。含めてしまうとメールが正しく送信できません。

message

送信するメッセージ。

改行コードは LF (\n) となります。各行の長さは 70 文字を超えては いけません。

警告

(Windows のみ) PHP が SMTP サーバと直接通信をする際、ピリオドから 始まる行は無視されます。これを防ぐには、行頭のピリオドを ピリオド 2 つに置き換えてください。

```
<?php
$text = str_replace("¥n.", "¥n..", $text);
?>
```

additional_headers (オプション)

メールヘッダの最後に挿入される文字列。

通常、これは追加のヘッダ（From、Cc、Bcc）のために用いられます。 複数のヘッダを追加する場合は CRLF（\r\n）で区切ります。

> **注意**: メールを送信する際には、必ず From ヘッダが含まれていなければなりません。 additional_headers パラメータで指定するか、あるいは php.ini にデフォルト値を指定します。
> 指定しなかった場合は、以下のようなエラーメッセージが返ります Warning: mail(): "sendmail_from" not set in php.ini or custom "From:" header missing 。 Windows では、From ヘッダを設定すると Return-Path も設定されます。

> **注意**: メッセージが受信されなかった場合には、LF（\n）のみを使ってみてください。 Unix の MTA の中には、自動的に LF を CRLF に変換してしまう ものがあります（もし CRLF を利用していた場合、CR が重複してしまいます）。 ただし、これは最後の手段です。というのも、これは » RFC 2822 に違反しているからです。

additional_parameters （オプション）

パラメータ additional_parameters は、 追加のパラメータをメール送信プログラムに渡す際に使用可能です。 メール送信プログラムは、設定オプション sendmail_path により設定されます。例えば、 sendmail を使用する際に -f オプションを使って エンベロープの sender アドレスを設定する際に使用できます。

この方法でエンベロープの sender ヘッダ（-f）を設定する際は、 'X-Warning' ヘッダが付加されないように Web サーバの実行ユーザを sendmail 設定に追加しておく必要があるかもしれません。 sendmail を利用している場合、これは /etc/mail/trusted-users で設定します。

返り値

メール送信が受け入れられた場合に **TRUE** 、それ以外の場合に **FALSE** を返します。

メールの配送が受け入れられたかどうかが基準であることに注意しましょう。 メールが実際にあて先に届いたかどうかでは「ありません」。

変更履歴

バージョン	説明
4.3.0（Windows のみ）	すべてのカスタムヘッダ（From、Cc、Bcc や Date など）がサポート されるようになり、大文字・小文字を区別しないようになりました （カスタムヘッダは MTA でパースされるのではなく PHP でパースされます。 PHP < 4.3 では Cc ヘッダのみがサポートされており、大文字・小文字が 区別されていました）。
4.2.3	safe_mode 時には additional_parameters パラメータを無効に しました。もし用いられた場合には mail() 関数は 警告メッセージを出して **FALSE** を返します。
4.0.5	additional_parameters パラメータが追加されました。

例

Example#1 メールを送信する

mail() を用いて単純なメールを送信する。

```
<?php
// 本文
$message = "Line 1¥nLine 2¥nLine 3";

// 1 行が 70 文字を超える場合のため、wordwrap() を用いる
$message = wordwrap($message, 70);

// 送信する
mail('caffinated@example.com', 'My Subject', $message);
?>
```

Example#2 追加ヘッダを付加してメールを送信する

基本ヘッダに加え、MUA に From および Reply-To アドレスを通知する。

```
<?php
$to      = 'nobody@example.com';
$subject = 'the subject';
$message = 'hello';
$headers = 'From: webmaster@example.com' . "¥r¥n" .
    'Reply-To: webmaster@example.com' . "¥r¥n" .
    'X-Mailer: PHP/' . phpversion();

mail($to, $subject, $message, $headers);
?>
```

Example#3 追加ヘッダ及び追加コマンドラインパラメータを指定してメールを送信する

sendmail_path を用いてメールを送信する際に利用する 追加パラメータとして、additional_parameters が用いられます。

```
<?php
mail('nobody@example.com', 'the subject', 'the message', null,
   '-fwebmaster@example.com');
?>
```

Example#4 HTML メールを送信する

mail() を用いて HTML メールを送信することも可能です。

```
<?php
// 複数の受信者を指定
$to  = 'aidan@example.com' . ', '; //  カンマに注意
$to .= 'wez@example.com';

// 表題
$subject = 'Birthday Reminders for August';

// 本文
```

```
$message = '
<html>
<head>
   <title>Birthday Reminders for August</title>
</head>
<body>
   <p>Here are the birthdays upcoming in August!</p>
   <table>
     <tr>
         <th>Person</th><th>Day</th><th>Month</th><th>Year</th>
     </tr>
     <tr>
         <td>Joe</td><td>3rd</td><td>August</td><td>1970</td>
     </tr>
     <tr>
         <td>Sally</td><td>17th</td><td>August</td><td>1973</td>
     </tr>
   </table>
</body>
</html>
';

// HTML メールを送信するには Content-type ヘッダが必須
$headers  = 'MIME-Version: 1.0' . "¥r¥n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "¥r¥n";

// 追加のヘッダ
$headers .= 'To: Mary <mary@example.com>, Kelly <kelly@example.com>' . "¥r¥n";
$headers .= 'From: Birthday Reminder <birthday@example.com>' . "¥r¥n";
$headers .= 'Cc: birthdayarchive@example.com' . "¥r¥n";
$headers .= 'Bcc: birthdaycheck@example.com' . "¥r¥n";

// 送信する
mail($to, $subject, $message, $headers);
?>
```

注意: HTML などの複雑な形式のメールを送信する場合は、PEAR パッケージ » PEAR::Mail_Mime を利用することを推奨します。

注意

注意: **mail()** の Windows 版実装は、Unix 版実装とは 多くの点で違います。 第一に、メッセージの生成にローカルのバイナリは使用せず、 ソケットを通じて直接操作するだけです。 これは MTA がネットワークソケットを listen している 必要があるということを意味します（ローカルホスト、リモートマシン どちらでもかまいません）。
第二に、 From: や Cc:・ Bcc:・ Date: のようなカスタムヘッダは MTA ではなく PHP によってパースされます。
そのため、to 引数には "Something <someone@example.com>" 形式の メールアドレスを与えることはできません。 MTA と通信する際に mail コマンドはこれを適切にパースできません。

注意: 添付ファイルや特殊な本文（HTML など）を含むメールも、この関数で 送信可能です。これは、MIME エンコーディングによって実現されています。 詳細な情報は » Zend の記事 または » PEAR Mime クラス を参照ください。

注意: **mail()** 関数は、大量のメールをループ内で送信するには 向いていないことに注意しましょう。この関数は 1 通のメールを送信するたびに SMTP ソケットをいったん閉じて開きなおします。これは非効率的です。
大量のメールを送信する場合は、 » PEAR::Mail および » PEAR::Mail_Queue パッケージを参照ください。

注意: 以下の RFC も有用です。 » RFC 1896、 » RFC 2045、 » RFC 2046、 » RFC 2047、 » RFC 2048、 » RFC 2049 および » RFC 2822

参考

- imap_mail()
- » PEAR::Mail
- » PEAR::Mail_Mime

目次

Mailparse 関数

導入

Mailparse は、電子メールのメッセージをパースして処理するための拡張モジュールです。 » RFC 822 および » RFC 2045 (MIME) に準拠したメッセージを扱うことができます。

Mailparse はストリームベースで動作します。つまり、 処理する際にファイルの内容をメモリにコピーする必要がないということです。 これにより、大きなメッセージを扱う際にリソースを効率的に使用することができます。

注意: Mailparse は、mbstring 拡張モジュールを必要とします。 また、mbstring は mailparse より前に読み込まれていなければなりません。

この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 4.2.0.

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報につ

いては、次の場所にあります。 » http://pecl.php.net/package/mailparse.

これらの関数を使用するためには、configure 時に --enable-mailparse オプションを指定し、 mailparse のサポートつきで PHP をコンパイルする必要があります。

Windows ユーザがこれらの関数を利用するには、php.ini の中で php_mailparse.dll を有効にします。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

php.ini の設定により動作が変化します。

Mailparse の設定オプション

名前	デフォルト	変更の可否	変更履歴
mailparse.def_charset	"us-ascii"	PHP_INI_ALL	PHP 4.1.0 以降で使用可能です。PHP 4.2.0 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

MAILPARSE_EXTRACT_OUTPUT (integer)
MAILPARSE_EXTRACT_STREAM (integer)
MAILPARSE_EXTRACT_RETURN (integer)

---

mailparse_determine_best_xfer_encoding

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_determine_best_xfer_encoding — 最も適したエンコーディングを取得する

説明

string **mailparse_determine_best_xfer_encoding** (resource $fp)

ファイルポインタから内容を読み込む際に最も適したエンコーディングを調べます。

パラメータ

fp

有効なファイルポインタ。これはシーク可能である必要があります。

返り値

mbstring モジュールがサポートする文字エンコーディングのいずれかを返します。

例

Example#1 mailparse_determine_best_xfer_encoding() の例

```
<?php

$fp = fopen('somemail.eml', 'r');
echo 'Best encoding: ' . mailparse_determine_best_xfer_encoding($fp);

?>
```

上の例の出力は、たとえば以下のようになります。

```
Best encoding: 7bit
```

---

mailparse_msg_create

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_create — mime メールリソースを作成する

説明

resource **mailparse_msg_create** (void)

MIME メールリソースを作成します。

返り値

メッセージのパースに使用するハンドルを返します。

参考

- mailparse_msg_free()
- mailparse_msg_parse_file()

---

mailparse_msg_extract_part_file

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_extract_part_file — メッセージセクションを展開/デコードする

説明

string **mailparse_msg_extract_part_file** (resource $mimemail , mixed $filename [, callback $callbackfunc])

指定したファイル名のメッセージセクションを展開/デコードします。

セクションの内容は、transfer encoding に応じて適切にデコードされます。 base64、quoted-printable および uuencode 形式をサポートしています。

パラメータ

mimemail

mailparse_msg_create() で作成した MIME リソース。

filename

ファイル名あるいは有効なストリームリソース。

callbackfunc

展開されたセクションに渡されるコールバック関数、あるいは **NULL** を指定すると、この関数は展開したセクションを返します。

省略した場合は標準出力に出力されます。

返り値

callbackfunc が **NULL** でない場合は、 成功時に **TRUE** を返します。

callbackfunc が **NULL** の場合は、 展開したセクションを文字列で返します。

エラー時には **FALSE** を返します。

参考

- mailparse_msg_extract_part()
- mailparse_msg_extract_whole_part_file()

---

mailparse_msg_extract_part

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_extract_part — メッセージセクションを展開/デコードする

説明

void **mailparse_msg_extract_part** (resource $mimemail , string $msgbody [, callback $callbackfunc])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

mimemail

有効な MIME リソース。

msgbody

callbackfunc

返り値

値を返しません。

参考

- mailparse_msg_extract_part_file()
- mailparse_msg_extract_whole_part_file()

---

mailparse_msg_extract_whole_part_file

(PECL mailparse:0.9-2.1.1)

mailparse_msg_extract_whole_part_file — ヘッダを含むメッセージセクションを、transfer encoding をデコードせずに展開する

説明

string **mailparse_msg_extract_whole_part_file** (resource $mimemail , string $filename [, callback $callbackfunc])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

mimemail

有効な MIME リソース。

filename

callbackfunc

返り値

参考

- mailparse_msg_extract_part()
- mailparse_msg_extract_part_file()

---

mailparse_msg_free

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_free — MIME リソースを解放する

説明

bool **mailparse_msg_free** (resource $mimemail)

MIME リソースを解放します。

パラメータ

mimemail

mailparse_msg_create() あるいは mailparse_msg_parse_file() が割り当てた 有効な MIME リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mailparse_msg_create()
- mailparse_msg_parse_file()

---

mailparse_msg_get_part_data

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_get_part_data — メッセージに関する情報の連想配列を返す

説明

array **mailparse_msg_get_part_data** (resource $mimemail)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

mimemail

有効な MIME リソース。

---

mailparse_msg_get_part

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_get_part — MIME メッセージの指定したセクションに関するハンドルを返す

説明

resource **mailparse_msg_get_part** (resource $mimemail , string $mimesection)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

mimemail

有効な MIME リソース。

mimesection

---

mailparse_msg_get_structure

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_get_structure — 指定したメッセージ内の MIME セクション名の配列を返す

説明

array **mailparse_msg_get_structure** (resource $mimemail)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

mimemail

有効な MIME リソース。

---

mailparse_msg_parse_file

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_parse_file — ファイルをパースする

説明

resource **mailparse_msg_parse_file** (string $filename)

ファイルをパースします。 ディスク上にあるメールファイルをパースするための最良の方法です。

パラメータ

filename

メッセージを保持するファイルへのパス。 このファイルがオープンされ、ストリームとしてパーサに流し込まれます。

返り値

構造を表す MIME リソース、あるいはエラー時に **FALSE** を返します。

参考

- mailparse_msg_free()
- mailparse_msg_create()

---

mailparse_msg_parse

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_msg_parse — データをパースし、バッファに追加する

説明

bool **mailparse_msg_parse** (resource $mimemail , string $data)

データを順にパースし、指定した mime メールリソースに格納します。

この関数により、ファイル全体を読み込んでからパースするのではなく 読み込んだ部分から順に処理していくことができます。

パラメータ

mimemail

有効な MIME リソース。

data

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

---

mailparse_rfc822_parse_addresses

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_rfc822_parse_addresses — RFC 822 準拠のアドレスをパースする

説明

*array **mailparse_rfc822_parse_addresses** (string $addresses)*

» RFC 822 準拠の受信者リスト、 たとえば To: ヘッダの内容などをパースします。

パラメータ

addresses

アドレスを含む文字列。たとえば Wez Furlong <wez@example.com>, doe@example.com のようになります。

> ***注意**: この文字列にはヘッダ名を含めてはいけません。*

返り値

各受信者について以下のキーをもつ連想配列の配列を返します。
display 表示用の受信者名。この部分が設定されていない場合は、 *address* と同じ値となります。
address メールアドレス。
is_group 受信者がニュースグループである場合に **TRUE**、そうでない場合に **FALSE**。

例

Example#1 mailparse_rfc822_parse_addresses() の例

```
<?php

$to = 'Wez Furlong <wez@example.com>, doe@example.com';
var_dump(mailparse_rfc822_parse_addresses($to));

?>
```

上の例の出力は以下となります。

```
array(2) {
  [0]=>
  array(3) {
    ["display"]=>
    string(11) "Wez Furlong"
    ["address"]=>
    string(15) "wez@example.com"
    ["is_group"]=>
    bool(false)
  }
  [1]=>
  array(3) {
    ["display"]=>
    string(15) "doe@example.com"
    ["address"]=>
    string(15) "doe@example.com"
    ["is_group"]=>
    bool(false)
  }
}
```

---

mailparse_stream_encode

(PHP 4 >= 4.0.7, PECL mailparse:0.9-2.1.1)

mailparse_stream_encode — ソースファイルポインタからストリームデータを取得し、 エンコーディングを適用し、出力ファイルポインタに書き込む

説明

*bool **mailparse_stream_encode** (resource $sourcefp , resource $destfp , string $encoding)*

ソースファイルポインタからストリームデータを取得し、 encoding を適用して それを出力ファイルポインタに書き込みます。

パラメータ

sourcefp

有効なファイルハンドル。このファイルからのストリームがパーサに流し込まれます。

destfp

エンコードしたデータを書き込むファイルハンドル。

encoding

mbstring モジュールがサポートする文字エンコーディングのいずれか。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 mailparse_stream_encode() の例

```
<?php

// email.eml の中身は: hello, this is some text=hello.
$fp = fopen('email.eml', 'r');

$dest = tmpfile();

mailparse_stream_encode($fp, $dest, "quoted-printable");

rewind($dest);

// ファイルの内容を表示します
fpassthru($dest);

?>
```

上の例の出力は以下となります。

```
hello, this is some text=3Dhello.
```

---

mailparse_uudecode_all

(PECL mailparse:0.9-2.1.1)

mailparse_uudecode_all — ファイルポインタからデータをスキャンし、uuencode されたファイルを展開する

説明

array **mailparse_uudecode_all** (resource $fp)

指定したファイルポインタからのデータを読み込み、 uuencode されたファイルを一時ファイルに展開します。

パラメータ

fp

有効なファイルポインタ。

返り値

ファイル名の情報を含む連想配列の配列を返します。
filename　　作成された一時ファイルへのパス。
origfilename もとのファイル名。uuencode されたパートにのみ存在します。
最初の filename エントリがメッセージ本文、次のエントリがデコードされた uuencode ファイルとなります。

例

Example#1 mailparse_uudecode_all() の例

```
<?php

$text = <<<EOD
To: fred@example.com

hello, this is some text hello.
blah blah blah.

begin 644 test.txt
/=&AI<R!I<R!A('1E<W0*
`
end

EOD;
```

```
$fp = tmpfile();
fwrite($fp, $text);

$data = mailparse_uudecode_all($fp);

echo "BODY¥n";
readfile($data[0]["filename"]);
echo "UUE  ({$data[1]['origfilename']})¥n";
readfile($data[1]["filename"]);

// 後始末をします
unlink($data[0]["filename"]);
unlink($data[1]["filename"]);

?>
```

上の例の出力は以下となります。

```
BODY
To:  fred@example.com

hello, this is some text hello.
blah blah blah.

UUE  (test.txt)
this is a test
```

目次

数学関数(Math)

導入

これらの数学関数は、実行するコンピューターの integer型および float型の範囲で のみ値を処理します。（これは、現在、それぞれ、C言語のlongおよびdoubleに対応します。）より大きな数を処理する必要がある場合には、 任意精度数学関数の使用をお勧めします。

マニュアルの算術演算子 のページも参照ください。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数は、PHP コアに含まれており、常に利用可能です。

Math の定数

定数	値	説明
M_PI	3.14159265358979323846	パイ（円周率）
M_E	2.7182818284590452354	e（自然対数の底）
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [5.2.0]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [5.2.0]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [5.2.0]
M_EULER	0.57721566490153286061	オイラー定数 [5.2.0]

PHP 4.0.0 以前は、M_PI のみが使用可能でした。それ以外の定数は PHP 4.0.0 以降で使用可能となり、[5.2.0] と示されている定数は PHP 5.2.0 以降で使用可能となりました。

---

abs

(PHP 4, PHP 5)

abs — 絶対値

説明

number ***abs*** *(mixed $number)*

number の絶対値を返します。

パラメータ

number

処理する数値。

返り値

number の絶対値を返します。もし number の型が float であった場合、返り値の型も float となります。それ以外の場合は 返り値の型は integer となります（float は、 integer の最大値より大きい値をとることがありえるからです）。

例

Example#1 abs() の例

```
<?php
$abs = abs(-4.2); // $abs = 4.2; (double/float)
$abs2 = abs(5);   // $abs2 = 5; (integer)
$abs3 = abs(-5);  // $abs3 = 5; (integer)
?>
```

---

acos

(PHP 4, PHP 5)

acos — 逆余弦（アークコサイン）

説明

float ***acos*** *(float $arg)*

arg のアークコサインをラジアンで返します。 acos() は cos() の逆関数で、 acos() がとりうる範囲内のすべての a について a==cos(acos(a)) が成立します。

パラメータ

arg

処理する角度。

返り値

arg のアークコサインをラジアンで返します。

参考

- *cos()*
- *acosh()*
- *asin()*
- *atan()*

---

acosh

(PHP 4 >= 4.0.7, PHP 5)

acosh — 逆双曲線余弦（アークハイパボリックコサイン）

説明

*float **acosh** (float $arg)*

arg のアークハイパボリックコサインを返します。 つまり、ハイパボリックコサインが arg となる値です。

> ***注意**: この関数は Windows 環境にはまだ実装されていません。*

パラメータ

arg

処理する値。

返り値

arg のアークハイパボリックコサインを返します。

参考

- *cosh()*
- *acos()*
- ***asinh ()***
- *atanh()*

---

asin

(PHP 4, PHP 5)

asin — 逆正弦（アークサイン）

説明

*float **asin** (float $arg)*

*arg のアークサインをラジアンで返します。 **asin()** は sin() の逆関数で、 **asin()** がとりうる範囲内のすべての a について a==sin(asin(a)) が成立します。*

パラメータ

arg

処理する角度。

返り値

arg のアークサインをラジアンで返します。

参考

- *sin()*
- *asinh()*
- *acos()*
- *atan()*

---

asinh

(PHP 4 >= 4.0.7, PHP 5)

asinh — 逆双曲線正弦（アークハイパボリックサイン）

説明

float **asinh** (float $arg)

arg のアークハイパボリックサインを返します。 つまり、ハイパボリックサインが arg となる値です。

注意: この関数は Windows 環境にはまだ実装されていません。

パラメータ

arg

処理する角度。

返り値

arg のアークハイパボリックサインを返します。

参考

- sinh()
- asin()
- acosh()
- atanh()

---

atan2

(PHP 4, PHP 5)

atan2 — 2 変数のアークタンジェント

説明

float **atan2** (float $y , float $x)

この関数は、2 つの変数 x および y のアークタンジェントを計算します。 y / x のアークタンジェントを計算するのに似ていますが、 2 つの引数の符号を用いて結果の象限を定義することが異なっています。

この関数は、結果を -PI から PI の間(両端を含む)のラジアンで返します。

パラメータ

y

被除数。

x

除数。

返り値

y /x のアークタンジェントをラジアンで返します。

参考

- atan()

---

atan

(PHP 4, PHP 5)

atan — 逆正接（アークタンジェント）

説明

float **atan** (float $arg)

arg のアークタンジェントをラジアンで返します。 **atan()** is the は tan() の逆関数で、 **atan()** がとりうる範囲内のすべての a について a==tan(atan(a)) が成立します。

パラメータ

arg

処理する引数。

返り値

arg のアークタンジェントをラジアンで返します。

参考

- tan()
- atanh()
- asin()
- acos()

---

atanh

(PHP 4 >= 4.0.7, PHP 5)

atanh — 逆双曲線正接（アークハイパボリックタンジェント）

説明

float **atanh** (float $arg)

arg のアークハイパボリックタンジェントを返します。 つまり、ハイパボリックタンジェントが arg となる値です。

注意: この関数は Windows 環境にはまだ実装されていません。

パラメータ

arg

処理する引数。

返り値

arg のアークハイパボリックタンジェントを返します。

参考

- tanh()
- **atan ()**
- asinh()
- acosh()

---

base_convert

(PHP 4, PHP 5)

base_convert — 数値の基数を任意に変換する

説明

string **base_convert** (string $number , int $frombase , int $tobase)

number を基数 tobase で表した文字列を返します。 number の基数は、 frombase で指定します。 frombase および tobase は、ともに 2 から 36 までである必要があります。 10 より大きな基数を有する数の各桁は、文字 a-z で表されます。 この場合、a は 10、b は 11、z は 35 を意味します。

警告

大きな数値で **base_convert()** を使用すると、 精度が失われてしまうことがあります。これは、内部で使用している "double" や "float" の性質によるものです。詳細な情報や制限については マニュアルの浮動小数点数 のセクションを参照ください。

パラメータ

number

変換する数値。

frombase

返還前の number の基数。

tobase

変換後の number の基数。

返り値

number を基数 tobase で表した値を返します。

例

Example#1 base_convert() の例

```
<?php
$hexadecimal = 'A37334';
echo base_convert($hexadecimal, 16, 2);
?>
```

上の例の出力は以下となります。

```
101000110111001100110100
```

参考

- intval()

---

bindec

(PHP 4, PHP 5)

bindec — 2 進数 を 10 進数に変換する

説明

number **bindec** (string $binary_string)

引数 binary_string により指定された 2 進数と等価な 10 進数を返します。

bindec() は、2 進数を integer に変換します。 サイズの問題により、必要に応じて float となることもあります。

パラメータ

binary_string

変換したい 2 進数文字列。

返り値

binary_string を 10 進に変換した値を返します。

変更履歴

バージョン	説明
Since 4.1.0	この関数は、integer 型の範囲をこえる大きな数値も変換できるようになりました。 そのような値は、float で返します。

例

Example#1 bindec() の例

```
<?php
echo bindec('110011') . "¥n";
echo bindec('000110011') . "¥n";

echo bindec('111');
?>
```

上の例の出力は以下となります。

```
51
51
7
```

参考

- decbin()
- octdec()
- hexdec()
- base_convert()

---

ceil

(PHP 4, PHP 5)

ceil — 端数の切り上げ

説明

float **ceil** (float $value)

value の次に大きい整数値を返します。

パラメータ

value

丸める値。

返り値

value の次に大きい整数値を返します。 **ceil()** の返り値は float 型と なります。これは、float 値の範囲は通常 int よりも広いためです。

例

Example#1 ceil() の例

```
<?php
echo ceil(4.3);    // 5
echo ceil(9.999);  // 10
echo ceil(-3.14);  // -3
?>
```

参考

- floor()
- round()

---

cos

(PHP 4, PHP 5)

cos — 余弦（コサイン）

説明

float **cos** (float $arg)

cos() は、arg のコサインを 返します。arg はラジアンです。

パラメータ

arg

ラジアンで表した角度。

返り値

arg のコサインを返します。

例

Example#1 cos() の例

```
<?php
echo cos(M_PI); // -1
?>
```

参考

- acos()
- sin()
- tan()
- deg2rad()

---

cosh

(PHP 4 >= 4.0.7, PHP 5)

cosh — 双曲線余弦（ハイパボリックコサイン）

説明

float **cosh** (float $arg)

arg のハイパボリックコサインを返します。 これは、(exp(arg) + exp(-arg))/2 で定義されます。

パラメータ

arg

処理する引数。

返り値

arg のハイパボリックコサインを返します。

参考

- cos()
- acosh()
- sinh()
- **cosh()**

---

decbin

(PHP 4, PHP 5)

decbin — 10 進数を 2 進数に変換する

説明

string **decbin** (*int $number*)

引数 number を 2 進数表現した文字列を返します。 変換することが出来る最大の数は 10 進数の 4294967295 であり、1 が 32 ビット並んだ 2 進数となります。

パラメータ

number

変換したい 10 進数値。

返り値

binary_string を 2 進文字列で表した値を返します。

例

Example#1 decbin() の例

```
<?php
echo decbin(12) . "¥n";
echo decbin(26);
?>
```

上の例の出力は以下となります。

```
1100
11010
```

参考

- bindec()
- decoct()
- dechex()
- base_convert()

---

dechex

(PHP 4, PHP 5)

dechex — 10 進数を 16 進数に変換する

説明

string **dechex** (*int $number*)

引数 number を 16 進数表現した文字列を返します。 変換できる最大の数字は 4294967295 であり、16 進数で表すと "ffffffff" です。

パラメータ

number

変換したい 10 進数値。

返り値

number を 16 進文字列で表した値を返します。

例

Example#1 dechex() の例

```
<?php
echo dechex(10) . "¥n";
echo dechex(47);
?>
```

上の例の出力は以下となります。

```
a
2f
```

参考

- hexdec()
- decbin()
- decoct()
- base_convert()

---

decoct

(PHP 4, PHP 5)

decoct — 10 進数を 8 進数に変換する

説明

string **decoct** (int $number)

引数 number を 8 進数表現した文字列を返します。 変換出来る最大の数字は 10 進数の 4294967295 であり、"37777777777" を返します。

パラメータ

number

変換したい 10 進数値。

返り値

number を 8 進文字列で表した値を返します。

例

Example#1 decoct() の例

```
<?php
echo decoct(15) . "¥n";
echo decoct(264);
?>
```

上の例の出力は以下となります。

```
17
410
```

参考

- octdec()
- decbin()
- dechex()
- base_convert()

---

deg2rad

(PHP 4, PHP 5)

deg2rad — 度単位の数値をラジアン単位に変換する

説明

float **deg2rad** (float $number)

この関数は、number の単位を度からラジアンに変換します。

パラメータ

number

度で表した角度。

返り値

number と同等な値をラジアンで表したものを返します。

例

Example#1 deg2rad() の例

```
<?php

echo deg2rad(45); //  0.785398163397
var_dump(deg2rad(45) === M_PI_4); //  bool(true)

?>
```

参考

- rad2deg()

---

exp

(PHP 4, PHP 5)

exp — **e** の累乗を計算する

説明

float **exp** (float $arg)

e を arg 乗した値を返します。

注意: 'e' は自然対数の底で、およそ 2.718282 です。

パラメータ

arg

処理する引数。

返り値

'e' の arg 乗を返します。

例

Example#1 exp() の例

```
<?php
echo exp(12) . "¥n";
echo exp(5.7);
?>
```

上の例の出力は以下となります。

```
1.6275E+005
298.87
```

参考

- log()
- pow()

---

expm1

(PHP 4 >= 4.0.7, PHP 5)

expm1 — 値がゼロに近い時にでも精度を保つために exp(number) - 1 を返す

説明

float **expm1** (float $arg)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

expm1() は、 'exp(number) - 1' の値を返します。 number がゼロに近く、 'exp (number) - 1' が引き算時の桁落ちのために 不正確となるような場合でも正確な値が計算できる方法を使用します。

> **注意**: この関数は Windows 環境にはまだ実装されていません。

パラメータ

arg

処理する引数。

返り値

'e' の arg 乗から 1 を引いた値を返します。

参考

- log1p()
- exp()

---

floor

(PHP 4, PHP 5)

floor — 端数の切り捨て

説明

float **floor** (float $value)

必要に応じて value を丸めることにより、 value をこえない最大の整数の値を返します。

パラメータ

number

丸める数値。

返り値

arg をこえない最大の整数の値を返します。 **floor()** の返り値は float 型のままとなります。これは、 float の範囲が int よりも広いためです。

例

Example#1 floor() の例

```
<?php
echo floor(4.3);   // 4
echo floor(9.999); // 9
echo floor(-3.14); // -4
?>
```

参考

- ceil()
- round()

---

fmod

(PHP 4 >= 4.2.0, PHP 5)

fmod — 引数で除算をした際の剰余を返す

説明

float **fmod** (float $x , float $y)

被除数（x ）を除数（y ） で割った余りを返します。余り（r）は、 整数 i を使用して x = i * y + r で定義されます。 y がゼロ以外の場合は r は x と同符号で、絶対値は y より小さくなります。

パラメータ

x

被除数。

y

除数。

返り値

x /y の剰余を返します。

例

Example#1 fmod() の使用法

```
<?php
$x = 5.7;
$y = 1.3;
$r = fmod($x, $y);
// $r は 0.5 です。なぜなら 4 * 1.3 + 0.5 = 5.7 だからです。
?>
```

---

getrandmax

(PHP 4, PHP 5)

getrandmax — 乱数の最大値を取得する

説明

int **getrandmax** (void)

rand() をコールすることにより得られる最大の値を返します。

返り値

rand() が返す乱数の最大値を返します。

参考

- rand()
- srand()
- mt_getrandmax()

---

hexdec

(PHP 4, PHP 5)

hexdec — 16 進数を 10 進数に変換する

説明

number **hexdec** (string $hex_string)

引数 hex_string により指定された 16 進数に 等価な 10 進数を返します。**hexdec()** は、16 進数を 表す文字列を 10 進数に変換します。

hexdec() は、16 進数以外の文字を一切無視します。

パラメータ

hex_string

変換したい 16 進文字列。

返り値

hex_string を 10 進で表した値を返します。

変更履歴

バージョン	説明
4.1.0 以降	この関数は、integer 型の範囲をこえる大きな数値も変換できるようになりました。 そのような値は、float で返します。

例

Example#1 hexdec() の例

```
<?php
var_dump(hexdec("See"));
var_dump(hexdec("ee"));
// 共に "int(238)" を出力

var_dump(hexdec("that")); // "int(10)" を出力
var_dump(hexdec("a0")); // "int(160)" を出力
?>
```

参考

- dechex()
- bindec()
- octdec()
- base_convert()

---

hypot

(PHP 4 >= 4.0.7, PHP 5)

hypot — 直角三角形の斜辺の長さを計算する

説明

float **hypot** (float $x , float $y)

hypot() は、直角をはさむ 2 辺の長さが x および y である 直角三角形の斜辺の長さ、すなわち原点と (x , y) との距離を返します。 これは sqrt(x*x + y*y) と等価です。

パラメータ

x

最初の辺の長さ。

y

二番目の辺の長さ。

返り値

斜辺の長さを返します。

---

is_finite

(PHP 4 >= 4.2.0, PHP 5)

is_finite — 値が有限の数値であるかどうかを判定する

説明

bool **is_finite** (float $val)

val が このプラットフォーム上で有効な有限値であるかどうかを調べます。

パラメータ

val

調べる値。

返り値

val が このプラットフォーム上の PHP の float 型で有効な範囲内の数である場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

参考

- is_infinite()
- is_nan()

---

is_infinite

(PHP 4 >= 4.2.0, PHP 5)

is_infinite — 値が無限大であるかどうかを判定する

説明

bool **is_infinite** (float $val)

val が（正または負の）無限大である場合に **TRUE** を返します。たとえば log(0) の結果、 あるいはこのプラットフォーム上で扱える float の範囲を超えた数などが あてはまります。

パラメータ

val

調べる値。

返り値

val が無限大である場合に TRUE、 そうでない場合に FALSE を返します。

参考

- is_finite()
- is_nan()

---

is_nan

(PHP 4 >= 4.2.0, PHP 5)

is_nan — 値が数値でないかどうかを判定する

説明

bool **is_nan** (float $val)

val が '非数値 (not a number)' であるかどうかを調べます。たとえば acos(1.01) の結果などがこれにあたります。

パラメータ

val

調べる値。

返り値

val が '非数値 (not a number)' の場合に TRUE、そうでない場合に FALSE を返します。

参考

- is_finite()
- is_infinite()

---

lcg_value

(PHP 4, PHP 5)

lcg_value — 複合線形合同法

説明

float **lcg_value** (void)

lcg_value() は、(0, 1)の範囲の疑似乱数を返します。 この関数は、周期が 2^31 - 85 および 2^31 - 249 の 2 つの CG を組み合わせます。この関数の周期はこれら 2 つの素数の積と等価です。

返り値

(0, 1) の範囲の疑似乱数を、浮動小数点数で返します。

参考

- rand()
- mt_rand()

---

log10

(PHP 4, PHP 5)

log10 — 底が 10 の対数

説明

float **log10** (float $arg)

底を 10 とする arg の対数を返します。

パラメータ

arg

処理する引数。

返り値

底を 10 とする arg の対数を返します。

参考

- log()

---

log1p

(PHP 4 >= 4.0.7, PHP 5)

log1p — 値がゼロに近い時にでも精度を保つ方法で計算した log(1 + number) を返す

説明

float **log1p** (float $number)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

log1p() は、 log(1 + number) の値を返します。 number がゼロに近い場合でも正確な値が 計算できる方法を使用します。 log() は、このような場合には 精度の問題で log(1) の値を返してしまいます。

注意: この関数は Windows 環境にはまだ実装されていません。

パラメータ

number

処理する引数。

返り値

log(1 + number) を返します。

参考

- expm1()
- log()
- log10()

---

log

(PHP 4, PHP 5)

log — 自然対数

説明

float **log** (float $arg [, float $base])

オプションの base パラメータを指定した場合は **log()** は $\log_{base}$ arg を返します。それ以外の場合は **log()** は arg の自然対数を返します。

パラメータ

arg

対数を計算する値。

base

オプションで指定する、底（デフォルトは 'e' で、これは自然対数となります）。

返り値

base を指定した場合はそれを底とする arg の対数、指定しない場合は自然対数を返します。

変更履歴

バージョン	説明
4.3.0 以降	オプションのパラメータ *base* が使用可能となりました。 これ以前のバージョンでも *b* を底とする *n* の対数を計算することが可能です。しかし、数学の等式 $\log_b(n) = \log(n)/\log(b)$ を使用する必要があります。ここで、log は自然対数です。

参考

- log10()

- exp()
- pow()

---

max

(PHP 4, PHP 5)

max — 最大値を返す

説明

```
mixed max ( array $values )
mixed max ( mixed $value1 , mixed $value2 [, mixed $value3... ] )
```

パラメータとして配列をひとつだけ渡した場合は、**max()** は配列の中で最も大きい数値を返します。 ふたつ以上のパラメータを指定した場合は、**max()** はそれらの中で最も大きいものを返します。

> **注意**: PHP は、数値として解釈できない string を integer と比較する場合には 0 と評価します。しかし、もしそれが最大値であった 場合、返り値はもとの文字列となります。0 と評価される 引数が複数存在した場合、**max()** はもしその中に 数値の 0 があればそれを返し、そうでなければ アルファベット順で一番大きな文字列の値が返されます。

パラメータ

values

値を含む配列。

返り値

max() は、パラメータとして渡した値の中で 数値として最も大きいものを返します。

例

Example#1 max() の使用例

```
<?php
echo max(1, 3, 5, 6, 7);  // 7
echo max(array(2, 4, 5)); // 5

echo max(0, 'hello');     // 0
echo max('hello', 0);     // hello
echo max(-1, 'hello');    // hello

// 複数の配列を渡すと、max はその要素を左から順に比較します。
// この例では 2 == 2 ですが 4 < 5 となります。
$val = max(array(2, 4, 8), array(2, 5, 7)); // array(2, 5, 7)

// 配列と配列でない値が渡された場合、常に
// 配列が最大と判定されます。
$val = max('string', array(2, 5, 7), 42);   // array(2, 5, 7)
?>
```

参考

- min()
- count()

---

min

(PHP 4, PHP 5)

min — 最小値を返す

説明

```
mixed min ( array $values )
mixed min ( mixed $value1 , mixed $value2 [, mixed $value3... ] )
```

パラメータとして配列をひとつだけ渡した場合は、 **min()** は配列の中で最も大きい数値を返します。 ふたつ以上のパラメータを指定した場合は、**min()** はそれらの中で最も小さいものを返します。

> **注意**: PHP は、数値として解釈できない string を integer と比較する場合には 0 と評価します。しかし、もしそれが最小値であった 場合、返り値はもとの文字列となります。0 と評価される 引数が複数存在した場合、**mix()** はもしその中に 文字列があればアルファベット順で一番小さな文字列の値を返し、 そうでなければ数値の 0 が返されます。

パラメータ

values

値を含む配列。

返り値

min() は、パラメータとして渡した値の中で 数値として最も小さいものを返します。

例

Example#1 min() の例

```
<?php
echo min(2, 3, 1, 6, 7);  // 1
echo min(array(2, 4, 5)); // 2

echo min(0, 'hello');     // 0
echo min('hello', 0);     // hello
echo min('hello', -1);    // -1

// 複数の配列を渡すと、mix はその要素を左から順に比較します。
// この例では 2 == 2 ですが 4 < 5 となります。
$val = min(array(2, 4, 8), array(2, 5, 1)); // array(2, 4, 8)

// 配列と配列でない値が渡された場合は常に
// 配列が最大と判定されるため、配列が返されることはありません。
$val = min('string', array(2, 5, 7), 42);   // string
?>
```

参考

- max()
- count()

---

mt_getrandmax

(PHP 4, PHP 5)

mt_getrandmax — 乱数値の最大値を表示する

説明

int **mt_getrandmax** (void)

mt_rand() のコールにより返される最大の値を返します。

返り値

mt_rand() が返す乱数の最大値を返します。

参考

- mt_rand()
- mt_srand()
- getrandmax()

---

mt_rand

(PHP 4, PHP 5)

mt_rand — よりよい乱数値を生成する

説明

int **mt_rand** ([int $min], int $max)

古い libc の多くの乱数発生器は、怪しげであるか特性が不明であったりし、 また低速でした。デフォルトでは、PHP は rand() において libc の乱数発生器を使用します。 **mt_rand()** 関数は、その代替品となるものです。 この関数は、その特性が既知の乱数生成器 » Mersenne Twister を使用し、 平均的な libc の rand()よりも 4 倍以上高速に乱数を生成します。

オプションの引数 min ,max を付けずに コールした場合、**mt_rand()** は 0 から **RAND_MAX** の間の擬似乱数値を返します。 例えば、5 から 15 まで(端点を含む)の間の乱数値を得たい場合には mt_rand(5, 15) としてください。

> **注意**: PHP 4.2.0 以降、 srand() または mt_srand() によりランダム数生成器にシードを与える必要はありません。これは、この処理が自動的に行われるためです。

パラメータ

min

オプションで指定する、返される値の最小値（デフォルトは 0）。

max

オプションで指定する、返される値の最大値（デフォルトは RAND_MAX）。

返り値

min（あるいは 0）から max（あるいは RAND_MAX、それぞれ端点を含む）までの間のランダムな整数値を返します。

変更履歴

バージョン	説明
3.0.7 以降	3.0.7 より前のバージョンでは、*max* の意味は *range* でした。これらのバージョンにおいて 同じ結果を得るために簡単な例を示すと、5 から 15 までの乱数を得たい 場合には *mt_rand (5, 11)* とする必要があります。

例

Example#1 mt_rand() の例

```
<?php
echo mt_rand() . "¥n";
echo mt_rand() . "¥n";

echo mt_rand(5, 15);
?>
```

上の例の出力は、たとえば以下のようになります。

```
1604716014
1478613278
6
```

参考

- mt_srand()
- mt_getrandmax()
- rand()

---

mt_srand

(PHP 4, PHP 5)

mt_srand — 改良型乱数生成器にシードを指定する

説明

void **mt_srand** ([int $seed])

seed により乱数生成器にシードを指定します。 seed を指定しなかった場合は、 ランダムな値を設定します。

> **注意**: PHP 4.2.0 以降、 srand() または **mt_srand()** によりランダム数生成器にシードを与える必要はありません。これは、この処理が自動的に行われるためです。

パラメータ

seed

オプションで指定するシード値。

変更履歴

バージョン	説明
4.2.0 以降	*seed* がオプションとなり、 省略した場合はデフォルトでランダムな値が設定されるようになりました。
5.2.1 以降	PHP の Mersenne Twister 実装は、Richard Wagner による新たなシード生成アルゴリズムを使用するようになりました。 これまでのバージョンのように、 同じ値のシーケンスで同じシードが生成されることはなくなりました。 この振る舞いが今後変わることはないでしょうが、この振る舞いに頼ってしまってはいけません。

例

Example#1 mt_srand() の例

```
<?php
// マイクロ秒でシードを指定します
function make_seed()
{
  list($usec, $sec) = explode(' ', microtime());
  return (float) $sec + ((float) $usec * 100000);
}
mt_srand(make_seed());
$randval = mt_rand();
?>
```

参考

- mt_rand()
- mt_getrandmax()

- srand()

---

octdec

(PHP 4, PHP 5)

octdec — 8 進数を 10 進数に変換する

説明

number **octdec** (string $octal_string)

octal_string により指定された 8 進数を 10 進数表現した数値を返します。

パラメータ

octal_string

変換したい 8 進文字列。

返り値

octal_string を 8 進で表した値を返します。

変更履歴

バージョン	説明
4.1.0 以降	この関数は、integer 型の範囲をこえる大きな数値も変換できるようになりました。 そのような値は、float で返します。

例

Example#1 octdec() の例

```
<?php
echo octdec('77') . "¥n";
echo octdec(decoct(45));
?>
```

上の例の出力は以下となります。

```
63
45
```

参考

- decoct()
- bindec()
- hexdec()
- base_convert()

---

pi

(PHP 4, PHP 5)

pi — 円周率の値を得る

説明

float **pi** (void)

円周率の近似値を返します。返される float 値の小数点以下の 桁数は、php.ini の precision ディレクティブに依存します。デフォルトは 14 です。 また、定数 **M_PI** を使用することで **pi()** とまったく同じ結果を取得することも可能です。

返り値

円周率の値を不動小数点数で返します。

例

Example#1 pi() の例

```
<?php
echo pi(); //  3.1415926535898
echo M_PI; //  3.1415926535898
?>
```

---

pow

(PHP 4, PHP 5)

pow — 指数表現

説明

number **pow** (number $base , number $exp)

base の exp 乗を返します。

警告

PHP 4.0.6 より前のバージョンでは、**pow()** は 常に float を返します。この場合、警告は発生しません。

パラメータ

base

使用する基数。

exp

指数。

返り値

base の exp 乗を 返します。可能な場合、この関数は、integer 型の値を 返します。累乗が計算できない場合は **FALSE** を返します。

変更履歴

バージョン	説明
4.0.6 以降	可能な場合は、結果を integer で返すようになりました。 以前は、結果を常に float で返していました。 そのため、値によっては間違った結果となることがありました。
4.2.0 以降	PHP 値が計算できない場合に警告を発生することはなくなり、 単に **FALSE** を返すだけとなりました。

例

Example#1 pow() の例

```
<?php

var_dump(pow(2, 8)); // int(256)
echo pow(-1, 20); // 1
echo pow(0, 0); // 1

echo pow(-1, 5.5); // エラー

?>
```

参考

- exp()
- sqrt()
- bcpow()
- gmp_pow()

---

rad2deg

(PHP 4, PHP 5)

rad2deg — ラジアン単位の数値を度単位に変換する

説明

float **rad2deg** (float $number)

この関数は、number の単位をラジアンから度に変換します。

パラメータ

number

ラジアン値。

返り値

number と同等な値を度で表したものを返します。

例

Example#1 rad2deg() の例

```
<?php

echo rad2deg(M_PI_4); // 45

?>
```

参考

- deg2rad()

---

rand

(PHP 4, PHP 5)

rand — 乱数を生成する

説明

int **rand** ([int $min], int $max)

オプションの引数 min ,max を省略してコールした場合、**rand()** は 0 と **RAND_MAX** の間の擬似乱数(整数)を返します。 例えば、5 から 15 まで（両端を含む）の乱数を得たい場合、 rand(5,15) とします。

> **注意**: (Windows のような) いくつかのプラットフォームでは、**RAND_MAX** は 32768 と小さな値となっています。 32768 より広い範囲にしたい場合、 min および max を指定することで、 RAND_MAX より大きな範囲の乱数を生成することができます。 もしくは、mt_rand() をかわりに使用してみてください。
>
> **注意**: PHP 4.2.0 以降、 srand() または mt_srand() によりランダム数生成器にシードを与える必要はありません。これは、この処理が自動的に行われるためです。

パラメータ

min

返す値の最小値（デフォルトは 0)。

max

返す値の最大値（デフォルトは RAND_MAX)。

返り値

min（あるいは 0）から max（あるいは RAND_MAX、それぞれ端点を含む）までの間の疑似乱数値を返します。

変更履歴

バージョン	説明
3.0.7 以降	3.0.7 より前のバージョンでは、max の意味は range でした。これらのバージョンにおいて 同じ結果を得るために簡単な例を示すと、 5 から 15 までの乱数を得たい場合には rand (5, 11) とする必要があります。

例

Example#1 rand() の例

```
<?php
echo rand() . "¥n";
echo rand() . "¥n";

echo rand(5, 15);
?>
```

上の例の出力は、たとえば以下のようになります。

```
7771
22264
11
```

参考

- srand()
- getrandmax()
- mt_rand()

---

round

(PHP 4, PHP 5)

round — 浮動小数点数を丸める

説明

float **round** (float $val [, int $precision])

val を、指定した precision (小数点以下の桁数)に丸めた値を 返します。precision を負またはゼロ(デフォルト) とすることも可能です。

> **注意**: PHP は、デフォルトでは "12,300.2" のような 文字列を正しく処理しません。文字列からの変換 を参照ください。
>
> **注意**: パラメータ precision は、PHP 4 以降でのみ 利用可能です。 (訳注：内部的な 2 進数表現と 10 進数表現の差により生じる丸め誤差の影響により 必ずしも小数点以下を四捨五入した結果を返さないことに注意してください。)

パラメータ

val

丸める値。

precision

オプションで指定する、丸める桁数。デフォルトは 0 です

返り値

The rounded value

例

Example#1 round() examples

```
<?php
echo round(3.4);          // 3
echo round(3.5);          // 4
echo round(3.6);          // 4
echo round(3.6, 0);       // 4
echo round(1.95583, 2);   // 1.96
echo round(1241757, -3); // 1242000
echo round(5.045, 2);     // 5.05
echo round(5.055, 2);     // 5.06
?>
```

参考

- ceil()
- floor()
- number_format()

---

sin

(PHP 4, PHP 5)

sin — 正弦（サイン）

説明

float **sin** (float $arg)

sin() は、arg のサインを 返します。arg はラジアンです。

パラメータ

arg

ラジアンで表した値。

返り値

arg のサインを返します。

例

Example#1 sin() の例

```
<?php

// 結果の精度は、指定によります。
echo sin(deg2rad(60));  //   0.866025403  ...
echo sin(60);           //  -0.304810621  ...

?>
```

参考

- asin()

- sinh()
- cos()
- tan()
- deg2rad()

---

sinh

(PHP 4 >= 4.0.7, PHP 5)

sinh — 双曲線正弦（ハイパボリックサイン）

説明

float **sinh** (float $arg)

arg のハイパボリックサインを返します。 これは、(exp(arg) - exp(-arg))/2 で定義されます。

パラメータ

arg

処理する引数。

返り値

arg のハイパボリックサインを返します。

参考

- sin()
- asinh()
- cosh()
- tanh()

---

sqrt

(PHP 4, PHP 5)

sqrt — 平方根

説明

float **sqrt** (float $arg)

arg の平方根を返します。

パラメータ

arg

処理する引数。

返り値

arg の平方根を返します。

例

Example#1 sqrt() の例

```
<?php
// 小数点以下の精度は、precision ディレクティブの設定に依存します
echo sqrt(9); // 3
echo sqrt(10); // 3.16227766 ...
?>
```

参考

- pow()

---

srand

(PHP 4, PHP 5)

srand — 乱数ジェネレータを初期化する

説明

void **srand** ([int $seed])

シード seed で乱数ジェネレータを初期化します。 seed を省略した場合はランダムな値が設定されます。

> **注意**: PHP 4.2.0 以降、 **srand()** または mt_srand() によりランダム数生成器にシードを与える必要はありません。これは、この処理が自動的に行われるためです。

パラメータ

seed

オプションで指定するシード値。

変更履歴

バージョン	説明
4.2.0 以降	seed はオプションになりました。 省略した場合はランダムな値がデフォルトで設定されます。

例

Example#1 srand() の例

```
<?php
// マイクロ秒でシードを設定します
function make_seed()
{
   list($usec, $sec) = explode(' ', microtime());
   return (float) $sec + ((float) $usec * 100000);
}
srand(make_seed());
$randval = rand();
?>
```

参考

- rand()
- getrandmax()
- mt_srand()

---

tan

(PHP 4, PHP 5)

tan — 正接（タンジェント）

説明

float **tan** (float $arg)

tan() は、arg のタンジェントを 返します。arg はラジアンです。

パラメータ

arg

処理する引数をラジアンで表したもの。

返り値

arg のタンジェントを返します。

例

Example#1 tan() の例

```
<?php

echo tan(M_PI_4); // 1

?>
```

参考

- atan()
- atan2()
- sin()
- cos()
- tanh()
- deg2rad()

---

tanh

(PHP 4 >= 4.0.7, PHP 5)

tanh — 双曲線正接（ハイパボリックタンジェント）

説明

float **tanh** (float $arg)

arg のハイパボリックタンジェントを返します。 これは sinh(arg)/cosh(arg) で定義されます。

パラメータ

arg

処理する引数。

返り値

arg のハイパボリックタンジェントを返します。

参考

- tan()
- atanh()
- sinh()
- cosh()

目次

- pow — 指数表現
- rad2deg — ラジアン単位の数値を度単位に変換する
- rand — 乱数を生成する
- round — 浮動小数点数を丸める
- sin — 正弦（サイン）
- sinh — 双曲線正弦（ハイパボリックサイン）
- sqrt — 平方根
- srand — 乱数ジェネレータを初期化する
- tan — 正接（タンジェント）
- tanh — 双曲線正接（ハイパボリックタンジェント）

MaxDB PHP 拡張モジュール

導入

MaxDB PHP 拡張モジュールにより、MaxDB 7.5.0 以降にアクセスできるようになります。 MaxDB データベースサーバについての詳細な情報は » http://www.mysql.com/products/maxdb/ で得られます。

MaxDB PHP 拡張モジュールは、MySQL の mysqli 拡張モジュールと互換性があります。 これらの間にはほんの少しの違いしかなく、その違いは MaxDB と MySQL の違いに基づくものです。

mysqli との主な相違点は、以下のようになります。

- maxdb_character_set_name() - ascii あるいは unicode しか返しません。
- maxdb_get_client_info() - 異なるバージョン文字列を返します。
- maxdb_get_client_version() - 異なるバージョン文字列を返します。
- maxdb_get_host_info() - localhost あるいはホスト名を返します。
- maxdb_get_server_info() - 異なるバージョン文字列を返します。
- maxdb_get_server_version() - 異なるバージョン文字列を返します。
- maxdb_kill() - 単にセッションを切断するだけです。
- maxdb_multi_query() - 複数の SQL 文を扱うことはできません。
- maxdb_next_result() - 常に false を返します。
- maxdb_options() - サポートするオプションが異なります。
- maxdb_report() - サポートする繰り返しモードが異なります。
- maxdb_stat() - 異なるシステム状態文字列を返します。
- maxdb_stmt_store_result() - MaxDB では不要です。
- maxdb_store_result() - MaxDB では不要です。

MaxDB についてのドキュメントは » http://dev.mysql.com/doc/maxdb/ にあります。

要件

これらの関数を使用するには、MaxDB のサポートを有効にして PHP をコンパイルする必要があります。さらに、MaxDB サーバにアクセスするために MaxDB SQLDBC ランタイムライブラリが必要です。

MaxDB SQLDBC についてのドキュメントは » http://dev.mysql.com/doc/maxdb/ にあります。

MaxDB SQLDBC パッケージは » http://dev.mysql.com/downloads/maxdb/clients.html からダウンロードします。

インストール手順

設定オプション --with-maxdb[=DIR] を使用すると、PHP から MaxDB データベースへのアクセス機能が有効となります。 [DIR] には、MaxDB SQLDBC パッケージのインストールされているディレクトリを指定します。

Windows ユーザは、php.ini の中で php_maxdb.dll を有効にする必要があります。

実行時設定

php.ini の設定により動作が変化します。

MaxDB 設定オプション

名前	デフォルト	変更の可否	変更履歴
maxdb.default_host	NULL	PHP_INI_ALL	
maxdb.default_db	NULL	PHP_INI_ALL	
maxdb.default_user	NULL	PHP_INI_ALL	
maxdb.default_pw	NULL	PHP_INI_ALL	
maxdb.long_readlen	"200"	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

maxdb.default_host *string*

データベースサーバへの接続時、ホスト名が指定されていない場合に 使用するデフォルトのホスト。

maxdb.default_db *string*

データベースが指定されていない場合に、 接続時に使用するデフォルトのサーバデータベース。

maxdb.default_user *string*

データベースサーバへの接続時、ユーザ名が指定されていない場合に 使用するデフォルトのユーザ名。

maxdb.default_pw *string*

データベースサーバへの接続時、パスワードが指定されていない場合に 使用するデフォルトのパスワード。

maxdb.long_readlen *integer*

MaxDB データベースサーバからロングデータを取得した場合に、 クライアントに転送される最大バイト数のデフォルト値。

定義済みクラス

maxdb

PHP と MaxDB データベースとの間の接続を表します。

コンストラクタ

- *maxdb - 新しい maxdb オブジェクトを作成する*

メソッド

- *autocommit - データベースの変更内容の自動コミット機能を有効あるいは無効にする*
- *change_user - 指定したデータベース接続のユーザ名を変更する*
- *character_set_name - データベース接続のデフォルト文字セットを返す*
- *close - 事前にオープンされた接続を閉じる*
- *commit - 現在のトランザクションをコミットする*
- *connect - MaxDB データベースサーバへの新しい接続をオープンする*
- *debug - デバッグ操作を実行する*
- *dump_debug_info - デバッグ情報を出力する*
- *get_client_info - クライアントのバージョンを返す*
- *get_host_info - 使用している接続の型を返す*
- *get_server_info - MaxDB サーバのバージョンを返す*
- *get_server_version - MaxDB サーバのバージョンを返す*
- *init - maxdb オブジェクトを初期化する*
- *info - 直近で実行されたクエリの情報を取得する*
- *kill - MaxDB スレッドの終了をサーバに要求する*
- *multi_query - 複数のクエリを実行する*
- *more_results - 現在実行している複数クエリに次の結果があるかどうかを調べる*
- *next_result - 現在実行している複数クエリの次の結果を読み込む*
- *options - オプションを設定する*
- *ping - サーバとの接続を確認し、接続されていない場合には再接続する*
- *prepare - SQL クエリを準備する*
- *query - クエリを実行する*
- *real_connect - MaxDB データベースサーバとの接続をオープンする*
- *escape_string - 接続の現在の文字セットを考慮したうえで、SQL 文で使用する文字列の特殊文字をエスケープする*
- *rollback - 現在のトランザクションをロールバックする*
- *select_db - デフォルトのデータベースを選択する*
- *ssl_set - ssl パラメータを設定する*
- *stat - 現在のシステム状態を取得する*
- *stmt_init- maxdb_stmt_prepare で使用するステートメントを初期化する*
- *store_result - 最後に実行したクエリの結果を転送する*
- *use_result - 最後に実行したクエリのバッファ化されていない結果を転送する*
- *thread-safe - スレッドセーフかどうかを返す*

プロパティ

- affected_rows - 直前の MaxDB 操作で変更された行数を取得する
- client_info - MaxDB クライアントのバージョンを文字列で返す
- client_version - MaxDB クライアントのバージョンを整数で返す
- errno - 直近の関数コールのエラーコードを返す
- error - 直近の関数コールのエラー文字列を返す
- field_count - 直近のクエリのカラム数を返す
- host_info - 使用している接続の型を表す文字列を返す
- info - 直近に実行されたクエリについての情報を取得する
- insert_id - 直近のクエリで使用した自動生成 ID を返す
- protocol_version - 使用している MaxDB プロトコルのバージョンを返す
- sqlstate - 直近のエラーについての SQLSTATE エラーコードを含む文字列を返す
- thread_id - 現在の接続のスレッド ID を返す
- warning_count - 直前の SQL 文の実行中に発生した警告の数を返す

maxdb_stmt

プリペアドステートメントを表します。

メソッド

- bind_param - プリペアドステートメントに変数をバインドする
- bind_result - 結果を保存するために、プリペアドステートメントに変数をバインドする
- close - プリペアドステートメントを閉じる
- data-seek - ステートメントの結果セットの任意の行に移動する
- execute - プリペアドステートメントを実行する
- fetch - プリペアドステートメントから結果を取得してバインド変数に保存する
- free_result - 指定したステートメントハンドルの結果メモリを開放する
- result_metadata - プリペアドステートメントからメタデータ情報の結果セットを取得する
- prepare - SQL クエリを準備する
- send_long_data - データをチャンクに送る
- close_long_data - long データの送信を終了する
- reset - プリペアドステートメントをリセットする
- store_result - プリペアドステートメントから結果セット全体をバッファに保存する

プロパティ

- affected_rows - 直近のステートメントの実行で変更された行数を返す
- errno - 直近のステートメント関数のエラーコードを返す
- errno - 直近のステートメントｋな数のエラーメッセージを返す
- param_count - 指定したプリペアドステートメントのパラメータの数を返す
- sqlstate - 直近のステートメント関数の SQLSTATE エラーコードを含む文字列を返す

maxdb_result

データベースに対するクエリによって取得した結果セットを表します。

メソッド

- close - 結果セットを閉じる
- data_seek - 内部結果ポインタを移動する
- fetch_field - 結果セットからカラム情報を取得する
- fetch_fields - 結果セットの全てのカラムの情報を取得する
- fetch_field_direct - 指定したカラムの情報を取得する
- fetch_array - 連想配列、数値添字配列あるいはその両方で結果の行を取得する
- fetch_assoc - 結果の行を連想配列で取得する
- fetch_object - 結果の行をオブジェクトとして取得する

- *fetch_row - 結果の行を数値添字の配列で取得する*
- *close - 結果のメモリを開放する*
- *field_seek - 指定したフィールドオフセットに結果ポインタを移動する*

プロパティ

- *current_field - 現在のフィールドポインタのオフセットを返す*
- *field_count - 結果セットのフィールド数を返す*
- *lengths - カラムの長さの配列を返す*
- *num_rows - 結果セットの行数を返す*

リソース型

この拡張モジュールではリソースを定義しています。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

以下の定数が、maxdb_options() で使用するために定義されています。これらの定数についての詳細な情報は » http://dev.mysql.com/doc/maxdb/ を参照ください。

MaxDB PHP クライアント定数

定数	説明
MAXDB_COMPNAME	SQLDBC ランタイム環境を初期化するために使用するコンポーネント名。
MAXDB_APPLICATION	データベースに接続しているアプリケーション。
MAXDB_APPVERSION	アプリケーションのバージョン。
MAXDB_SQLMODE	SQL モード。
MAXDB_UNICODE	unicode (UCS2) クライアントによる接続の場合に TRUE、そうでない場合に FALSE。
MAXDB_TIMEOUT	データベースへの接続がシステムによって閉じられるまでの 最大無活動時間。
MAXDB_ISOLATIONLEVEL	共有ロックおよび排他ロックを暗黙的に要求/開放するかどうか、 またどのように行うかを指定します。
MAXDB_PACKETCOUNT	接続に使用するリクエストパケットの数。
MAXDB_STATEMENTCACHESIZE	接続内での再利用のためにキャッシュされるプリペアドステートメントの数。
MAXDB_CURSORPREFIX	自動的に命名される結果テーブルに使用するプレフィックス。

maxdb_fetch_array() 関数は、 結果の配列の型を指定するために定数を使用します。 以下の定数が定義されています。

MaxDB フェッチ定数

定数	説明
MAXDB_ASSOC	フィールド名をインデックスとする配列で、カラムを返します。
MAXDB_ASSOC_UPPER	大文字に変換したフィールド名をインデックスとする配列で、 カラムを返します。
MAXDB_ASSOC_LOWER	小文字に変換したフィールド名をインデックスとする配列で、 カラムを返します。
MAXDB_BOTH	数値インデックスおよびフィールド名インデックスの両方を含む配列で、 カラムを返します。
MAXDB_NUM	フィールドの数値インデックスを持つ配列で、カラムを返します。 インデックスは 0 から始まり、これが結果の最初のフィールドを表します。

例

MaxDB PHP マニュアルの全ての例は、MaxDB から得られるデモデータベース HOTELDB を使用しています。このデータベースについての情報は » http://dev.mysql.com/doc/maxdb/en/98/11b83fa6b33c17e10000000a114084/frameset.htm で得られます。

MaxDB PHP マニュアルのデータベースを使用するには、チュートリアルのデータを データベースに読み込む必要があります。その後、php.ini の中の maxdb.default_db に、チュートリアルデータを含むデータベースを設定します。

この単純な例では、MaxDB データベースへの接続・クエリの実行・ 結果の行の表示・接続の切断の方法を示します。

Example#1 MaxDB 拡張モジュールの概要

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

/* SQL クエリを実行します */
$query = "SELECT * FROM hotel.city";
$result = maxdb_query($link, $query) or die("クエリに失敗しました:  " . maxdb_error());
```

```
/* 結果を HTML で表示します */
echo "<table>¥n";
while ($line = maxdb_fetch_array($result, MAXDB_ASSOC)) {
    echo "   <tr>¥n";
    foreach ($line as $col_value) {
        echo "      <td>$col_value</td>¥n";
    }
    echo "   </tr>¥n";
}
echo "</table>¥n";

/* 結果セットを開放します */
maxdb_free_result($result);

/* 接続を閉じます */
maxdb_close($link);
?>
```

以下の例では、SELECT INTO 文に変数をバインドする方法を示します。

Example#2 SELECT INTO 文の使用例

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* SQL クエリを実行します */
$stmt = maxdb_prepare ($link, "SELECT percentage INTO ? FROM hotel.countrylanguage where language = ?");
if (!$stmt) {
  printf ("準備に失敗しました: %s\n", maxdb_error($link));
}

$name = "Mbundu";

maxdb_stmt_bind_param($stmt, 'ds', $percentage, $name);
maxdb_stmt_execute($stmt);

printf ("%f¥n", $percentage);

maxdb_stmt_close ($stmt);
?>
```

以下の例では、MaxDB のプロシージャを使用する方法を示します。

Example#3 データベースプロシージャの使用

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP DBPROC test_proc");
maxdb_report (MAXDB_REPORT_ERROR);

$query = "create dbproc test_proc (INOUT e_text char(72)) AS select * from SYSDBA.DUAL; fetch into :e_text;";

maxdb_query($link, $query);

/* SQL クエリを実行します */
$stmt = maxdb_prepare ($link, "CALL test_proc (?)");
if (!$stmt) {
  printf ("準備に失敗しました: %s\n", maxdb_error($link));
}

maxdb_stmt_bind_param($stmt, 's', $result);
maxdb_stmt_execute($stmt);

printf ("%s¥n", $result);

maxdb_stmt_close ($stmt);
?>
```

---

maxdb_affected_rows

maxdb->affected_rows

(PECL maxdb:1.0-7.6.00.38)

maxdb->affected_rows — 直前の MaxDB の操作で変更された行数を取得する

説明

手続き型

int **maxdb_affected_rows** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
int$affected_rows;

maxdb_affected_rows() は、与えられた link パラメータに関連した直近の INSERT、UPDATE あるいは DELETE クエリによって変更された行の数を返します。 この数が決定できない場合には、-1 を返します。

> **注意**: SELECT 文の場合には、**maxdb_affected_rows()** は maxdb_num_rows() と同じように動作します。

maxdb_affected_rows() 関数は、 テーブルを変更するようなクエリについてのみ動作します。 SELECT クエリの返す行数を調べるには、代わりに maxdb_num_rows() を使用してください。

返り値

ゼロより大きい整数の場合は、変更された行数あるいは取得された行数を表します。 ゼロの場合は、UPDATE ステートメントで行が更新されなかった、 WHERE 句に一致する行がなかった、あるいはクエリが実行されなかったなどを表します。 -1 の場合は、変更された行数が取得できなかったことを表します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_report (MAXDB_REPORT_OFF);
$maxdb->query("DROP TABLE mycustomer");
maxdb_report (MAXDB_REPORT_ERROR);

/* 行を挿入します */
$maxdb->query("CREATE TABLE mycustomer AS SELECT * from hotel.customer");
printf("Affected rows (INSERT): %d¥n", $maxdb->affected_rows);

$maxdb->query("ALTER TABLE mycustomer ADD Status int default 0");

/* 行を更新します */
$maxdb->query("UPDATE mycustomer SET Status=1 WHERE cno > 50");
printf("Affected rows (UPDATE): %d¥n", $maxdb->affected_rows);

/* 行を削除します */
$maxdb->query("DELETE FROM mycustomer WHERE cno < 50");
printf("Affected rows (DELETE): %d¥n", $maxdb->affected_rows);

/* すべての行を選択します */
$result = $maxdb->query("SELECT title FROM mycustomer");
printf("Affected rows (SELECT): %d¥n", $maxdb->affected_rows);

$result->close();

/* テーブルを削除します */
$maxdb->query("DROP TABLE mycustomer");

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

if (!$link) {
   printf("localhost 接続できません。エラー: %s\n", maxdb_connect_error());
    exit();
}

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE mycustomer");
maxdb_report (MAXDB_REPORT_ERROR);

/* 行を挿入します */
maxdb_query($link, "CREATE TABLE mycustomer AS SELECT * from hotel.customer");
printf("Affected rows (INSERT): %d¥n", maxdb_affected_rows($link));

maxdb_query($link, "ALTER TABLE mycustomer ADD Status int default 0");

/* 行を更新します */
maxdb_query($link, "UPDATE mycustomer SET Status=1 WHERE cno > 50");
printf("Affected rows (UPDATE): %d¥n", maxdb_affected_rows($link));

/* 行を削除します */
maxdb_query($link, "DELETE FROM mycustomer WHERE cno < 50");
printf("Affected rows (DELETE): %d¥n", maxdb_affected_rows($link));

/* すべての行を選択します */
$result = maxdb_query($link, "SELECT title FROM mycustomer");
printf("Affected rows (SELECT): %d¥n", maxdb_affected_rows($link));

maxdb_free_result($result);

/* テーブルを削除します */
maxdb_query($link, "DROP TABLE mycustomer");
```

```
/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Affected rows (INSERT): 15
Affected rows (UPDATE): 15
Affected rows (DELETE): 0
Affected rows (SELECT): 15
```

参考

- maxdb_num_rows()
- maxdb_info()

---

maxdb_autocommit

maxdb->auto_commit

(No version information available, might be only in CVS)

maxdb->auto_commit — データベースの変更内容の自動コミット機能を有効あるいは無効にする

説明

手続き型

bool **maxdb_autocommit** (resource $link , bool $mode)

オブジェクト指向型（メソッド）

maxdb
bool **auto_commit** (bool $mode)

maxdb_autocommit() は、link リソースが表すデータベース接続上のクエリについて、 自動コミットモードを有効または無効にするために使用されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 自動コミットを有効にします */
$maxdb->autocommit(TRUE);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

if (!$link) {
   printf("localhost に接続できません。エラー: %s\n", maxdb_connect_error());
    exit();
}

/* 自動コミットを有効にします */
maxdb_autocommit($link, TRUE);

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例は、なにも出力しません。

参考

- maxdb_commit()
- maxdb_rollback()

---

maxdb_bind_param

(PECL maxdb:1.0)

maxdb_bind_param — maxdb_stmt_bind_param() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_bind_param()

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_bind_result

(PECL maxdb:1.0)

maxdb_bind_result — maxdb_stmt_bind_result() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_bind_result().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_change_user

maxdb->change_user

(PECL maxdb:1.0-7.6.00.38)

maxdb->change_user — 指定したデータベース接続のユーザを変更する

説明

手続き型

bool **maxdb_change_user** (resource $link , string $user , string $password , string $database)

オブジェクト指向型（メソッド）

maxdb
bool **change_user** (string $user , string $password , string $database)

maxdb_change_user() は、パラメータ link で指定したデータベース接続のユーザを変更し、現在のデータベースを database で指定したものに変更するために使用します。

ユーザの変更を行うには、username および password に有効な値が指定されていること、 そして指定したデータベースに対する適切なアクセス権がユーザに与えられていることが必要です。 何らかの理由で認証に失敗すると、現在のユーザ認証のままの状態となります。

> **注意**: このコマンドを使用すると、その操作が正常に終了したか否かにかかわらず、 新しくデータベースに接続したのと同じように扱われます。つまり、 現在アクティブなトランザクションはすべてロールバックされ、 一時テーブルはすべて閉じられ、テーブルに対するロックもすべて解除されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php

/* test データベースに接続します */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if ($result = $maxdb->query("SELECT * FROM dual")) {
   $row = $result->fetch_row();
   printf("結果: %s\n", $row[0]);
   $result->free();
}

/* すべてをリセットし、新しいデータベースを選択します */
if (!$maxdb->change_user("DBADMIN", "SECRET", "DEMODB")) {
  printf("データベースが稼動していません\n");
} else {
```

```
  printf("データベースが稼動中です\n");
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if ($result = maxdb_query($link, "SELECT * FROM dual")) {
   $row = maxdb_fetch_row($result);
   printf("結果: %s\n", $row[0]);
   maxdb_free_result($result);
}

/* すべてをリセットし、新しいデータベースを選択します */
if (!maxdb_change_user($link, "DBADMIN", "SECRET", "DEMODB")) {
  printf("データベースが稼動していません\n");
} else {
  printf("データベースが稼動中です\n");
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
結果: a
データベースが稼動中です
```

参考

- maxdb_connect()
- maxdb_select_db()

---

maxdb_character_set_name

maxdb->character_set_name

(PECL maxdb:1.0-7.6.00.38)

maxdb->character_set_name — データベース接続のデフォルトの文字セットを返す

説明

手続き型

string **maxdb_character_set_name** (resource $link)

オブジェクト指向型（メソッド）

maxdb
string **character_set_name** (void)

link パラメータで指定したデータベース接続の、 現在の文字セットを返します。

返り値

現在の接続のデフォルトの文字セットを、ascii あるいは unicode で返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 現在の文字セットを表示します */
$charset = $maxdb->character_set_name();
printf ("現在の文字セットは %s です\n", $charset);
```

```
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 現在の文字セットを表示します */
$charset = maxdb_character_set_name($link);
printf ("現在の文字セットは %s です\n", $charset);

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
現在の文字セットは ascii です
```

参考

- maxdb_client_encoding()
- maxdb_real_escape_string()

---

maxdb_client_encoding

(PECL maxdb:1.0)

maxdb_client_encoding — maxdb_character_set_name() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_character_set_name().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_close_long_data

maxdb->close_long_data

(PECL maxdb:1.0-7.6.00.38)

maxdb->close_long_data — maxdb_stmt_close_long_data() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_close_long_data().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_close

maxdb->close

(PECL maxdb:1.0-7.6.00.38)

maxdb->close — 事前にオープンされたデータベース接続を閉じる

説明

手続き型

bool **maxdb_close** (resource $link)

オブジェクト指向型（メソッド）

maxdb
bool **close** (void)

maxdb_close() 関数は、link パラメータで指定したデータベース接続（事前にオープンされたもの）を閉じます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- maxdb_connect()
- maxdb_init()
- maxdb_real_connect()

---

maxdb_commit

maxdb->commit

(PECL maxdb:1.0-7.6.00.38)

maxdb->commit — 現在のトランザクションをコミットする

説明

手続き型

bool **maxdb_commit** (resource $link)

オブジェクト指向型（メソッド）

maxdb
bool **commit** (void)

link パラメータで指定したデータベース接続について、 現在のトランザクションをコミットします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 自動コミットを無効にします */
$maxdb->autocommit(FALSE);

maxdb_report (MAXDB_REPORT_OFF);
$maxdb->query("DROP TABLE mycustomer");
maxdb_report (MAXDB_REPORT_ERROR);

$maxdb->query("CREATE TABLE mycustomer LIKE hotel.customer");

/* 何かの値を挿入します */
$maxdb->query("INSERT INTO mycustomer VALUES (3000,'Mrs','Jenny','Porter','10580','1340 N.Ash Street, #3')");
$maxdb->query("INSERT INTO mycustomer VALUES (3100,'Mr','Peter','Brown','48226','1001 34th Str., APT.3')");

/* トランザクションをコミットします */
$maxdb->commit();

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 自動コミットを無効にします */
maxdb_autocommit($link, FALSE);

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE mycustomer");
maxdb_report (MAXDB_REPORT_ERROR);

maxdb_query($link, "CREATE TABLE mycustomer LIKE hotel.customer");

/* 何かの値を挿入します */
maxdb_query($link, "INSERT INTO mycustomer VALUES (3000,'Mrs','Jenny','Porter','10580','1340 N.Ash Street, #3')");
```

```
maxdb_query($link, "INSERT INTO mycustomer VALUES (3100,'Mr','Peter','Brown','48226','1001 34th Str., APT.3')");

/* トランザクションをコミットします */
maxdb_commit($link);

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例は、なにも出力しません。

参考

- maxdb_autocommit()
- maxdb_rollback()

---

maxdb_connect_errno

(PECL maxdb:1.0-7.6.00.38)

maxdb_connect_errno — 直近の接続コールのエラーコードを返す

説明

int **maxdb_connect_errno** (void)

maxdb_connect_errno() 関数は、 直近の maxdb_connect() のコールのエラーコードを返します。 エラーが発生しなかった場合は、この関数はゼロを返します。

返り値

直近の maxdb_connect() のコールのエラーコードを返します。 ゼロの場合は、エラーが発生しなかったことを意味します。

例

Example#1 maxdb_connect_errno の例

```
<?php
$link = maxdb_connect("localhost", "XXXXXXXX", "YYYYYYYYY");

if (!$link) {
   printf("localhost に接続できません。エラーコード: %d\n", maxdb_connect_errno());
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
PHP Warning:  maxdb_connect(): -4008 POS(1) Unknown user name/password combination [08004] <...>
localhost に接続できません。エラーコード: -4008
```

参考

- maxdb_connect()
- maxdb_connect_error()
- maxdb_errno()
- maxdb_error()
- maxdb_sqlstate()

---

maxdb_connect_error

(PECL maxdb:1.0-7.6.00.38)

maxdb_connect_error — 直近の接続エラーについての説明を文字列で返す

説明

string **maxdb_connect_error** (void)

maxdb_connect_error() 関数は、対応する maxdb_connect_errno() 関数とほぼ同じですが、 直近の maxdb_connect() call で発生したエラーについて、 エラーコードの代わりに文字列を返します。エラーが発生しなかった場合は、 この関数は空の文字列を返します。

返り値

エラーについての文字列を返します。 エラーが発生しなかった場合は空の文字列を返します。

例

Example#1 maxdb_connect_error の例

```
<?php
```

```
$link = maxdb_connect("localhost", "nonexisting_user", "");

if (!$link) {
   printf("localhost に接続できません。エラー: %s\n", maxdb_connect_error());
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
PHP Warning:  maxdb_connect(): -4008 POS(1) Unknown user name/password combination <...>
localhost に接続できません。エラー: POS(1) Unknown user name/password combination
```

参考

- maxdb_connect()
- maxdb_connect_errno()
- maxdb_errno()
- maxdb_error()
- maxdb_sqlstate()

---

maxdb_connect

maxdb()

(PECL maxdb:7.5.00.24-7.6.00.38)

maxdb() — MaxDB サーバへの新しい接続をオープンする

説明

手続き型

```
resource maxdb_connect ([ string $host [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket ]]]]] ])
```

オブジェクト指向型（コンストラクタ）

```
maxdb
__construct ([ string $host [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket ]]]]] ])
```

maxdb_connect() 関数は、host で稼動している MaxDB サーバへの接続をオープンしようと試みます。 host にはホスト名あるいは IP アドレスが指定可能です。 "localhost" を渡すと、ローカルホストが使用されます。接続に成功すると、 **maxdb_connect()** はデータベースへの接続を表すリソースを返します。 失敗した場合は **FALSE** を返します。

username および password で、MaxDB サーバに接続する際のユーザ名およびパスワードを指定します。 パスワードが指定されなかった場合（**NULL** が渡された場合）は、php.ini の maxdb.default_pw を使用して MaxDB サーバへの接続を試みます。

dbname では、 クエリを実行する際のデフォルトのデータベースを指定します。 指定されなかった場合は、php.ini の maxdb.default_db エントリの内容が使用されます。

port および socket は、MaxDB サーバでは無視されます。

返り値

MaxDB サーバへの接続を表す文字列、あるいは接続に失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

printf("ホスト情報: %s\n", $maxdb->host_info);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
```

```
}

printf("ホスト情報:  %s\n", maxdb_get_host_info($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
ホスト情報:  localhost
```

---

maxdb_data_seek

result->data_seek

(No version information available, might be only in CVS)

result->data_seek — 結果ポインタを、結果の任意の行に移動する

説明

手続き型

bool **maxdb_data_seek** (resource $result , int $offset)

オブジェクト指向型（メソッド）

result
bool **data_seek** (int $offset)

maxdb_data_seek() 関数は、result が表す結果セットの offset で指定した任意の位置に、 結果ポインタを移動します。offset は、 ゼロから全行数マイナス 1 の間 (0..maxdb_num_rows() - 1) でなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER BY name";
if ($result = $maxdb->query( $query)) {

   /* 行番号 10 に移動します */
   $result->data_seek(10);

   /* 行を取得します */
   $row = $result->fetch_row();

   printf ("City: %s   State: %s¥n", $row[0], $row[1]);

   /* 結果セットを開放します */
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER BY name";

if ($result = maxdb_query($link, $query)) {

   /* 行番号 10 に移動します */
   maxdb_data_seek($result, 10);

   /* 行を取得します */
```

```
    $row = maxdb_fetch_row($result);

    printf ("City: %s   State: %s¥n", $row[0], $row[1]);

   /* 結果セットを開放します */
    maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
City: Irvine   State: CA
```

参考

- maxdb_store_result()
- maxdb_fetch_row()
- maxdb_num_rows()

---

maxdb_debug

(PECL maxdb:1.0-7.6.00.38)

maxdb_debug — デバッグ操作を行う

説明

void **maxdb_debug** (string $debug)

maxdb_debug() は、SQLDBC 通信をトレースするために使用されます。以下の文字列が、 **maxdb_debug()** に対するパラメータとして使用可能です。

- TRACE SHORT ON|OFF - メソッドコールのトレースを有効/無効にします。
- TRACE LONG ON|OFF - メソッドの引数および詳細なデバッグトレースを有効/無効にします。
- TRACE PACKET ON|OFF|<size> - パケットのトレースを有効/無効にし、 トレースされるオブジェクトのバイト数の制限を指定します。 サイズが指定されなかった場合は 1000 バイトとなります。
- TRACE SQL ON|OFF - 高レベルの API トレースを有効/無効にします。
- TRACE TIMESTAMP ON|OFF - トレースされた各行へのプレフィックスとして、 タイムスタンプの付加を有効/無効にします。
- TRACE SIZE <size> - トレースファイルのサイズを最大 <size> バイトに制限します。少なくとも 8192 バイトが必要です。

返り値

maxdb_debug() は、何も値を返しません。

例

Example#1 手続き型

```
<?php

maxdb_debug("trace packet 200");

$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
    printf("接続に失敗しました: %s\n", maxdb_connect_error());
     exit();
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例は、何も出力しません。

---

maxdb_disable_reads_from_master

maxdb->disable_reads_from_master

(PECL maxdb:1.0-7.6.00.38)

maxdb->disable_reads_from_master — マスタからの読み込みを無効にする

説明

手続き型

bool ***maxdb_disable_reads_from_master*** *(resource $link)*

オブジェクト指向型（メソッド）

maxdb
void ***disable_reads_from_master*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_disable_rpl_parse

(PECL maxdb:1.0-7.6.00.38)

maxdb_disable_rpl_parse — RPL のパースを無効にする

説明

bool ***maxdb_disable_rpl_parse*** *(resource $link)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_dump_debug_info

(PECL maxdb:1.0-7.6.00.38)

maxdb_dump_debug_info — デバッグ情報をログに出力する

説明

bool ***maxdb_dump_debug_info*** *(resource $link)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_embedded_connect

(PECL maxdb:1.0-7.6.00.38)

maxdb_embedded_connect — 組み込み MaxDB サーバへの接続をオープンする

説明

resource ***maxdb_embedded_connect*** *([string $dbname])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_enable_reads_from_master

(PECL maxdb:1.0-7.6.00.38)

maxdb_enable_reads_from_master — マスタからの読み込みを有効にする

説明

bool ***maxdb_enable_reads_from_master*** *(resource $link)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_enable_rpl_parse

(PECL maxdb:1.0-7.6.00.38)

maxdb_enable_rpl_parse — RPL のパースを有効にする

説明

bool ***maxdb_enable_rpl_parse*** *(resource $link)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_errno

maxdb->errno

(PECL maxdb:1.0-7.6.00.38)

maxdb->errno — 直近の関数コールのエラーコードを返す

説明

手続き型

int **maxdb_errno** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
int$errno;

maxdb_errno() 関数は、link で指定したデータベースリンクに関して、直近の MaxDB 関数コールのエラーコードを返します。 エラーが発生しなかった場合は、この関数はゼロを返します。

返り値

直近のコールのエラーコードの値を返します。 ゼロの場合は、エラーが発生しなかったことを意味します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if (!$maxdb->query("SELECT xxx FROM hotel.city")) {
   printf("エラーコード: %d\n", $maxdb->errno);
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if (!maxdb_query($link, "SELECT xxx FROM hotel.city")) {
   printf("エラーコード: %d\n", maxdb_errno($link));
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
PHP Warning:  maxdb_query(): -4005 POS(8) Unknown column name:XXX [42000] <...>
エラーコード: -4005
```

参考

- maxdb_connect_errno()
- maxdb_connect_error()
- maxdb_error()
- maxdb_sqlstate()

---

maxdb_error

(PECL maxdb:1.0-7.6.00.38)

maxdb_error — 直近のエラーについて説明する文字列を返す

説明

手続き型

```
string maxdb_error ( resource $link )
```

オブジェクト指向型（プロパティ）

```
maxdb
string$error;
```

maxdb_error() 関数は、対応する maxdb_errno() 関数とあらゆる点で同じです。 ただ **maxdb_error()** は、 整数のエラーコードは返しません。 代わりに、 link で表されるデータベース接続で、 直近に発生したエラーを説明する文字列を返します。 エラーが発生しなかった場合は、この関数は空の文字列を返します。

返り値

エラーを表す文字列を返します。 エラーが発生しなかった場合は空の文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if (!$maxdb->query("SELECT xxx FROM hotel.city")) {
   printf("エラーメッセージ: %s\n", $maxdb->error);
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if (!maxdb_query($link, "SELECT xxx FROM hotel.city")) {
   printf("エラーメッセージ: %s\n", $maxdb->error);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
PHP Warning:  maxdb_query(): -4005 POS(8) Unknown column name:XXX [42000]
エラーメッセージ: POS(8) Unknown column name:XXX
```

参考

- maxdb_connect_errno()
- maxdb_connect_error()
- maxdb_errno()
- maxdb_sqlstate()

---

maxdb_escape_string

(PECL maxdb:1.0)

maxdb_escape_string — maxdb_real_escape_string() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_real_escape_string().

---

maxdb_execute

(PECL maxdb:1.0)

maxdb_execute — maxdb_stmt_execute() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_execute().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_fetch_array

result->fetch_array

(No version information available, might be only in CVS)

result->fetch_array — 結果の行を連想配列、数値添字配列あるいはその両方で取得する

説明

手続き型

mixed **maxdb_fetch_array** (resource $result [, int $resulttype])

オブジェクト指向型（メソッド）

result
mixed **fetch_array** ([int $resulttype])

取得した行に対応する配列を返します。 result が表す結果セットに行がもう存在しない場合には **NULL** を返します。

maxdb_fetch_array() は maxdb_fetch_row() 関数を拡張したものです。 データを数値添字の配列に保存することに加え、**maxdb_fetch_array()** 関数は結果を連想配列でも保存します。 その際は、結果セットのフィールド名をキーとして使用します。

> **注意**: この関数により返されるフィールド名は 大文字小文字を区別 します。
>
> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

結果の中に同名のカラムが複数存在する場合は、最後のカラムが優先され、 その前に現れたデータを上書きします。同名の複数のカラムにアクセスするには、 数値添字形式の配列を使用します。

オプションの 2 番目の引数 resulttype は、 行データからどのような形式の配列を作成するかを指定する定数です。 このパラメータに指定可能な値は、 MAXDB_ASSOC、MAXDB_ASSOC_UPPER、MAXDB_ASSOC_LOWER、MAXDB_NUM あるいは MAXDB_BOTH のいずれかです。デフォルトでは **maxdb_fetch_array()** 関数は MAXDB_BOTH を使用します。これは、このパラメータに MAXDB_NUM および MAXDB_ASSOC を指定したのと同じ動作をします。

定数 MAXDB_ASSOC を使用すると、この関数は maxdb_fetch_assoc() と同じように動作します。一方、MAXDB_NUM の場合は maxdb_fetch_row() と同じ動作となります。 最後のオプション MAXDB_BOTH は、両方の属性をひとつの配列に含めます。

定数 MAXDB_ASSOC_UPPER を使用すると、この関数の動作は MAXDB_ASSOC を使用した場合とほぼ同じになります。ただ、 配列のインデックスが、フィールド名を大文字にしたものになるという点が違います。

定数 MAXDB_ASSOC_LOWER を使用すると、この関数の動作は MAXDB_ASSOC を使用した場合とほぼ同じになります。ただ、 配列のインデックスが、フィールド名を小文字にしたものになるという点が違います。

返り値

取得した行に対応する配列、あるいは結果セットに行がもうない場合に **NULL** を返します。

例

Example#1 オブジェクト指向型

```php
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";
$result = $maxdb->query($query);

/* 数値添字の配列 */
$row = $result->fetch_array(MAXDB_NUM);
printf ("%s (%s)¥n", $row[0], $row[1]);

/* 連想配列 */
$row = $result->fetch_array(MAXDB_ASSOC);
printf ("%s (%s)¥n", $row["NAME"], $row["STATE"]);

/* 連想配列および数値添字の配列 */
$row = $result->fetch_array(MAXDB_BOTH);
printf ("%s (%s)¥n", $row[0], $row["STATE"]);

/* 結果セットを開放します */
$result->close();

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";
$result = maxdb_query($link, $query);

/* 数値添字の配列 */
$row = maxdb_fetch_array($result, MAXDB_NUM);
printf ("%s  (%s)¥n", $row[0], $row[1]);

/* 連想配列 */
$row = maxdb_fetch_array($result, MAXDB_ASSOC);
printf ("%s  (%s)¥n", $row["NAME"], $row["STATE"]);

/* 連想配列および数値添字の配列 */
$row = maxdb_fetch_array($result, MAXDB_BOTH);
printf ("%s  (%s)¥n", $row[0], $row["STATE"]);

/* 結果セットを開放します */
maxdb_free_result($result);

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
New York (NY)
New York (NY)
Long Island (NY)
```

参考

- maxdb_fetch_assoc()
- maxdb_fetch_row()
- **maxdb_fetch_resource()**

---

maxdb_fetch_assoc

maxdb->fetch_assoc

(PECL maxdb:1.0-7.6.00.38)

maxdb->fetch_assoc — 結果の行を連想配列として取得する

説明

手続き型

array **maxdb_fetch_assoc** (resource $result)

オブジェクト指向型（メソッド）

result
array **fetch_assoc** (void)

取得した行に対応する連想配列を返します。行がもう存在しない場合には **NULL** を返します。

maxdb_fetch_assoc() 関数は、result が表す結果の結果セット内の次の行を表す連想配列を返すために使用されます。 連想配列の各キーは、結果セットのカラム名を表します。

結果の中に同名のカラムが複数存在する場合は、最後のカラムが優先されます。 同名の他のカラムにアクセスするには、 maxdb_fetch_row() を使用して数値添字の配列を使用するか、 あるいはカラム名のエイリアスを使用します。

> **注意**: この関数により返されるフィールド名は 大文字小文字を区別 します。
>
> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

返り値

取得した行に対応する配列、あるいは結果セットに行がもうない場合に **NULL** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
```

```
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = $maxdb->query($query)) {

   /* 連想配列を取得します */
   while ($row = $result->fetch_assoc()) {
       printf ("%s (%s)¥n", $row["NAME"], $row["STATE"]);
   }

   /* 結果セットを開放します */
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = maxdb_query($link, $query)) {

   /* 連想配列を取得します */
   while ($row = maxdb_fetch_assoc($result)) {
       printf ("%s (%s)¥n", $row["NAME"], $row["STATE"]);
   }

   /* 結果セットを開放します */
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
New York (NY)
New York (NY)
Long Island (NY)
Albany (NY)
Washington (DC)
Washington (DC)
Washington (DC)
Silver Spring (MD)
Daytona Beach (FL)
Deerfield Beach (FL)
Clearwater (FL)
Cincinnati (OH)
Detroit (MI)
Rosemont (IL)
Chicago (IL)
Chicago (IL)
New Orleans (LA)
Dallas (TX)
Los Angeles (CA)
Hollywood (CA)
Long Beach (CA)
Palm Springs (CA)
Irvine (CA)
Santa Clara (CA)
Portland (OR)
```

参考

- *maxdb_fetch_array()*
- *maxdb_fetch_row()*
- ***maxdb_fetch_resource()***

---

maxdb_fetch_field_direct

result->fetch_field_direct

(No version information available, might be only in CVS)

result->fetch_field_direct — 単一のフィールドのメタデータを取得する

説明

手続き型

mixed **maxdb_fetch_field_direct** (resource $result , int $fieldnr)

オブジェクト指向型（メソッド）

result
mixed **fetch_field_direct** (int $fieldnr)

maxdb_fetch_field_direct() は、 指定した結果セットのフィールド定義情報を含むリソースを返します。 fieldnr の値は、0 から フィールド数 - 1 の間でなければなりません。

返り値

フィールド定義情報を含むリソースを返します。 指定した fieldnr に対応するフィールド情報が存在しない場合は **FALSE** を返します。

オブジェクトの属性

属性	説明
name	カラムの名前
max_length	結果セットのフィールドの最大幅
type	このフィールドのデータ型
decimals	使用している桁数 (整数フィールドの場合)

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY name";

if ($result = $maxdb->query($query)) {

   /* 'SurfaceArea' のフィールド情報を取得します */
   $finfo = $result->fetch_field_direct(1);

   printf("Name:      %s¥n", $finfo->name);
   printf("Table:     %s¥n", $finfo->table);
   printf("max. Len:  %d¥n", $finfo->max_length);
   printf("Flags:     %d¥n", $finfo->flags);
   printf("Type:      %d¥n", $finfo->type);

   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY name";

if ($result = maxdb_query($link, $query)) {

   /* 'SurfaceArea' のフィールド情報を取得します */
   $finfo = maxdb_fetch_field_direct($result, 1);

   printf("Name:      %s¥n", $finfo->name);
   printf("Table:     %s¥n", $finfo->table);
   printf("max. Len:  %d¥n", $finfo->max_length);
   printf("Flags:     %d¥n", $finfo->flags);
   printf("Type:      %d¥n", $finfo->type);

   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Name:      CNO
```

```
Table:
max. Len:  4
Flags:     -1
Type:      0
```

参考

- maxdb_num_fields()
- maxdb_fetch_field()
- maxdb_fetch_fields()

---

maxdb_fetch_field

result->fetch_field

(No version information available, might be only in CVS)

result->fetch_field — 結果セットの次のフィールドを返す

説明

手続き型

```
mixed maxdb_fetch_field ( resource $result )
```

オブジェクト指向型（メソッド）

```
result
mixed fetch_field ( void )
```

maxdb_fetch_field() は、結果セットのカラムの定義を リソースとして返します。この関数を繰り返しコールすると、 結果セットのすべてのカラムの情報を取得できます。 **maxdb_fetch_field()** は、フィールドがもう残っていない場合に **FALSE** を返します。

返り値

フィールド定義の情報を含むリソースを返します。 フィールド情報が取得できない場合には **FALSE** を返します。

オブジェクトのプロパティ

プロパティ	説明
name	カラムの名前
max_length	結果セットのフィールドの最大幅
type	このフィールドのデータ型
decimals	使用している桁数 (整数フィールドの場合)

例

Example#1 オブジェクト指向型

```php
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = $maxdb->query($query)) {

   /* すべてのカラムのフィールド情報を取得します */
   while ($finfo = $result->fetch_field()) {

        printf("Name:      %s¥n", $finfo->name);
        printf("Table:     %s¥n", $finfo->table);
        printf("max. Len:  %d¥n", $finfo->max_length);
        printf("Flags:     %d¥n", $finfo->flags);
        printf("Type:      %d¥n¥n", $finfo->type);
   }
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```php
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");
```

```
/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* すべてのカラムのフィールド情報を取得します */
   while ($finfo = maxdb_fetch_field($result)) {

        printf("Name:      %s¥n", $finfo->name);
        printf("Table:     %s¥n", $finfo->table);
        printf("max. Len:  %d¥n", $finfo->max_length);
        printf("Flags:     %d¥n", $finfo->flags);
        printf("Type:      %d¥n¥n", $finfo->type);
   }
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Name:      NAME
Table:
max. Len:  10
Flags:     -1
Type:      2

Name:      CNO
Table:
max. Len:  4
Flags:     -1
Type:      0
```

参考

- maxdb_num_fields()
- maxdb_fetch_field_direct()
- maxdb_fetch_fields()
- maxdb_field_seek()

---

maxdb_fetch_fields

result->fetch_fields

(No version information available, might be only in CVS)

result->fetch_fields — 結果セット内のフィールドを表すリソースの配列を返す

説明

手続き型

mixed **maxdb_fetch_fields** (resource $result)

オブジェクト指向型（メソッド）

result
mixed **fetch_fields** (void)

この関数は、maxdb_fetch_field() と同じ目的で使用します。ただひとつの違いは、 各フィールドに対してひとつづつリソースを返すのではなく、リソースの配列を返すという点です。

返り値

フィールド定義の情報を含む、リソースの配列を返します。 フィールド情報が存在しない場合には **FALSE** を返します。

オブジェクトのプロパティ

プロパティ	説明
name	カラムの名前
max_length	結果セットのフィールドの最大幅
type	このフィールドのデータ型
decimals	使用している桁数 (整数フィールドの場合)

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = $maxdb->query($query)) {

   /* すべてのカラムのフィールド情報を取得します */
   $finfo = $result->fetch_fields();

    foreach ($finfo as $val) {
        printf("Name:      %s¥n", $val->name);
        printf("Table:     %s¥n", $val->table);
        printf("max. Len:  %d¥n", $val->max_length);
        printf("Flags:     %d¥n", $val->flags);
        printf("Type:      %d¥n¥n", $val->type);
   }
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* すべてのカラムのフィールド情報を取得します */
   $finfo = maxdb_fetch_fields($result);

    foreach ($finfo as $val) {
        printf("Name:      %s¥n", $val->name);
        printf("Table:     %s¥n", $val->table);
        printf("max. Len:  %d¥n", $val->max_length);
        printf("Flags:     %d¥n", $val->flags);
        printf("Type:      %d¥n¥n", $val->type);
   }
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Name:      NAME
Table:
max. Len:  10
Flags:     -1
Type:      2

Name:      CNO
Table:
max. Len:  4
Flags:     -1
Type:      0
```

参考

- maxdb_num_fields()
- maxdb_fetch_field()
- maxdb_fetch_field_direct()

---

maxdb_fetch_lengths

result->lengths

(No version information available, might be only in CVS)

result->lengths — 結果セットの現在の行のカラムの長さを返す

説明

手続き型

array **maxdb_fetch_lengths** (resource $result)

オブジェクト指向型（プロパティ）

result
array$lengths;

maxdb_fetch_lengths() 関数は、result が表す結果セット内の現在の行の、すべてのカラムの長さを配列で返します。 成功した場合はカラムの長さを含む数値添字の配列、 失敗した場合は **FALSE** を返します。

返り値

各カラムの長さ（終端の null 文字を含まない）を表す整数値の配列、 あるいはエラー時には **FALSE** を返します。

maxdb_fetch_lengths() は、 結果セットの現在の行に対してのみ有効です。 maxdb_fetch_row/array/resource をコールする前や、 結果セットのすべての行を取得した後にこの関数をコールすると、 **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT * from hotel.customer WHERE cno = 3000";

if ($result = $maxdb->query($query)) {

   $row = $result->fetch_row();

   /* カラムの長さを表示します */
   foreach ($result->lengths as $i => $val) {
       printf("フィールド %2d の長さは %2d\n", $i+1, $val);
   }
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT * from hotel.customer WHERE cno = 3000";

if ($result = maxdb_query($link, $query)) {

   $row = maxdb_fetch_row($result);

   /* カラムの長さを表示します */
   foreach (maxdb_fetch_lengths($result) as $i => $val) {
       printf("フィールド %2d の長さは %2d\n", $i+1, $val);
   }
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
フィールド  1 の長さは  4
フィールド  2 の長さは  3
フィールド  3 の長さは  5
フィールド  4 の長さは  6
フィールド  5 の長さは  5
フィールド  6 の長さは 21
```

maxdb_fetch_object

result->fetch_object

(No version information available, might be only in CVS)

result->fetch_object — 結果セットの現在の行をオブジェクトとして返す

説明

手続き型

object **maxdb_fetch_object** (object $result)

オブジェクト指向型（メソッド）

result
object **fetch_object** (void)

maxdb_fetch_object() は、結果セットの現在の行を オブジェクトとして返します。オブジェクトの属性が結果セットのフィールド名に対応します。 現在の結果セットに行がもう存在しない場合は **NULL** が返されます。

返り値

取得した行に対応するオブジェクトを返します。行がもう存在しない場合には **NULL** を返します。

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = $maxdb->query($query)) {

   /* オブジェクトの配列を取得します */
   while ($obj = $result->fetch_object()) {
       printf ("%s (%s)¥n", $obj->NAME, $obj->STATE);
   }

   /* 結果セットを開放します */
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = maxdb_query($link, $query)) {

   /* オブジェクトの配列を取得します */
   while ($obj = maxdb_fetch_object($result)) {
       printf ("%s (%s)¥n", $obj->NAME, $obj->STATE);
   }

   /* 結果セットを開放します */
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
New York (NY)
New York (NY)
Long Island (NY)
Albany (NY)
Washington (DC)
```

```
Washington (DC)
Washington (DC)
Silver Spring (MD)
Daytona Beach (FL)
Deerfield Beach (FL)
Clearwater (FL)
Cincinnati (OH)
Detroit (MI)
Rosemont (IL)
Chicago (IL)
Chicago (IL)
New Orleans (LA)
Dallas (TX)
Los Angeles (CA)
Hollywood (CA)
Long Beach (CA)
Palm Springs (CA)
Irvine (CA)
Santa Clara (CA)
Portland (OR)
```

参考

- maxdb_fetch_array()
- maxdb_fetch_assoc()
- maxdb_fetch_row()

---

maxdb_fetch_row

result->fetch_row

(No version information available, might be only in CVS)

result->fetch_row — 結果の行を数値添字の配列として取得する

説明

手続き型

```
mixed maxdb_fetch_row ( resource $result )
```

オブジェクト指向型（メソッド）

```
result
mixed fetch_row ( void )
```

取得した行に対応する連想配列を返します。行がもう存在しない場合には **NULL** を返します。

maxdb_fetch_row() は、result で表される結果セットから行のデータを取得し、 それを数値添字の配列で返します。各カラムは、0（ゼロ）から始まる添字で保存されます。これ以降の **maxdb_fetch_row()** 関数のコールでは、 結果セット内の次の行を返します。もう行が存在しない場合には **FALSE** を返します。

返り値

maxdb_fetch_row() は、取得した行に対応する配列を返します。 結果セットに行がもうない場合には **NULL** を返します。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

例

Example#1 オブジェクト指向型

```php
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = $maxdb->query($query)) {

   /* 数値添字の配列を取得します */
   while ($row = $result->fetch_row()) {
       printf ("%s (%s)¥n", $row[0], $row[1]);
   }

   /* 結果セットを開放します */
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, state FROM hotel.city ORDER by zip";

if ($result = maxdb_query($link, $query)) {

   /* 数値添字の配列を取得します */
   while ($row = maxdb_fetch_row($result)) {
       printf ("%s (%s)¥n", $row[0], $row[1]);
   }

   /* 結果セットを開放します */
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
New York (NY)
New York (NY)
Long Island (NY)
Albany (NY)
Washington (DC)
Washington (DC)
Washington (DC)
Silver Spring (MD)
Daytona Beach (FL)
Deerfield Beach (FL)
Clearwater (FL)
Cincinnati (OH)
Detroit (MI)
Rosemont (IL)
Chicago (IL)
Chicago (IL)
New Orleans (LA)
Dallas (TX)
Los Angeles (CA)
Hollywood (CA)
Long Beach (CA)
Palm Springs (CA)
Irvine (CA)
Santa Clara (CA)
Portland (OR)
```

参考

- maxdb_fetch_array()
- maxdb_fetch_assoc()
- **maxdb_fetch_resource()**

---

maxdb_fetch

(PECL maxdb:1.0)

maxdb_fetch — maxdb_stmt_fetch() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_fetch().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_field_count

maxdb->field_count

(PECL maxdb:1.0-7.6.00.38)

maxdb->field_count — 直近のクエリのカラム数を返す

説明

手続き型

int ***maxdb_field_count*** *(resource $link)*

オブジェクト指向型（メソッド）

maxdb
int ***field_count*** *(void)*

link で表される接続の、 直近のクエリのカラム数を返します。クエリの詳細を知らなくても そのクエリが結果を返すかどうかを知ることができるので、 この関数は maxdb_store_result() を使用する際に有用です。

返り値

結果セットのフィールド数を表す整数値を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

maxdb_report (MAXDB_REPORT_OFF);
$maxdb->query("DROP TABLE friends");
maxdb_report (MAXDB_REPORT_ERROR);

$maxdb->query( "CREATE TABLE friends (id int, name varchar(20))");

$maxdb->query( "INSERT INTO friends VALUES (1,'Hartmut')");
$maxdb->query( "INSERT INTO friends VALUES (2, 'Ulf')");

if ($maxdb->field_count()) {
   /* select/show あるいは describe クエリだった場合 */
   $result = $maxdb->store_result();

   /* 結果セットを処理します */
   $row = $result->fetch_row();

   /* 結果セットを開放します */
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE friends");
maxdb_report (MAXDB_REPORT_ERROR);

maxdb_query($link, "CREATE TABLE friends (id int, name varchar(20))");

maxdb_query($link, "INSERT INTO friends VALUES (1,'Hartmut')");
maxdb_query($link, "INSERT INTO friends VALUES (2, 'Ulf')");

if (maxdb_field_count($link)) {
   /* select/show あるいは describe クエリだった場合 */
   $result = maxdb_store_result($link);

   /* 結果セットを処理します */
   $row = maxdb_fetch_row($result);

   /* 結果セットを開放します */
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例は、何も出力しません。

---

maxdb_field_seek

result->field_seek

(No version information available, might be only in CVS)

result->field_seek — 結果ポインタを、指定したフィールドオフセットに移動する

説明

手続き型

bool ***maxdb_field_seek*** *(resource $result , int $fieldnr)*

オブジェクト指向型（メソッド）

result
bool field_seek (int $fieldnr)

フィールドカーソルを、指定したオフセットに移動します。 次に maxdb_fetch_field() をコールすると、 そのオフセットに関連付けられたカラムのフィールド定義が取得されます。

注意: 行の最初に移動するには、オフセットの値にゼロを指定します。

返り値

maxdb_field_seek() は、 フィールドカーソルの変更前の値を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = $maxdb->query($query)) {

   /* 2 番目のカラムのフィールド情報を取得します */
   $result->field_seek(1);
   $finfo = $result->fetch_field();

   printf("Name:      %s¥n", $finfo->name);
   printf("Table:     %s¥n", $finfo->table);
   printf("max. Len:  %d¥n", $finfo->max_length);
   printf("Flags:     %d¥n", $finfo->flags);
   printf("Type:      %d¥n¥n", $finfo->type);

   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* 2 番目のカラムのフィールド情報を取得します */
   maxdb_field_seek($result, 1);
   $finfo = maxdb_fetch_field($result);

   printf("Name:      %s¥n", $finfo->name);
   printf("Table:     %s¥n", $finfo->table);
   printf("max. Len:  %d¥n", $finfo->max_length);
   printf("Flags:     %d¥n", $finfo->flags);
   printf("Type:      %d¥n¥n", $finfo->type);

   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Name:      NAME
Table:
max. Len:  10
Flags:     -1
Type:      2
```

参考

- maxdb_fetch_field()

---

maxdb_field_tell

result->current_field

(No version information available, might be only in CVS)

result->current_field — 結果ポインタの現在のフィールドオフセットを取得する

説明

手続き型

int **maxdb_field_tell** (resource $result)

オブジェクト指向型（プロパティ）

result
int $current_field ;

直近の maxdb_fetch_field() コールで使用した フィールドカーソルの位置を返します。この値は、maxdb_field_seek() の引数として使用されます。

返り値

フィールドカーソルの現在のオフセットを返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = $maxdb->query($query)) {

   /* すべてのカラムのフィールド情報を取得します */
   while ($finfo = $result->fetch_field()) {

       /* フィールドポインタのオフセットを取得します */
       $currentfield = $result->current_field;

       printf("Column      %d:¥n", $currentfield);
       printf("Name:       %s¥n", $finfo->name);
       printf("Table:      %s¥n", $finfo->table);
       printf("max. Len:  %d¥n", $finfo->max_length);
       printf("Flags:      %d¥n", $finfo->flags);
       printf("Type:       %d¥n¥n", $finfo->type);
   }
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, cno from hotel.customer ORDER BY cno";

if ($result = maxdb_query($link, $query)) {

   /* すべてのカラムのフィールド情報を取得します */
   while ($finfo = maxdb_fetch_field($result)) {

       /* フィールドポインタのオフセットを取得します */
       $currentfield = maxdb_field_tell($result);

       printf("Column      %d:¥n", $currentfield);
       printf("Name:       %s¥n", $finfo->name);
       printf("Table:      %s¥n", $finfo->table);
       printf("max. Len:  %d¥n", $finfo->max_length);
       printf("Flags:      %d¥n", $finfo->flags);
       printf("Type:       %d¥n¥n", $finfo->type);
   }
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Column     1:
Name:      NAME
Table:
max. Len:  10
Flags:     -1
Type:      2

Column     2:
Name:      CNO
Table:
max. Len:  4
Flags:     -1
Type:      0
```

参考

- maxdb_fetch_field()
- maxdb_field_seek()

---

maxdb_free_result

result->free

(No version information available, might be only in CVS)

result->free — 結果に関連付けられたメモリを開放する

説明

手続き型

void **maxdb_free_result** (resource $result)

オブジェクト指向型（メソッド）

result
void **free** (void)

maxdb_free_result() 関数は、 result が表す結果に関連付けられたメモリを開放します。 このメモリは、maxdb_query()、maxdb_store_result() あるいは maxdb_use_result() で割り当てられたものです。

注意: 結果リソースが必要なくなった際には、常に **maxdb_free_result()** で結果を開放すべきです。

返り値

この関数は値を返しません。

参考

- maxdb_query()
- maxdb_stmt_store_result()
- maxdb_store_result()
- maxdb_use_result()

---

maxdb_get_client_info

(PECL maxdb:1.0-7.6.00.38)

maxdb_get_client_info — MaxDB クライアントのバージョンを文字列で返す

説明

string **maxdb_get_client_info** (void)

maxdb_get_client_info() 関数は、 MaxDB 拡張モジュールで使用されている、 クライアントのバージョンを表す文字列を返すために使用されます。

返り値

MaxDB クライアントライブラリのバージョンを表す文字列を返します。

例

Example#1 maxdb_get_client_info

```
<?php

/* MaxDB クライアントライブラリのバージョンを
   知るには、接続する必要はありません。 */
```

```
printf("クライアントライブラリのバージョン:  %s\n", maxdb_get_client_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
クライアントライブラリのバージョン:  libSQLDBC <...>
```

参考

- maxdb_get_client_version()
- maxdb_get_server_info()
- maxdb_get_server_version()

---

maxdb_get_client_version

(PECL maxdb:1.0-7.6.00.38)

maxdb_get_client_version — MaxDB クライアントの情報を取得する

説明

int **maxdb_get_client_version** (void)

クライアントのバージョン番号を整数値で返します。

返り値

MaxDB クライアントライブラリのバージョンを表す数値を、 以下のフォーマットで返します。 main_version*10000 + minor_version *100 + sub_version 例えば、7.5.0 の場合は 70500 となります。

これは、何らかの機能が存在するかどうかを知るために クライアントライブラリのバージョンを手早く取得するなどの際に有用です。

例

Example#1 maxdb_get_client_version

```
<?php

/* MaxDB クライアントライブラリのバージョンを
    知るには、接続する必要はありません。 */

printf("クライアントライブラリのバージョン:  %d\n", maxdb_get_client_version());
?>
```

上の例の出力は、たとえば以下のようになります。

```
クライアントライブラリのバージョン:  7.<...>
```

参考

- maxdb_get_client_info()
- maxdb_get_server_info()
- maxdb_get_server_version()

---

maxdb_get_host_info

maxdb->get_host_info

(PECL maxdb:1.0-7.6.00.38)

maxdb->get_host_info — 使用している接続の型を表す文字列を返す

説明

手続き型

string **maxdb_get_host_info** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
string$host_info;

maxdb_get_host_info() 関数は、 link で表される接続を表す文字列を返します。

返り値

サーバのホスト名および接続の型を表す文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* ホスト情報を表示します */
printf("ホスト情報: %s\n", $maxdb->host_info);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* ホスト情報を表示します */
printf("ホスト情報: %s\n", maxdb_get_host_info($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
ホスト情報: localhost
```

参考

- maxdb_get_proto_info()

---

maxdb_get_metadata

(PECL maxdb:1.0)

maxdb_get_metadata — maxdb_stmt_result_metadata() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_result_metadata().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_get_proto_info

maxdb->protocol_version

(No version information available, might be only in CVS)

maxdb->protocol_version — 使用している MaxDB プロトコルのバージョンを返す

説明

手続き型

```
int maxdb_get_proto_info ( resource $link )
```

オブジェクト指向型（プロパティ）

```
maxdb
string$protocol_version;
```

link で表される接続で使用している MaxDB プロトコルのバージョンを表す整数値を返します。

返り値

プロトコルのバージョンを表す整数値（定数の 10）を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb  = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

/* プロトコルのバージョンを出力します */
printf("プロトコルのバージョン:  %d\n", $maxdb->protocol_version);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link  = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

/* プロトコルのバージョンを出力します */
printf("プロトコルのバージョン:  %d\n", maxdb_get_proto_info($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
プロトコルのバージョン: 10
```

参考

- maxdb_get_host_info()

---

maxdb_get_server_info

maxdb->server_info

(No version information available, might be only in CVS)

maxdb->server_info — MaxDB サーバのバージョンを返す

説明

手続き型

string **maxdb_get_server_info** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
string$server_info;

MaxDB 拡張モジュールが接続している（link パラメータで表される）MaxDB サーバのバージョンを表す文字列を返します。

返り値

サーバのバージョンを表す文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb  = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

/* サーバのバージョンを表示します */
printf("サーバのバージョン:  %s\n", maxdb_get_server_info($link));
```

```
/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* サーバのバージョンを表示します */
printf("サーバのバージョン: %s\n", maxdb_get_server_info($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
サーバのバージョン: Kernel     7<...>
```

参考

- maxdb_get_client_info()
- maxdb_get_client_version()
- maxdb_get_server_version()

---

maxdb_get_server_version

(PECL maxdb:1.0-7.6.00.38)

maxdb_get_server_version — MaxDB サーバのバージョンを整数値で返す

説明

手続き型

```
int maxdb_get_server_version ( resource $link )
```

オブジェクト指向型（プロパティ）

```
maxdb
int$server_version;
```

maxdb_get_server_version() 関数は、 MaxDB 拡張モジュールが接続している (link パラメータで表される) MaxDB サーバのバージョンを表す整数値を返します。

バージョン番号のフォーマットは main_version * 10000 + minor_version * 100 + sub_version (つまり、バージョン 7.5.0 は 70500) となります。

返り値

サーバのバージョンを表す整数値を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* サーバのバージョンを表示します */
printf("サーバのバージョン: %d\n", $maxdb->server_version);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
```

```
}

/* サーバのバージョンを表示します */
printf("サーバのバージョン:  %d\n", maxdb_get_server_version($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
サーバのバージョン: 7<...>
```

参考

- maxdb_get_client_info()
- maxdb_get_client_version()
- maxdb_get_server_info()

---

maxdb_info

maxdb->info

(PECL maxdb:1.0-7.6.00.38)

maxdb->info — 直近に実行したクエリについての情報を取得する

説明

手続き型

```
string maxdb_info ( resource $link )
```

オブジェクト指向型（プロパティ）

```
maxdb
string$info;
```

maxdb_info() は、最後に実行されたクエリについての情報を 文字列で返します。文字列の内容は、以下のようになります。

maxdb_info の返す値

クエリの型	返される文字列の例
INSERT INTO...SELECT...	Records: 100 Duplicates: 0 Warnings: 0
INSERT INTO...VALUES (...),(...),(...)	Records: 3 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...	Records: 1 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE ...	Records: 3 Duplicates: 0 Warnings: 0
UPDATE ...	Rows matched: 40 Changed: 40 Warnings: 0

注意: 上のいずれにもあてはまらない形式のクエリはサポートされていません。 そのような場合、**maxdb_info()** は空の文字列を返します。

返り値

直近に実行されたクエリについての追加情報を表す文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb  = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query("CREATE  TABLE  temp.t1  LIKE  hotel.city");

/* INSERT INTO .. SELECT */
$maxdb->query("INSERT  INTO  temp.t1  SELECT  *  FROM  hotel.city");
printf("%s¥n", $maxdb->info);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_query($link, "CREATE TABLE temp.t1 LIKE hotel.city");

/* INSERT INTO .. SELECT */
maxdb_query($link, "INSERT INTO temp.t1 SELECT * FROM hotel.city");
printf("%s¥n", maxdb_info($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Records: 25 Duplicates: 0 Warnings: 0
```

参考

- maxdb_affected_rows()
- maxdb_warning_count()
- maxdb_num_rows()

---

maxdb_init

(PECL maxdb:1.0-7.6.00.38)

maxdb_init — MaxDB を初期化し、maxdb_real_connect で使用するリソースを返す

説明

resource **maxdb_init** (void)

maxdb_options() および maxdb_real_connect() で使用する MaxDB リソースを確保、あるいは初期化します。

> **注意**: maxdb_real_connect() がコールされるまでは、 これ以降の (maxdb_options() 以外の) maxdb 関数のコールは失敗します。

返り値

リソースを返します。

参考

- maxdb_options()
- maxdb_close()
- maxdb_real_connect()
- maxdb_connect()

---

maxdb_insert_id

maxdb->insert_id

(PECL maxdb:1.0-7.6.00.38)

maxdb->insert_id — 直近のクエリで使用した、自動生成 ID を返す

説明

手続き型

mixed **maxdb_insert_id** (resource $link)

オブジェクト指向型 (プロパティ)

maxdb
mixed$insert_id;

maxdb_insert_id() 関数は、DEFAULT SERIAL 属性をもつカラムが存在するテーブルに対するクエリが生成した ID を返します。 直近のクエリが INSERT や UPDATE でなかった場合、 あるいは対象のテーブルに DEFAULT SERIAL 属性をもつカラムが存在しなかった場合は、 この関数はゼロを返します。

返り値

直近のクエリによって更新された DEFAULT SERIAL フィールドの値を返します。直近のクエリがない場合、 あるいは直近のクエリが DEFAULT_SERIAL を更新しなかった場合はゼロを返します。

注意: 結果の数値が int の最大値をこえる場合は、**maxdb_insert_id()** は文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_report (MAXDB_REPORT_OFF);
$maxdb->query("DROP TABLE mycity");
maxdb_report (MAXDB_REPORT_ERROR);

$maxdb->query("CREATE TABLE mycity LIKE hotel.city");
$maxdb->query("ALTER TABLE mycity ADD id FIXED(11) DEFAULT SERIAL");

$query = "INSERT INTO mycity (zip,name,state) VALUES ('12203','Albany' ,'NY')";
$maxdb->query($query);

printf ("新しいレコードの ID は、%d です。\n", $maxdb->insert_id);

/* テーブルを削除します */
$maxdb->query("DROP TABLE mycity");

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP TABLE mycity");
maxdb_report (MAXDB_REPORT_ERROR);

maxdb_query($link, "CREATE TABLE mycity LIKE hotel.city");
maxdb_query($link, "ALTER TABLE mycity ADD id FIXED(11) DEFAULT SERIAL");

$query = "INSERT INTO mycity (zip,name,state) VALUES ('12203','Albany' ,'NY')";
maxdb_query($link, $query);

printf ("新しいレコードの ID は、%d です。\n", $maxdb->insert_id);

/* テーブルを削除します */
maxdb_query($link, "DROP TABLE mycity");

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
新しいレコードの ID は、1 です。
```

---

maxdb_kill

maxdb->kill

(PECL maxdb:1.0-7.6.00.38)

maxdb->kill — MaxDB サーバから切断する

説明

手続き型

bool **maxdb_kill** (resource $link , int $processid)

オブジェクト指向型 (メソッド)

maxdb
bool **kill** (int $processid)

この関数は、processid で指定した MaxDB サーバから切断するために使用されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* スレッド ID を調べます */
$thread_id = $maxdb->thread_id;

/* 接続を殺します */
$maxdb->kill($thread_id);

/* これは、エラーとなります */
if (!$maxdb->query("CREATE TABLE myCity LIKE City")) {
   printf("エラー: %s\n", $maxdb->error);
    exit;
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* スレッド ID を調べます */
$thread_id = maxdb_thread_id($link);

/* 接続を殺します */
maxdb_kill($link, $thread_id);

/* これは、エラーとなります */
if (!maxdb_query($link, "CREATE TABLE myCity LIKE City")) {
   printf("エラー: %s\n", maxdb_error($link));
    exit;
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
エラー: Session not connected
```

参考

- maxdb_thread_id()

---

maxdb_master_query

(PECL maxdb:1.0-7.6.00.38)

maxdb_master_query — マスタ/スレーブ構成において、クエリをマスタ側で実行することを強制する

説明

```
bool maxdb_master_query ( resource $link , string $query )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_more_results

maxdb->more_results

(PECL maxdb:1.0-7.6.00.38)

maxdb->more_results — 複数クエリの結果の中に結果セットがまだあるかどうかを調べる

説明

bool **maxdb_more_results** (resource $link)

maxdb_more_results() は、事前の maxdb_multi_query() のコールによって取得した結果に まだ結果セットが含まれているかどうかを示します。

返り値

常に **FALSE** を返します。

例

maxdb_multi_query() を参照ください。

参考

- maxdb_multi_query()
- maxdb_next_result()
- maxdb_store_result()
- maxdb_use_result()

---

maxdb_multi_query

maxdb->multi_query

(PECL maxdb:1.0-7.6.00.38)

maxdb->multi_query — データベース上でクエリを実行する

説明

手続き型

bool **maxdb_multi_query** (resource $link , string $query)

オブジェクト指向型（メソッド）

maxdb
bool **multi_query** (string $query)

maxdb_multi_query() は、 maxdb_query() と同様の動作をします。 複数クエリは、まだサポートされていません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query  = "SELECT * FROM dual";

/* 複数クエリを実行します */
if ($maxdb->multi_query($query)) {
    do {
        /* 最初の結果セットを保存します */
        if ($result = $maxdb->store_result()) {
            while ($row = $result->fetch_row()) {
                printf("%s¥n", $row[0]);
            }
            $result->close();
        }
        /* 区切りを表示します */
        if ($maxdb->more_results()) {
            printf("-----------------¥n");
        }
    } while ($maxdb->next_result());
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```php
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT * FROM dual";

/* 複数クエリを実行します */
if (maxdb_multi_query($link, $query)) {
    do {
        /* 最初の結果セットを保存します */
        if ($result = maxdb_store_result($link)) {
            while ($row = maxdb_fetch_row($result)) {
                printf("%s¥n", $row[0]);
            }
            maxdb_free_result($result);
        }
        /* 区切りを表示します */
        if (maxdb_more_results($link)) {
            printf("-----------------¥n");
        }
    } while (maxdb_next_result($link));
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
a
```

参考

- maxdb_use_result()
- maxdb_store_result()
- maxdb_next_result()
- maxdb_more_results()

---

maxdb_next_result

maxdb->next_result

(PECL maxdb:1.0-7.6.00.38)

maxdb->next_result — multi_query の、次の結果を準備する

説明

bool **maxdb_next_result** (resource $link)

複数クエリはまだサポートされていないので、**maxdb_next_result()** は常に **FALSE** を返します。

返り値

FALSE を返します。

参考

- maxdb_multi_query()
- maxdb_more_results()
- maxdb_store_result()
- maxdb_use_result()

---

maxdb_num_fields

result->field_count

(No version information available, might be only in CVS)

result->field_count — 結果のフィールド数を取得する

説明

手続き型

```
int maxdb_num_fields ( resource $result )
```

オブジェクト指向型（プロパティ）

```
result
int$field_count;
```

maxdb_num_fields() は、 指定した結果セットのフィールド数を返します。

返り値

結果セットのフィールド数を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if ($result = $maxdb->query("SELECT * FROM hotel.city ORDER BY zip")) {

   /* 結果セットのフィールド数を調べます */
   $field_cnt = $result->field_count;

   printf("結果セットのフィールド数は %d です。\n", $field_cnt);

   /* 結果セットを閉じます */
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if ($result = maxdb_query($link, "SELECT * FROM hotel.city ORDER BY zip")) {

   /* 結果セットのフィールド数を調べます */
   $field_cnt = maxdb_num_fields($result);

   printf("結果セットのフィールド数は %d です。\n", $field_cnt);

   /* 結果セットを閉じます */
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
結果セットのフィールド数は 3 です。
```

参考

- maxdb_fetch_field()

---

maxdb_num_rows

(PECL maxdb:1.0-7.6.00.38)

maxdb_num_rows — 結果の行数を取得する

説明

手続き型

```
int maxdb_num_rows ( resource $result )
```

オブジェクト指向型（プロパティ）

```
maxdb
```

int$num_rows;

結果セットの行数を返します。

使用している結果セットがバッファ化されているかどうかによって、 **maxdb_num_rows()** の使用法は変わります。 バッファ化されていない結果セットの場合、 結果セットのすべての行を取得するまでは **maxdb_num_rows()** は正しい結果を返しません。

返り値

結果セットの行の数を返します。

> **注意**: 行数が int 型の最大値をこえる場合は、結果が文字列として返されます。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if ($result = $maxdb->query("SELECT cno, name FROM hotel.customer ORDER BY name")) {

   /* 結果セットの行数を取得します */
   $row_cnt = $result->num_rows;

   printf("Result set has %d rows.¥n", $row_cnt);

   /* 結果セットを閉じます */
   $result->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

if ($result = maxdb_query($link, "SELECT cno, name FROM hotel.customer ORDER BY name")) {

   /* 結果セットの行数を取得します */
   $row_cnt = maxdb_num_rows($result);

   printf("Result set has %d rows.¥n", $row_cnt);

   /* 結果セットを閉じます */
   maxdb_free_result($result);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Result set has 15 rows.
```

参考

- maxdb_affected_rows()
- maxdb_store_result()
- maxdb_use_result()
- maxdb_query()

---

maxdb_options

maxdb->options

(PECL maxdb:1.0-7.6.00.38)

maxdb->options — オプションを設定する

説明

手続き型

bool ***maxdb_options*** *(resource $link , int $option , mixed $value)*

オブジェクト指向型（メソッド）

maxdb
bool ***options*** *(int $option , mixed $value)*

maxdb_options() *は、 接続時の追加のオプションを設定し、接続の振る舞いを変更するために使用されます。*

この関数は、いくつかのオプションを設定するために複数回コールすることが可能です。

maxdb_options() *は、 maxdb_init() をコールした後、 maxdb_real_connect() がコールされるまでにコールしなければなりません。*

パラメータ option は設定したいオプションで、 value はオプションの値です。 オプションについての詳細な説明は、 » http://dev.mysql.com/doc/maxdb/ を参照ください。 パラメータ option は、以下のいずれかの値となります。

使用可能なオプション

名前	説明
MAXDB_COMPNAME	SQLDBC 実行環境の初期化に使用するコンポーネントの名前。
MAXDB_APPLICATION	データベースに接続するアプリケーション。
MAXDB_APPVERSION	アプリケーションのバージョン。
MAXDB_SQLMODE	SQL モード。
MAXDB_UNICODE	unicode (UCS2) クライアントからの接続の場合に TRUE、そうでない場合に FALSE。
MAXDB_TIMEOUT	データベースへの接続がシステムによって閉じられるまでの、 無通信時間の最大値。
MAXDB_ISOLATIONLEVEL	共有ロックおよび排他ロックのどちらがどのように要求/開放されるかを指定します。
MAXDB_PACKETCOUNT	接続に使用される、要求パケットの数。
MAXDB_STATEMENTCACHESIZE	再利用のために接続にキャッシュされるプリペアドステートメントの数。
MAXDB_CURSORPREFIX	自動的に命名された結果テーブルに使用するプレフィックス。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

例

maxdb_real_connect() を参照ください。

参考

- *maxdb_init()*
- *maxdb_real_connect()*

---

maxdb_param_count

(PECL maxdb:1.0)

maxdb_param_count — maxdb_stmt_param_count() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_param_count().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_ping

maxdb->ping

(PECL maxdb:1.0-7.6.00.38)

maxdb->ping — サーバとの接続を確認し、接続が確立されていない場合は再接続を試みる

説明

手続き型

bool ***maxdb_ping*** *(resource $link)*

オブジェクト指向型（メソッド）

maxdb
bool ***ping*** *(void)*

サーバとの接続が正常に動作しているかどうかを調べます。 もし接続が確立できていない場合、グローバルオプション maxdb.reconnect が有効になっていれば、 自動的に再接続が試みられます。

この関数は、クライアント側で長時間アイドル状態が続いた場合などに、 サーバが接続を閉じてしまったかどうかを調べ、 必要なら再度接続するために使用されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* サーバが動作中かどうかを調べます */
if ($maxdb->ping()) {
   printf ("接続は有効です!\n");
} else {
   printf ("エラー: %s\n", $maxdb->error);
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* サーバが動作中かどうかを調べます */
if (maxdb_ping($link)) {
   printf ("接続は有効です!\n");
} else {
   printf ("エラー: %s\n", $maxdb->error);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
接続は有効です!
```

---

maxdb_prepare

maxdb->prepare

(PECL maxdb:1.0-7.6.00.38)

maxdb->prepare — 後で実行するための SQL 文を準備する

説明

手続き型

```
resource maxdb_prepare ( resource $link , string $query )
```

オブジェクト指向型 (メソッド)

```
stmt
resource prepare ( string $query )
```

maxdb_prepare() は、 後で実行するための SQLクエリをヌル終端の文字列で準備し、 今後のステートメントに対する操作に使用するためのハンドルを返します。 クエリは、単一の SQL 文である必要があります。

> **注意**: 文の最後に、セミコロンや ¥g をつけてはいけません。

query では、SQL 文の中に、 ひとつあるいは複数のパラメータマーカを含めることが可能です。 適切な場所にクエスチョンマーク (?) を埋め込みます。

> **注意**: マーカは、SQL 文の中の適切な箇所にある場合にのみ有効です。 例えば INSERT 文の VALUES() リスト (その行のカラムの値を指定する) あるいは WHERE 句でカラムの値と 比較する条件を指定する場合などが有効です。
> しかし、識別子 (テーブル名やカラム名)、SELECT 文が返すカラム名の一覧、 あるいは (例えば = のような) 二項演算子の両側などに

マーカを指定することはできません。最後の制限が必要なのは、 この場合にパラメータの型が決定できなくなるからです。 一般的に、パラメータはデータ操作言語 (DML) 文で使用し、 データ定義言語 (DDL) 文では使用しません。

パラメータマーカは、文を実行したり行を取得したりする前に、必ず maxdb_stmt_bind_param() や maxdb_stmt_bind_result() でアプリケーションの変数にバインドしなければなりません。

返り値

maxdb_prepare() は、ステートメントリソースを返します。 エラーが発生した場合は **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$city = "Rosemont";

/* プリペアドステートメントを作成します */
if ($stmt = $maxdb->prepare("SELECT state FROM hotel.city WHERE name=?")) {

   /* マーカにパラメータをバインドします */
   $stmt->bind_param("s", $city);

   /* クエリを実行します */
   $stmt->execute();

   /* 結果変数にバインドします */
   $stmt->bind_result($district);

   /* 値を取得します */
   $stmt->fetch();

   printf("%s is in district %s¥n", $city, $district);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$city = "Rosemont";

/* プリペアドステートメントを作成します */
if ($stmt = maxdb_prepare($link, "SELECT state FROM hotel.city WHERE name=?")) {

   /* マーカにパラメータをバインドします */
   maxdb_stmt_bind_param($stmt, "s", $city);

   /* クエリを実行します */
   maxdb_stmt_execute($stmt);

   /* 結果変数にバインドします */
   maxdb_stmt_bind_result($stmt, $district);

   /* 値を取得します */
   maxdb_stmt_fetch($stmt);

   printf("%s is in district %s¥n", $city, $district);

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Rosemont is in district IL
```

参考

- maxdb_stmt_execute()

- maxdb_stmt_fetch()
- maxdb_stmt_bind_param()
- maxdb_stmt_bind_result()
- maxdb_stmt_close()

---

maxdb_query

maxdb->query

(PECL maxdb:1.0-7.6.00.38)

maxdb->query — データベース上でクエリを実行する

説明

手続き型

```
mixed maxdb_query ( resource $link , string $query [, int $resultmode ] )
```

オブジェクト指向型（メソッド）

```
maxdb
mixed query ( string $query )
```

maxdb_query() 関数は、link が表すデータベースに対するクエリの実行を単純化するために使用されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 SELECT, SHOW, DESCRIBE あるいは EXPLAIN の場合は、**maxdb_query()** は結果リソースを返します。

例

Example#1 オブジェクト指向型

```php
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* テーブルの作成の場合は結果セットを返しません */
if ($maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city") === TRUE) {
   printf("mycity テーブルの作成に成功しました。\n");
}

/* select クエリは結果セットを返します */
if ($result = $maxdb->query("SELECT name FROM hotel.city")) {
   printf("select が %d 行のデータを返しました。\n", maxdb_num_rows($result));

   /* 結果セットを開放します */
   $result->close();
}

/* 大量のデータを取得しなければならない場合は MAXDB_USE_RESULT を使用します */
if ($result = $maxdb->query("SELECT * FROM hotel.city", MAXDB_USE_RESULT)) {
   $result->close();
}

$maxdb->close();
?>
```

Example#2 手続き型

```php
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* テーブルの作成の場合は結果セットを返しません */
if (maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city") === TRUE) {
   printf("mycity テーブルの作成に成功しました。\n");
}

/* select クエリは結果セットを返します */
if ($result = maxdb_query($link, "SELECT name FROM hotel.city")) {
   printf("select が %d 行のデータを返しました。\n", maxdb_num_rows($result));

   /* 結果セットを開放します */
   maxdb_free_result($result);
}

/* 大量のデータを取得しなければならない場合は MAXDB_USE_RESULT を使用します */
if ($result = maxdb_query($link, "SELECT * FROM hotel.city", MAXDB_USE_RESULT)) {
```

```
    maxdb_free_result($result);
}

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
mycity テーブルの作成に成功しました。
select が 25 行のデータを返しました。
```

参考

- maxdb_real_query()
- maxdb_multi_query()
- maxdb_free_result()

---

maxdb_real_connect

maxdb->real_connect

(PECL maxdb:1.0-7.6.00.38)

maxdb->real_connect — MaxDB サーバへの接続をオープンする

説明

手続き型

```
bool maxdb_real_connect ( resource $link [, string $hostname [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket ]]]]]] )
```

オブジェクト指向型（メソッド）

```
maxdb
bool real_connect ([ string $hostname [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket ]]]]] ])
```

maxdb_real_connect() は、 hostname で稼動中の MaxDB データベースエンジンに対して、 接続の確立を試みます。

この関数は、maxdb_connect() とは以下の点で違います。

- **maxdb_real_connect()** には、 maxdb_init() で作成した有効なリソースが必要です。
- maxdb_options() 関数により、 接続の際のさまざまなオプションを設定することが可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php

/* 接続オブジェクトを作成しますが、まだ接続していません */
$maxdb  = maxdb_init();

/* 接続オプションを設定します */
$maxdb->options(MAXDB_UNICODE, "FALSE");
$maxdb->options(MAXDB_TIMEOUT, 5);

/* サーバに接続します */
$maxdb->real_connect('localhost', 'MONA', 'RED', 'DEMODB');

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

printf ("接続:  %s\n.", $maxdb->host_info);

$maxdb->close();
?>
```

Example#2 手続き型

```
<?php

/* 接続オブジェクトを作成しますが、まだ接続していません */
$link  = maxdb_init();

/* 接続オプションを設定します */
maxdb_options($link, MAXDB_UNICODE, "FALSE");
maxdb_options($link, MAXDB_TIMEOUT, 5);
```

```
/* サーバに接続します */
maxdb_real_connect($link, 'localhost', 'MONA', 'RED', 'DEMODB');

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

printf ("接続:  %s\n.", maxdb_get_host_info($link));

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
接続:  localhost  <...>
```

参考

- maxdb_connect()
- maxdb_init()
- maxdb_options()
- maxdb_ssl_set()
- maxdb_close()

---

maxdb_real_escape_string

maxdb->real_escape_string

(PECL maxdb:1.0-7.6.00.38)

maxdb->real_escape_string — 現在の接続の文字セットを考慮したうえで、 SQL 文で使用される文字列中の特殊文字をエスケープする

説明

手続き型

```
string maxdb_real_escape_string ( resource $link , string $escapestr )
```

オブジェクト指向型（メソッド）

```
maxdb
string real_escape_sring ( string $escapestr )
```

この関数は、SQL 文で使用するために、SQL で使用可能な文字列を作成するために使用します。 文字列 escapestr が、エスケープされた SQL 文字列にエンコードされます。その際、接続の現在の文字セットを考慮します。

エンコードされる文字は ', " です。

返り値

エスケープされた文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query("CREATE  TABLE  temp.mycity  LIKE  hotel.city");

$city = "'s  Hertogenbosch";

/* $city をエスケープしていないため、このクエリは失敗します */
if (!$maxdb->query("INSERT  into  temp.mycity  VALUES  ('11111','$city','NY')")) {
   printf("エラー:  %s\n", $maxdb->sqlstate);
}

$city = $maxdb->real_escape_string($city);

/* このクエリは、$city をエスケープしているので動作します */
if ($maxdb->query("INSERT  into  temp.mycity  VALUES  ('22222','$city','NY')")) {
   printf("%d  行挿入されました。\n", $maxdb->affected_rows);
}

$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

$city = "'s Hertogenbosch";

/* $city をエスケープしていないため、このクエリは失敗します */
if (!maxdb_query($link, "INSERT into temp.mycity VALUES ('11111','$city','NY')")) {
   printf("エラー: %s\n", maxdb_sqlstate($link));
}

$city = maxdb_real_escape_string($link, $city);

/* このクエリは、$city をエスケープしているので動作します */
if (maxdb_query($link, "INSERT into temp.mycity VALUES ('22222','$city','NY')")) {
   printf("%d 行挿入されました。\n", $maxdb->affected_rows);
}

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_query(): -5016 POS(43) Missing delimiter: ) <...>
エラー: 42000
1 行挿入されました。
```

参考

- maxdb_character_set_name()

---

maxdb_real_query

maxdb->real_query

(PECL maxdb:1.0-7.6.00.38)

maxdb->real_query — SQL クエリを実行する

説明

手続き型

bool **maxdb_real_query** (resource $link , string $query)

オブジェクト指向型（メソッド）

maxdb
bool **real_query** (string $query)

maxdb_real_query() は、機能的に maxdb_query() とまったく同じです。

> **注意**: 指定したクエリが結果セットを返すかどうかを調べるには、 maxdb_field_count() を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- maxdb_query()
- maxdb_store_result()
- maxdb_use_result()

---

maxdb_report

(PECL maxdb:1.0)

maxdb_report — 内部のレポート関数を有効あるいは無効にする

説明

bool **maxdb_report** (int $flags)

例

Example#1 手続き型

```
<?php
/* レポートを有効にします */
maxdb_report(MAXDB_REPORT_ERROR);

$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* このクエリはエラーを報告します */
$result = maxdb_query($link,"SELECT Name FROM Nonexistingtable WHERE population > 50000");

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_query(): -4004 POS(18) Unknown table name:NONEXISTINGTABLE <...>
```

---

maxdb_rollback

maxdb->rollback

(PECL maxdb:1.0-7.6.00.38)

maxdb->rollback — 現在のトランザクションをロールバックする

説明

```
bool maxdb_rollback ( resource $link )
maxdb
bool rollback ( void )
```

link パラメータで指定したデータベース接続について、 現在のトランザクションをロールバックします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 自動コミットを無効にします */
$maxdb->autocommit(FALSE);

$maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city");
$maxdb->query("INSERT INTO temp.mycity SELECT * FROM hotel.city");

/* insert をコミットします */
$maxdb->commit();

/* すべての行を削除します */
$maxdb->query("DELETE FROM temp.mycity");

if ($result = $maxdb->query("SELECT COUNT(*) FROM temp.mycity")) {
   $row = $result->fetch_row();
   printf("%d rows in table mycity.¥n", $row[0]);
   /* 結果を開放します */
   $result->close();
}

/* ロールバックします */
$maxdb->rollback();

if ($result = $maxdb->query("SELECT COUNT(*) FROM temp.mycity")) {
   $row = $result->fetch_row();
   printf("%d rows in table mycity (after rollback).¥n", $row[0]);
   /* 結果を開放します */
   $result->close();
}

/* myCity テーブルを削除します */
$maxdb->query("DROP TABLE temp.mycity");
```

```
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 自動コミットを無効にします */
maxdb_autocommit($link, FALSE);

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");

/* commit insert */
maxdb_commit($link);

/* delete all rows */
maxdb_query($link, "DELETE FROM temp.mycity");

if ($result = maxdb_query($link, "SELECT COUNT(*) FROM temp.mycity")) {
   $row = maxdb_fetch_row($result);
   printf("%d rows in table mycity.¥n", $row[0]);
   /* 結果を開放します */
   maxdb_free_result($result);
}

/* ロールバックします */
maxdb_rollback($link);

if ($result = maxdb_query($link, "SELECT COUNT(*) FROM temp.mycity")) {
   $row = maxdb_fetch_row($result);
   printf("%d rows in table mycity (after rollback).¥n", $row[0]);
   /* 結果を開放します */
   maxdb_free_result($result);
}

/* myCity テーブルを削除します */
maxdb_query($link, "DROP TABLE temp.mycity");

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
0 rows in table mycity.
25 rows in table mycity (after rollback).
```

参考

- maxdb_commit()
- maxdb_autocommit()

---

maxdb_rpl_parse_enabled

(PECL maxdb:1.0-7.6.00.38)

maxdb_rpl_parse_enabled — RPL のパースが有効かどうかを調べる

説明

```
int maxdb_rpl_parse_enabled ( resource $link )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_rpl_probe

(PECL maxdb:1.0-7.6.00.38)

maxdb_rpl_probe — RPL を調べる

説明

```
bool maxdb_rpl_probe ( resource $link )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_rpl_query_type

maxdb->rpl_query_type

(PECL maxdb:1.0-7.6.00.38)

maxdb->rpl_query_type — RPL クエリ型を返す

説明

int **maxdb_rpl_query_type** (resource $link)

オブジェクト指向型（メソッド）

maxdb
int **rpl_query_type** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_select_db

maxdb->select_db

(PECL maxdb:1.0-7.6.00.38)

maxdb->select_db — データベースクエリ用のデフォルトデータベースを選択する

説明

bool **maxdb_select_db** (resource $link , string $dbname)

maxdb_select_db() 関数は、 link が表すデータベース接続に対して クエリを実行する際に使用するデフォルトのデータベースを （dbname で指定したものに）選択します。

> **注意**: この関数は、接続のデフォルトデータベースを変更する場合にのみ使用すべきです。 デフォルトのデータベースは、maxdb_connect() の 4 番目のパラメータで指定することができます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 現在のデフォルトデータベースの名前を返します */
if ($result = $maxdb->query("SELECT SERVERDB FROM USERS WHERE USERNAME='MONA'")) {
   $row = $result->fetch_row();
   printf("デフォルトデータベースは %s です。\n", $row[0]);
   $result->close();
}

/* 存在しない db に変更します */
$maxdb->select_db("XXXXXXXX");

/* 現在のデフォルトデータベースの名前を返します */
if ($result = $maxdb->query("SELECT SERVERDB FROM USERS WHERE USERNAME='MONA'")) {
   $row = $result->fetch_row();
   printf("デフォルトデータベースは %s です。\n", $row[0]);
   $result->close();
}

$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 現在のデフォルトデータベースの名前を返します */
if ($result = maxdb_query($link, "SELECT SERVERDB FROM USERS WHERE USERNAME='MONA'")) {
   $row = maxdb_fetch_row($result);
```

```
   printf("デフォルトデータベースは %s です。\n", $row[0]);
   maxdb_free_result($result);
}

/* 存在しない db に変更します */
maxdb_select_db($link, "XXXXXXXX");

/* 現在のデフォルトデータベースの名前を返します */
if ($result = maxdb_query($link, "SELECT SERVERDB FROM USERS WHERE USERNAME='MONA'")) {
   $row = maxdb_fetch_row($result);
   printf("デフォルトデータベースは %s です。\n", $row[0]);
   maxdb_free_result($result);
}

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
デフォルトデータベースは <...> です。

Warning: maxdb_select_db(): -10709 Connection failed (RTE:database not running) <...>

Warning: maxdb_query(): -10821 Session not connected [] <...>

Warning: maxdb_close(): -10821 Session not connected [] <...>
```

参考

- maxdb_connect()
- maxdb_real_connect()

---

maxdb_send_long_data

(PECL maxdb:1.0-7.6.00.38)

maxdb_send_long_data — maxdb_stmt_send_long_data() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_stmt_send_long_data().

この関数エイリアスは非推奨であり、下位互換性維持のために残されています。今後、PHP から削除される可能性がありますので、この関数を使用しないことを推奨します。

---

maxdb_send_query

maxdb->send_query

(PECL maxdb:1.0-7.6.00.38)

maxdb->send_query — クエリを送信する

説明

bool **maxdb_send_query** (resource $link , string $query)

オブジェクト指向型 (メソッド)

maxdb
bool **send_query** (string $query)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_server_end

(PECL maxdb:1.0-7.6.00.38)

maxdb_server_end — 埋め込みサーバをシャットダウンする

説明

void **maxdb_server_end** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_server_init

(PECL maxdb:1.0-7.6.00.38)

maxdb_server_init — 埋め込みサーバを初期化する

説明

bool **maxdb_server_init** ([array $server [, array $groups]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_set_opt

(PECL maxdb:1.0)

maxdb_set_opt — maxdb_options() のエイリアス

説明

この関数は次の関数のエイリアスです。 maxdb_options().

---

maxdb_sqlstate

maxdb->sqlstate

(PECL maxdb:1.0-7.6.00.38)

maxdb->sqlstate — 直近の MaxDB 操作の SQLSTATE エラーを返します

説明

手続き型

string **maxdb_sqlstate** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
string$sqlstate;

直近のエラーの SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 つの文字で構成されています。'00000' は、エラーが発生しなかったことを意味します。 コードの内容は ANSI SQL および ODBC で指定されています。

> **注意**: 今のところ、すべての MaxDB エラーが SQLSTATE に関連付けられているわけではないことに注意しましょう。 関連付けられていないエラーについては、 HY000 （一般的なエラー）が使用されます。

返り値

直近のエラーについての SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 つの文字で構成されています。'00000' は、エラーが発生しなかったことを意味します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* City テーブルは既に存在するので、エラーとなります */
if (!$maxdb->query("CREATE TABLE hotel.city (ID INT, Name VARCHAR(30))")) {
   printf("Error - SQLSTATE %s.¥n", $maxdb->sqlstate);
}

$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* City テーブルは既に存在するので、エラーとなります */
```

```
if (!maxdb_query($link, "CREATE TABLE hotel.city (ID INT, Name VARCHAR(30))")) {
   printf("Error - SQLSTATE %s.¥n", maxdb_sqlstate($link));
}

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_query(): -6000 POS(20) Duplicate table name:CITY [I6000] <...>
Error - SQLSTATE I6000.
```

参考

- maxdb_errno()
- maxdb_error()

---

maxdb_ssl_set

maxdb->ssl_set

(PECL maxdb:1.0-7.6.00.38)

maxdb->ssl_set — SSL を使用したセキュアな接続を確立するために使用する

説明

手続き型

bool **maxdb_ssl_set** (resource $link , string $key , string $cert , string $ca , string $capath , string $cipher)

オブジェクト指向型（メソッド）

maxdb
bool **ssl_set** (string $key , string $cert , string $ca , string $capath , string $cipher)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_stat

maxdb->stat

(PECL maxdb:1.0-7.6.00.38)

maxdb->stat — 現在のシステム状態を取得する

説明

手続き型

string **maxdb_stat** (resource $link)

オブジェクト指向型（メソッド）

maxdb
string **maxdb->stat** (void)

maxdb_stat() は、稼動中の MaxDB サーバに関する情報を含む文字列を返します。

返り値

サーバの状態を説明する文字列、あるいはエラーが発生した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

printf ("システムの状態: %s\n", $maxdb->stat());

$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

printf("システムの状態: %s\n", maxdb_stat($link));

maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
システムの状態: Kernel      7<...>
```

参考

- maxdb_get_server_info()

---

maxdb_stmt_affected_rows

maxdb_stmt->affected_rows

(PECL maxdb:1.0-7.6.00.38)

maxdb_stmt->affected_rows — 直近のステートメントによって変更、削除あるいは挿入された行の数を返す

説明

手続き型

int **maxdb_stmt_affected_rows** (resource $stmt)

オブジェクト指向型（プロパティ）

stmt
int$affected_rows;

maxdb_stmt_affected_rows() は、 INSERT、UPDATE あるいは DELETE クエリによって変更された行の数を返します。 直近のクエリが無効だった場合、あるいは行数が取得できなかった場合は、 この関数は -1 を返します。

返り値

ゼロより大きい整数の場合は、変更された行数あるいは取得した行数を表します。 ゼロの場合は、UPDATE/DELETE 文で 1 行も更新されなかったか、クエリの WHERE 句にマッチする行がなかった、 あるいはクエリがまだ実行されていないことを表します。 -1 は、クエリがエラーを返したか行数が取得できなかったことを表します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 一時テーブルを作成します */
$maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city");

$query = "INSERT INTO temp.mycity SELECT * FROM hotel.city WHERE state LIKE ?";

/* ステートメントを準備します */
if ($stmt = $maxdb->prepare($query)) {

   /* 変数をプレースホルダにバインドします */
   $code = 'N%';
   $stmt->bind_param("s", $code);

   /* ステートメントを実行します */
   $stmt->execute();

   printf("挿入された行数: %d\n", $stmt->affected_rows);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* 一時テーブルを作成します */
maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

$query = "INSERT INTO temp.mycity SELECT * FROM hotel.city WHERE state LIKE ?";

/* ステートメントを準備します */
if ($stmt = maxdb_prepare($link, $query)) {

   /* 変数をプレースホルダにバインドします */
   $code = 'N%';
   maxdb_stmt_bind_param($stmt, "s", $code);

   /* ステートメントを実行します */
   maxdb_stmt_execute($stmt);

   printf("挿入された行数: %d\n", maxdb_stmt_affected_rows($stmt));

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
挿入された行数: 4
```

参考

- maxdb_stmt_num_rows()
- maxdb_prepare()

---

maxdb_stmt_bind_param

stmt->bind_param

(No version information available, might be only in CVS)

stmt->bind_param — プリペアドステートメントに、変数をパラメータとしてバインドする

説明

手続き型

```
bool maxdb_stmt_bind_param ( resource $stmt , string $types , mixed &$var1 [, mixed &$... ] )
```

オブジェクト指向型（メソッド）

```
stmt
bool bind_param ( string $types , mixed &$var1 [, mixed &$... ] )
```

手続き型（拡張構文）

```
bool maxdb_stmt_bind_param ( resource $stmt , string $types , array &$var )
```

オブジェクト指向型（メソッド）（拡張構文）

```
stmt
bool bind_param ( string $types , array &$var )
```

maxdb_stmt_bind_param() は、 maxdb_prepare() に渡された SQL 文のパラメータマーカに、 変数をバインドするために使用されます。 文字列 types にはひとつあるいは複数の文字が含まれ、 これによって対応するバインド変数の型を指定します。

maxdb_stmt_bind_param() の拡張構文では、 パラメータを、PHP の変数リストではなくひとつの配列として渡せるようになります。 **maxdb_stmt_bind_param()** のコール前に配列変数が使用されていない場合は、空の配列として初期化されます。 **maxdb_stmt_bind_param()** の拡張構文の使用法については、 例を参照ください。

SELECT INTO SQL 文の変数は、**maxdb_stmt_bind_param()** でバインドすることも可能です。データベースプロシージャのパラメータについては **maxdb_stmt_bind_param()** を使用してバインドできます。 このような場合の **maxdb_stmt_bind_param()** の使用法については、例を参照ください。

SQL 文の INTO 変数にバインドする変数が事前に使用されていた場合は、 その変数の内容は SELECT INTO 文のデータで上書きされます。 **maxdb_stmt_bind_param()** をコールした後は、 この変数への参照は無効になります。

データベースプロシージャの INOUT パラメータの場合、 バインドされた INOUT 変数の内容は、データベースプロシージャの出力で 上書きされます。 **maxdb_stmt_bind_param()** をコールした後は、 この変数への参照は無効になります。

型指定文字

文字	説明
i	対応する変数は integer 型です
d	対応する変数は double 型です
s	対応する変数は string 型です
b	対応する変数は blob で、一括して送信されます

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb('localhost', 'MONA', 'RED', 'DEMODB');

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query ("CREATE TABLE temp.mycity LIKE hotel.city");
$maxdb->query ("INSERT INTO temp.mycity SELECT * FROM hotel.city");

$stmt = $maxdb->prepare("INSERT INTO temp.mycity VALUES (?, ?, ?)");
$stmt->bind_param('sss', $zip, $name, $state);

$zip = '11111';
$name = 'Georgetown';
$state = 'NY';

/* プリペアドステートメントを実行します */
$stmt->execute();

printf("%d 行挿入されました。\n", $stmt->affected_rows);

/* ステートメントおよび接続を閉じます */
$stmt->close();

/* CountryLanguage テーブルを掃除します */
$maxdb->query("DELETE FROM temp.mycity WHERE name='Georgetown'");
printf("%d 行削除されました。\n", $maxdb->affected_rows);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_query ($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query ($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");

$stmt = maxdb_prepare($link, "INSERT INTO temp.mycity VALUES (?, ?, ?)");
maxdb_stmt_bind_param($stmt, 'sss', $zip, $name, $state);

$zip = '11111';
$name = 'Georgetown';
$state = 'NY';

/* プリペアドステートメントを実行します */
maxdb_stmt_execute($stmt);

printf("%d 行挿入されました。\n", maxdb_stmt_affected_rows($stmt));

/* ステートメントおよび接続を閉じます */
maxdb_stmt_close($stmt);

/* CountryLanguage テーブルを掃除します */
maxdb_query($link, "DELETE FROM temp.mycity WHERE name='Georgetown'");
printf("%d 行削除されました。\n", maxdb_affected_rows($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
1 行挿入されました。
1 行削除されました。
```

Example#3 手続き型 (SELECT INTO)

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* SQL クエリを実行します */
$stmt = maxdb_prepare ($link, "SELECT price INTO ? FROM hotel.room where hno = ? and type = ?");
if (!$stmt) {
  printf ("Prepare failed: %s¥n", maxdb_error($link));
}

$hno = "50";
$rtype = "suite";

maxdb_stmt_bind_param($stmt, 'dss', $price, $hno, $rtype);
maxdb_stmt_execute($stmt);

printf ("%f¥n", $price);

maxdb_stmt_close ($stmt);
?>
```

上の例の出力は、たとえば以下のようになります。

```
21.600000
```

Example#4 手続き型 (DB プロシージャ)

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_report (MAXDB_REPORT_OFF);
maxdb_query($link,"DROP DBPROC test_proc");
maxdb_report (MAXDB_REPORT_ERROR);

$query = "create dbproc test_proc (INOUT e_text char(72)) AS select * from SYSDBA.DUAL; fetch into :e_text;";

maxdb_query($link, $query);

/* SQL クエリを実行します */
$stmt = maxdb_prepare ($link, "CALL test_proc (?)");
if (!$stmt) {
  printf ("Prepare failed: %s¥n", maxdb_error($link));
}

maxdb_stmt_bind_param($stmt, 's', $result);
maxdb_stmt_execute($stmt);

printf ("%s¥n", $result);

maxdb_stmt_close ($stmt);
?>
```

上の例の出力は、たとえば以下のようになります。

```
a
```

Example#5 オブジェクト指向型 (拡張構文)

```
<?php
$maxdb = new maxdb('localhost', 'MONA', 'RED', 'DEMODB');

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query ("CREATE TABLE temp.mycity LIKE hotel.city");
$maxdb->query ("INSERT INTO temp.mycity SELECT * FROM hotel.city");

$stmt = $maxdb->prepare("INSERT INTO temp.mycity VALUES (?, ?, ?)");

$arr = array();

$stmt->bind_param('iss', $arr);

$arr[0] = 11111;
$arr[1] = 'Georgetown';
$arr[2] = 'NY';
```

```
/* プリペアドステートメントを実行します */
$stmt->execute();

printf("%d 行挿入されました。\n", maxdb_stmt_affected_rows($stmt));

$arr[0] = 22222;
$arr[1] = 'New Orleans';
$arr[2] = 'LA';

/* プリペアドステートメントを実行します */
$stmt->execute();

printf("%d 行挿入されました。\n", $stmt->affected_rows);

/* ステートメントおよび接続を閉じます */
$stmt->close();

$result = $maxdb->query("SELECT * from temp.mycity WHERE zip = '11111' OR zip = '22222'");
if ($result) {
  while ($row = $result->fetch_row()) {
    printf ("%s %s %s¥n", $row[0], $row[1], $row[2]);
  }
}

/* CountryLanguage テーブルを掃除します */
$maxdb->query("DELETE FROM temp.mycity WHERE name='Georgetown'");
$maxdb->query("DELETE FROM temp.mycity WHERE name='New Orleans'");
printf("%d 行削除されました。\n", $maxdb->affected_rows);

/* 接続を閉じます */
$maxdb->close();
?>
```

上の例の出力は、たとえば以下のようになります。

```
1 行挿入されました。
1 行挿入されました。
11111 Georgetown NY
22222 New Orleans LA
2 行削除されました。
```

参考

- maxdb_stmt_bind_result()
- maxdb_stmt_execute()
- maxdb_stmt_fetch()
- maxdb_prepare()
- maxdb_stmt_send_long_data()
- maxdb_stmt_errno()
- maxdb_stmt_error()

---

maxdb_stmt_bind_result

stmt->bind_result

(No version information available, might be only in CVS)

stmt->bind_result — 結果を保存するために、変数をプリペアドステートメントにバインドする

説明

手続き型

bool **maxdb_stmt_bind_result** (resource $stmt , mixed &$var1 [, mixed &$...])

オブジェクト指向型（メソッド）

stmt
bool **bind_result** (mixed &$var1 [, mixed &$...])

maxdb_stmt_bind_result() は、 結果セット内のカラムを変数に関連付ける（バインドする）ために使用されます。 データを取得するために maxdb_stmt_fetch() をコールすると、MaxDB クライアント/サーバプロトコルが、 バインドされたカラムのデータを指定した変数 var1, ... に保存します。

> **注意**: すべてのカラムは、maxdb_stmt_fetch() のコール前にバインドされなければならないことに注意しましょう。 バインド変数は、カラムの型に応じて自動的に対応する PHP 型に変換されます。
> 結果セットが部分的に取得された後であっても、カラムはいつでも バインド/再バインドできます。新しいバインド内容は、次に maxdb_stmt_fetch() がコールされた時点で反映されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* ステートメントを準備します */
if ($stmt = $maxdb->prepare("SELECT zip, name FROM hotel.city ORDER BY name")) {
   $stmt->execute();

   /* 変数をプリペアドステートメントにバインドします */
   $stmt->bind_result($col1, $col2);

   /* 値を取得します */
   while ($stmt->fetch()) {
       printf("%s %s¥n", $col1, $col2);
   }

   /* ステートメントを閉じます */
   $stmt->close();
}
/* 接続を閉じます */
$maxdb->close();

?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (!$link) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* ステートメントを準備します */
if ($stmt = maxdb_prepare($link, "SELECT zip, name FROM hotel.city ORDER BY name")) {
   maxdb_stmt_execute($stmt);

   /* 変数をプリペアドステートメントにバインドします */
   maxdb_stmt_bind_result($stmt, $col1, $col2);

   /* 値を取得します */
   while (maxdb_stmt_fetch($stmt)) {
       printf("%s %s¥n", $col1, $col2);
   }

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
12203 Albany
60601 Chicago
60615 Chicago
45211 Cincinnati
33575 Clearwater
75243 Dallas
32018 Daytona Beach
33441 Deerfield Beach
48226 Detroit
90029 Hollywood
92714 Irvine
90804 Long Beach
11788 Long Island
90018 Los Angeles
70112 New Orleans
10019 New York
10580 New York
92262 Palm Springs
97213 Portland
60018 Rosemont
95054 Santa Clara
20903 Silver Spring
20005 Washington
20019 Washington
20037 Washington
```

参考

- *maxdb_stmt_bind_param()*
- *maxdb_stmt_execute()*
- *maxdb_stmt_fetch()*
- *maxdb_prepare()*

- maxdb_stmt_prepare()
- maxdb_stmt_init()
- maxdb_stmt_errno()
- maxdb_stmt_error()

---

maxdb_stmt_close_long_data

stmt->close_long_data

(No version information available, might be only in CVS)

stmt->close_long_data — maxdb_stmt_send_long_data() のシーケンスを終了する

説明

手続き型

bool **maxdb_stmt_close_long_data** (resource $stmt , int $param_nr)

オブジェクト指向型（メソッド）

maxdb_stmt
bool **maxdb_stmt->close_long_data** (void)

この関数は、maxdb_execute() で開始した後、 maxdb_stmt_send_long_data() のシーケンスの後にコールする必要があります。

param_nr は、 データの最後にどのパラメータを関連付けるかを示します。 パラメータは、0 番から順に番号がつけられます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- maxdb_prepare()
- maxdb_stmt_bind_param()

---

maxdb_stmt_close

maxdb_stmt->close

(PECL maxdb:1.0-7.6.00.38)

maxdb_stmt->close — プリペアドステートメントを閉じる

説明

手続き型

bool **maxdb_stmt_close** (resource $stmt)

オブジェクト指向型（メソッド）

maxdb_stmt
bool **maxdb_stmt->close** (void)

プリペアドステートメントを閉じます。 **maxdb_stmt_close()** は、 stmt が指すステートメントハンドルを開放します。 現在のステートメントに処理中あるいはまだ読み込まれていない結果がある場合、 この関数はそれらをキャンセルし、次のクエリを実行できるようにします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- maxdb_prepare()

---

maxdb_stmt_data_seek

stmt->data_seek

(No version information available, might be only in CVS)

stmt->data_seek — ステートメントの結果セットの、任意の行に移動する

説明

手続き型

```
bool maxdb_stmt_data_seek ( resource $statement , int $offset )
```

オブジェクト指向型（メソッド）

```
stmt
bool data_seek ( int $offset )
```

maxdb_stmt_data_seek() 関数は、statement が表すステートメント結果セットの中の offset で指定した任意のオフセットに結果ポインタを移動します。 offset は、ゼロから全行数マイナス 1 までの間 (0..maxdb_stmt_num_rows() - 1) である必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, zip FROM hotel.city ORDER BY name";
if ($stmt = $maxdb->prepare($query)) {

   /* クエリを実行します */
   $stmt->execute();

   /* 結果変数にバインドします */
   $stmt->bind_result($name, $code);

   /* 結果を保存します */
   $stmt->store_result();

   /* 行番号 5 に移動します */
   $stmt->data_seek(5);

   /* 値を取得します */
   $stmt->fetch();

   printf ("City: %s  Zip: %s¥n", $name, $code);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, zip FROM hotel.city ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* クエリを実行します */
   maxdb_stmt_execute($stmt);

   /* 結果変数にバインドします */
   maxdb_stmt_bind_result($stmt, $name, $code);

   /* 結果を保存します */
   maxdb_stmt_store_result($stmt);

   /* 行番号 5 に移動します */
   maxdb_stmt_data_seek($stmt, 5);

   /* 値を取得します */
   maxdb_stmt_fetch($stmt);

   printf ("City: %s  Zip: %s¥n", $name, $code);

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
City:  Dallas   Zip:  75243
```

参考

- maxdb_prepare()

---

maxdb_stmt_errno

maxdb_stmt->errno

(PECL maxdb:1.0-7.6.00.38)

maxdb_stmt->errno — 直近のステートメントコール時のエラーコードを返す

説明

手続き型

int **maxdb_stmt_errno** (resource $stmt)

オブジェクト指向型（プロパティ）

stmt
int$errno;

stmt で指定したステートメントについて、 **maxdb_stmt_errno()** は 直近に実行されたステートメントのエラーコードを返します。

> **注意**: 返される可能性のあるエラーコードについては、SQLDBC のドキュメント » http://dev.mysql.com/doc/maxdb/ を参照ください。

返り値

エラーコードの値を返します。エラーが発生しなかった場合はゼロを返します。

参考

- maxdb_stmt_error()
- maxdb_stmt_sqlstate()

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city");
$maxdb->query("INSERT INTO temp.mycity SELECT * FROM hotel.city");


$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = $maxdb->prepare($query)) {

   /* テーブルを削除します */
   $maxdb->query("DROP TABLE temp.mycity");

   /* クエリを実行します */
   $stmt->execute();

   printf("エラー: %d.\n", $stmt->errno);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
```

```
    exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");


$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* テーブルを削除します */
   maxdb_query($link, "DROP TABLE temp.mycity");

   /* クエリを実行します */
   maxdb_stmt_execute($stmt);

   printf("エラー: %d.\n", maxdb_stmt_errno($stmt));

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_stmt_execute(): -4004 POS(23) Unknown table name:MYCITY [42000] <...>
エラー: -4004.
```

参考

- maxdb_stmt_error()
- maxdb_stmt_sqlstate()

---

maxdb_stmt_error

maxdb_stmt->error

(PECL maxdb:1.0-7.6.00.38)

maxdb_stmt->error — 直近のステートメントコール時のエラー文字列を返す

説明

手続き型

string **maxdb_stmt_error** (resource $stmt)

オブジェクト指向型（プロパティ）

stmt
string$error;

stmt で指定したステートメントについて、 **maxdb_stmt_error()** は 直近に実行されたステートメントのエラーメッセージを返します。

返り値

エラーを説明する文字列を返します。エラーが発生しなかった場合は空の文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city");
$maxdb->query("INSERT INTO temp.mycity SELECT * FROM hotel.city");


$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = $maxdb->prepare($query)) {

   /* テーブルを削除します */
   $maxdb->query("DROP TABLE temp.mycity");

   /* クエリを実行します */
   $stmt->execute();
```

```
   printf("エラー:  %s.\n", $stmt->error);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");


$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* テーブルを削除します */
   maxdb_query($link, "DROP TABLE temp.mycity");

   /* クエリを実行します */
   maxdb_stmt_execute($stmt);

   printf("エラー:  %s.\n", maxdb_stmt_error($stmt));

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_stmt_execute(): -4004 POS(23) Unknown table name:MYCITY [42000] <...>
エラー: POS(23) Unknown table name:MYCITY.
```

参考

- maxdb_stmt_errno()
- maxdb_stmt_sqlstate()

---

maxdb_stmt_execute

stmt->execute

(No version information available, might be only in CVS)

stmt->execute — プリペアドクエリを実行する

説明

手続き型

bool **maxdb_stmt_execute** (resource $stmt)

オブジェクト指向型（メソッド）

stmt
bool **execute** (void)

maxdb_stmt_execute() 関数は、 stmt で表される、事前に maxdb_prepare() で準備したクエリを実行します。実行の際には、 すべてのパラメータマーカが適切な値に置き換えられます。

UPDATE、DELETE あるいは INSERT 文の場合は、 maxdb_stmt_affected_rows() 関数を使用すると 変更された行の総数が取得できます。同じく、 結果セットを返すクエリの場合は maxdb_fetch() 関数を使用して結果を取得できます。

> **注意**: **maxdb_stmt_execute()** を使用する際は、 他のクエリを実行する前に必ず maxdb_fetch() でデータを取得しなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city");

/* insert 文を準備します */
$query = "INSERT INTO temp.mycity (zip, name, state) VALUES (?,?,?)";
$stmt = $maxdb->prepare($query);

$stmt->bind_param("sss", $val1, $val2, $val3);

$val1 = '11111';
$val2 = 'Georgetown';
$val3 = 'NY';

/* ステートメントを実行します */
$stmt->execute();

$val1 = '22222';
$val2 = 'Hubbatown';
$val3 = 'CA';

/* ステートメントを実行します */
$stmt->execute();

/* ステートメントを閉じます */
$stmt->close();

/* myCity からすべての行を取得します */
$query = "SELECT zip, name, state FROM temp.mycity";
if ($result = $maxdb->query($query)) {
    while ($row = $result->fetch_row()) {
        printf("%s (%s,%s)¥n", $row[0], $row[1], $row[2]);
    }
   /* 結果セットを開放します */
   $result->close();
}

/* テーブルを削除します */
$maxdb->query("DROP TABLE temp.mycity");

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

/* insert 文を準備します */
$query = "INSERT INTO temp.mycity (zip, name, state) VALUES (?,?,?)";
$stmt = maxdb_prepare($link, $query);

maxdb_stmt_bind_param($stmt, "sss", $val1, $val2, $val3);

$val1 = '11111';
$val2 = 'Georgetown';
$val3 = 'NY';

/* ステートメントを実行します */
maxdb_stmt_execute($stmt);

$val1 = '22222';
$val2 = 'Hubbatown';
$val3 = 'CA';

/* ステートメントを実行します */
maxdb_stmt_execute($stmt);

/* ステートメントを閉じます */
maxdb_stmt_close($stmt);

/* myCity からすべての行を取得します */
$query = "SELECT zip, name, state FROM temp.mycity";
if ($result = maxdb_query($link, $query)) {
    while ($row = maxdb_fetch_row($result)) {
        printf("%s (%s,%s)¥n", $row[0], $row[1], $row[2]);
    }
   /* 結果セットを開放します */
   maxdb_free_result($result);
}

/* テーブルを削除します */
maxdb_query($link, "DROP TABLE temp.mycity");
```

```
/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
11111  (Georgetown,NY)
22222  (Hubbatown,CA)
```

参考

- maxdb_prepare()
- maxdb_stmt_bind_param()

---

maxdb_stmt_fetch

stmt->fetch

(No version information available, might be only in CVS)

stmt->fetch — プリペアドステートメントの結果を取得し、バインド変数に格納する

説明

手続き型

bool **maxdb_stmt_fetch** (resource $stmt)

オブジェクト指向型（メソッド）

stmt
bool **fetch** (void)

maxdb_stmt_fetch() は、 maxdb_stmt_bind_result() でバインドした変数を使用して、 行のデータを返します。

> **注意**: **maxdb_stmt_fetch()** をコールする前に、 すべてのカラムがアプリケーションによってバインドされている必要があることに注意しましょう。

返り値

返り値

値	説明
TRUE	成功。データが取得されました。
FALSE	エラーが発生しました。
NULL	行/データが、もう存在しません。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT zip, name FROM hotel.city ORDER by name";

if ($stmt = $maxdb->prepare($query)) {

   /* ステートメントを実行します */
   $stmt->execute();

   /* 結果変数にバインドします */
   $stmt->bind_result($name, $code);

   /* 値を取得します */
   while ($stmt->fetch()) {
       printf ("%s  (%s)¥n", $name, $code);
   }

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT zip, name FROM hotel.city ORDER by name";

if ($stmt = maxdb_prepare($link, $query)) {

   /* ステートメントを実行します */
   maxdb_stmt_execute($stmt);

   /* 結果変数にバインドします */
   maxdb_stmt_bind_result($stmt, $name, $code);

   /* 値を取得します */
   while (maxdb_stmt_fetch($stmt)) {
       printf ("%s (%s)¥n", $name, $code);
   }

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
12203 (Albany)
60601 (Chicago)
60615 (Chicago)
45211 (Cincinnati)
33575 (Clearwater)
75243 (Dallas)
32018 (Daytona Beach)
33441 (Deerfield Beach)
48226 (Detroit)
90029 (Hollywood)
92714 (Irvine)
90804 (Long Beach)
11788 (Long Island)
90018 (Los Angeles)
70112 (New Orleans)
10019 (New York)
10580 (New York)
92262 (Palm Springs)
97213 (Portland)
60018 (Rosemont)
95054 (Santa Clara)
20903 (Silver Spring)
20005 (Washington)
20019 (Washington)
20037 (Washington)
```

参考

- maxdb_prepare()
- maxdb_stmt_errno()
- maxdb_stmt_error()
- maxdb_stmt_bind_result()

---

maxdb_stmt_free_result

stmt->free_result

(No version information available, might be only in CVS)

stmt->free_result — 指定したステートメントハンドルの結果を保存しているメモリを開放する

説明

手続き型

void **maxdb_stmt_free_result** (resource $stmt)

オブジェクト指向型（メソッド）

stmt
void **free_result** (void)

maxdb_stmt_free_result() 関数は、 stmt パラメータで指定した結果メモリ（これは maxdb_stmt_store_result() によって確保されたものです）を開放します。

返り値

この関数は、なにも値を返しません。

参考

- maxdb_stmt_store_result()

---

maxdb_stmt_init

maxdb->stmt_init

(PECL maxdb:1.0-7.6.00.38)

maxdb->stmt_init — ステートメントを初期化し、maxdb_stmt_prepare で使用するリソースを返す

説明

手続き型

resource **maxdb_stmt_init** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
object **stmt_init** (void)

maxdb_stmt_prepare() で使用するステートメントリソースを 確保して初期化します。

注意: maxdb_stmt_prepare() がコールされるまでは、 これ以降のすべての maxdb_stmt 関数のコールは失敗します。

返り値

リソースを返します。

参考

- maxdb_stmt_prepare()

---

maxdb_stmt_num_rows

stmt->num_rows

(No version information available, might be only in CVS)

stmt->num_rows — ステートメントの結果セットの行数を返す

説明

手続き型

int **maxdb_stmt_num_rows** (resource $stmt)

オブジェクト指向型（プロパティ）

stmt
int$num_rows;

結果セット内の行の数を返します。

返り値

結果セット内の行の数を表す整数値を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT zip, name FROM hotel.city ORDER BY name";
if ($stmt = $maxdb->prepare($query)) {
```

```
   /* クエリを実行します */
   $stmt->execute();

   /* 結果を保存します */
   $stmt->store_result();

   printf("行数: %d.\n", $stmt->num_rows);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT zip, name FROM hotel.city ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* クエリを実行します */
   maxdb_stmt_execute($stmt);

   /* 結果を保存します */
   maxdb_stmt_store_result($stmt);

   printf("行数: %d.\n", maxdb_stmt_num_rows($stmt));

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
行数: 25.
```

参考

- maxdb_stmt_affected_rows()
- maxdb_prepare()
- maxdb_stmt_store_result()

---

maxdb_stmt_param_count

stmt->param_count

(No version information available, might be only in CVS)

stmt->param_count — 指定したステートメントのパラメータ数を返す

説明

手続き型

int **maxdb_stmt_param_count** (resource $stmt)

オブジェクト指向型（プロパティ）

stmt
int$param_count;

maxdb_stmt_param_count() は、 プリペアドステートメント内に存在するパラメータマーカの数を返します。

返り値

パラメータ数を表す整数値を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb  = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

if ($stmt  = $maxdb->prepare("SELECT  name  FROM  hotel.city  WHERE  name=?  OR  state=?")) {

   $marker  = $stmt->param_count;
   printf("ステートメントのパラメータ数は  %d  です。\n", $marker);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link  = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました:  %s\n", maxdb_connect_error());
    exit();
}

if ($stmt  = maxdb_prepare($link, "SELECT  name  FROM  hotel.city  WHERE  name=?  OR  state=?")) {

   $marker  = maxdb_stmt_param_count($stmt);
   printf("ステートメントのパラメータ数は  %d  です。\n", $marker);

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
ステートメントのパラメータ数は 2 です。
```

参考

- *maxdb_prepare()*

---

maxdb_stmt_prepare

stmt->prepare

(No version information available, might be only in CVS)

stmt->prepare — 後で実行するための SQL 文を準備する

説明

手続き型

*bool **maxdb_stmt_prepare** (resource $stmt , string $query)*

オブジェクト指向型（メソッド）

stmt
*mixed **prepare** (string $query)*

***maxdb_stmt_prepare()** は、 後で実行するための SQLクエリをヌル終端の文字列で準備します。 ステートメントのリソースは、maxdb_stmt_init() で確保しなければなりません。 クエリは、単一の SQL 文である必要があります。*

> ***注意**: 文の最後に、セミコロンや ¥g をつけてはいけません。*

query では、SQL 文の中に、 ひとつあるいは複数のパラメータマーカを含めることが可能です。 適切な場所にクエスチョンマーク (?) を埋め込みます。

> ***注意**: マーカは、SQL 文の中の適切な箇所にある場合にのみ有効です。 例えば INSERT 文の VALUES() リスト（その行のカラムの値を指定する）あるいは WHERE 句でカラムの値と 比較する条件を指定する場合などが有効です。
> しかし、識別子（テーブル名やカラム名）、SELECT 文が返すカラム名の一覧、 あるいは（例えば = のような）二項演算子の両側などにマーカを指定することはできません。最後の制限が必要なのは、 この場合にパラメータの型が決定できなくなるからです。 一般的に、パラメータはデータ操作言語 (DML) 文で使用し、 データ定義言語 (DDL) 文では使用しません。*

パラメータマーカは、文を実行したり行を取得したりする前に、必ず maxdb_stmt_bind_param() や maxdb_stmt_bind_result() でアプリケーションの変数にバインドしなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$city = "Portland";

/* プリペアドステートメントを作成します */
$stmt =  $maxdb->stmt_init();
if ($stmt->prepare("SELECT state FROM hotel.city WHERE name=?")) {

   /* マーカにパラメータをバインドします */
   $stmt->bind_param("s", $city);

   /* クエリを実行します */
   $stmt->execute();

   /* 結果変数にバインドします */
   $stmt->bind_result($district);

   /* 値を取得します */
   $stmt->fetch();

   printf("%s is in district %s¥n", $city, $district);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$city = "Portland";

/* プリペアドステートメントを作成します */
$stmt = maxdb_stmt_init($link);
if (maxdb_stmt_prepare($stmt, "SELECT state FROM hotel.city WHERE name=?")) {

   /* マーカにパラメータをバインドします */
   maxdb_stmt_bind_param($stmt, "s", $city);

   /* クエリを実行します */
   maxdb_stmt_execute($stmt);

   /* 結果変数にバインドします */
   maxdb_stmt_bind_result($stmt, $district);

   /* 値を取得します */
   maxdb_stmt_fetch($stmt);

   printf("%s is in district %s¥n", $city, $district);

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Portland is in district OR
```

参考

- maxdb_stmt_init()
- maxdb_stmt_execute()
- maxdb_stmt_fetch()

- maxdb_stmt_bind_param()
- maxdb_stmt_bind_result()
- maxdb_stmt_close()

---

maxdb_stmt_reset

stmt->reset

(No version information available, might be only in CVS)

stmt->reset — プリペアドステートメントをリセットする

説明

手続き型

bool **maxdb_stmt_reset** (resource $stmt)

オブジェクト指向型（メソッド）

stmt
bool **reset** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

maxdb_stmt_result_metadata

(PECL maxdb:1.0-7.6.00.38)

maxdb_stmt_result_metadata — プリペアドステートメントから、結果セットのメタデータを返す

説明

手続き型

resource **maxdb_stmt_result_metadata** (resource $stmt)

オブジェクト指向型（メソッド）

stmt
resource **result_metadata** (void)

maxdb_prepare() に渡されたステートメントが 結果セットを作成するものであった場合に、 **maxdb_stmt_result_metadata()** は結果リソースを返します。 これを使用することで、フィールドの総数や各フィールドの情報といったメタ情報を 処理することができます。

> **注意**: この結果セットポインタは、結果セットメタデータを処理するフィールドベースの関数、 例えば以下のような関数への引数として渡すことができます。
>
> - maxdb_num_fields()
> - maxdb_fetch_field()
> - maxdb_fetch_field_direct()
> - maxdb_fetch_fields()
> - maxdb_field_count()
> - maxdb_field_seek()
> - maxdb_field_tell()
> - maxdb_free_result()

結果セットの構造体は、使用が終わったあとに開放しなければなりません。 そのためには、それをmaxdb_free_result() に渡します。

> **注意**: **maxdb_stmt_result_metadata()** が返す結果セットには、 メタデータのみが含まれます。結果の行は含まれません。行を取得するには、 ステートメントハンドルを maxdb_fetch() に渡します。

返り値

maxdb_stmt_result_metadata() は結果リソースを返します。 エラーが発生した場合には **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

$maxdb->query("CREATE TABLE temp.friends (id int, name varchar(20))");

$maxdb->query("INSERT INTO temp.friends VALUES (1,'Hartmut')");
```

```
$maxdb->query("INSERT INTO temp.friends VALUES (2, 'Ulf')");

$stmt = $maxdb->prepare("SELECT id, name FROM temp.friends");
$stmt->execute();

/* メタデータの結果セットを取得します */
$result = $stmt->result_metadata();

/* メタデータの結果セットからフィールド情報を取得します */
$field = $result->fetch_field();

printf("フィールド名: %s\n", $field->name);

/* 結果セットを閉じます */
$result->close();

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

maxdb_query($link, "CREATE TABLE temp.friends (id int, name varchar(20))");

maxdb_query($link, "INSERT INTO temp.friends VALUES (1,'Hartmut')");
maxdb_query($link, "INSERT INTO temp.friends VALUES (2, 'Ulf')");

$stmt = maxdb_prepare($link, "SELECT id, name FROM temp.friends");
maxdb_stmt_execute($stmt);

/* メタデータの結果セットを取得します */
$result = maxdb_stmt_result_metadata($stmt);

/* メタデータの結果セットからフィールド情報を取得します */
$field = maxdb_fetch_field($result);

printf("フィールド名: %s\n", $field->name);

/* 結果セットを閉じます */
maxdb_free_result($result);

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
フィールド名: ID
```

参考

- maxdb_prepare()
- maxdb_free_result()

---

maxdb_stmt_send_long_data

stmt->send_long_data

(No version information available, might be only in CVS)

stmt->send_long_data — データを複数ブロックで送信する

説明

手続き型

bool **maxdb_stmt_send_long_data** (resource $stmt , int $param_nr , string $data)

オブジェクト指向型（メソッド）

stmt
bool **stmt_send_long_data** (int $param_nr , string $data)

パラメータのデータを、いくつかに分割してサーバに送信できるようにします。 この関数は、文字データやバイナリデータをカラムに送信するために複数回コールすることができます。このカラムは TEXT 型あるいは BLOB 型でなければなりません。

param_nr は、データに関連付けるパラメータを指定します。 パラメータは 0 から数え始めます。 data は、送信するデータを含む文字列です。

> **注意**: 効率を考えると、この関数は maxdb_execute() をコールした後にコールすべきです。この場合、 データはクライアント側には保存されません。このシーケンスを終える際には、 最後に maxdb_stmt_close_long_data() をコールしなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- maxdb_prepare()
- maxdb_stmt_bind_param()

---

maxdb_stmt_sqlstate

(PECL maxdb:1.0-7.6.00.38)

maxdb_stmt_sqlstate — 事前のステートメントの捜査からの SQLSTATE エラーを返す

説明

string **maxdb_stmt_sqlstate** (resource $stmt)

直近に起動したプリペアドステートメントについての SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 つの文字で構成されています。'00000' は、エラーが発生しなかったことを意味します。 コードの内容は ANSI SQL および ODBC で指定されています。

> **注意**: 今のところ、すべての MaxDB エラーが SQLSTATE に関連付けられているわけではないことに注意しましょう。 関連付けられていないエラーについては、 HY000 (一般的なエラー) が使用されます。

返り値

直近のエラーについての SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 つの文字で構成されています。'00000' は、エラーが発生しなかったことを意味します。

例

Example#1 オブジェクト指向型

```php
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city");
$maxdb->query("INSERT INTO temp.mycity SELECT * FROM hotel.city");

$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = $maxdb->prepare($query)) {

   /* テーブルを削除します */
   $maxdb->query("DROP TABLE temp.mycity");

   /* クエリを実行します */
   $stmt->execute();

   printf("エラー: %s.\n", $stmt->sqlstate);

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```php
<?php
/* 接続をオープンします */
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");
maxdb_query($link, "INSERT INTO temp.mycity SELECT * FROM hotel.city");

$query = "SELECT name, zip FROM temp.mycity ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {

   /* テーブルを削除します */
   maxdb_query($link, "DROP TABLE temp.mycity");

   /* クエリを実行します */
   maxdb_stmt_execute($stmt);

   printf("エラー: %s.\n", maxdb_stmt_sqlstate($stmt));

   /* ステートメントを閉じます */
   maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
```

```
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_stmt_execute(): -4004 POS(23) Unknown table name:MYCITY [42000] <...>
エラー: 42000.
```

参考

- maxdb_stmt_errno()
- maxdb_stmt_error()

---

maxdb_stmt_store_result

maxdb->store_result

(PECL maxdb:1.0-7.6.00.38)

maxdb->store_result — プリペアドステートメントから結果を転送する

説明

手続き型

bool **maxdb_stmt_store_result** (resource $stmt)

オブジェクト指向型（メソッド）

maxdb
object **store_result** (void)

maxdb_stmt_store_result() は、何の動作もしません。 MaxDB サーバからデータを取得するために使用すべきではありません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, zip FROM hotel.city ORDER BY name";
if ($stmt = $maxdb->prepare($query)) {

   /* クエリを実行します */
   $stmt->execute();

   /* 結果を保存します */
   $stmt->store_result();

   printf("行数: %d.\n", $stmt->num_rows);

   /* free result */
   $stmt->free_result();

   /* ステートメントを閉じます */
   $stmt->close();
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query = "SELECT name, zip FROM hotel.city ORDER BY name";
if ($stmt = maxdb_prepare($link, $query)) {
```

```
    /* クエリを実行します */
    maxdb_stmt_execute($stmt);

    /* 結果を保存します */
    maxdb_stmt_store_result($stmt);

    printf("行数: %d.\n", maxdb_stmt_num_rows($stmt));

    /* free result */
    maxdb_stmt_free_result($stmt);

    /* ステートメントを閉じます */
    maxdb_stmt_close($stmt);
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
行数: 25.
```

参考

- maxdb_prepare()
- maxdb_stmt_result_metadata()
- maxdb_fetch()

---

maxdb_store_result

maxdb->store_result

(PECL maxdb:1.0-7.6.00.38)

maxdb->store_result — 直近のクエリから結果セットを転送する

説明

手続き型

resource **maxdb_store_result** (resource $link)

オブジェクト指向型（メソッド）

maxdb
object **store_result** (void)

この関数は、何の機能も持ちません。

返り値

結果リソース、あるいはエラーが発生した場合に **FALSE** を返します。

例

maxdb_multi_query() を参照ください。

参考

- maxdb_real_query()
- maxdb_use_result()

---

maxdb_thread_id

maxdb->thread_id

(PECL maxdb:1.0-7.6.00.38)

maxdb->thread_id — 現在の接続のスレッド ID を返す

説明

手続き型

int **maxdb_thread_id** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
int$thread_id;

maxdb_thread_id() 関数は、現在の接続のスレッド ID を返します。maxdb_kill() 関数を使用することで、 この接続を殺すことが可能です。接続を見失ったあとで maxdb_ping() を使用して再接続した場合には、 スレッド ID は別の値になります。そのため、スレッド ID は、必要になった時点でのみ取得すべきです。

> **注意**: スレッド ID は、接続ごとに割り当てられます。したがって、 接続が壊れてしまい改めて確立しなおした場合には、スレッド ID は別の値が割り当てられます。

返り値

maxdb_thread_id() は、 現在の接続のスレッド ID を返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* スレッド ID を調べます */
$thread_id = $maxdb->thread_id;

/* 接続を殺します */
$maxdb->kill($thread_id);

/* これは、エラーとなります */
if (!$maxdb->query("CREATE TABLE mycity LIKE hotel.city")) {
   printf("エラー: %s\n", $maxdb->error);
    exit;
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

/* スレッド ID を調べます */
$thread_id = maxdb_thread_id($link);

/* 接続を殺します */
maxdb_kill($link, $thread_id);

/* これは、エラーとなります */
if (!maxdb_query($link, "CREATE TABLE mycity LIKE hotel.city")) {
   printf("エラー: %s\n", maxdb_error($link));
    exit;
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_query(): -10821 Session not connected <...>
エラー: Session not connected
```

参考

- maxdb_kill()

---

maxdb_thread_safe

(No version information available, might be only in CVS)

maxdb_thread_safe — スレッドセーフであるかどうかを返す

説明

手続き型

bool **maxdb_thread_safe** (void)

maxdb_thread_safe() は、 クライアントライブラリがスレッドセーフにコンパイルされているかどうかを示します。

返り値

クライアントライブラリがスレッドセーフである場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

---

maxdb_use_result

maxdb->use_result

(PECL maxdb:1.0-7.6.00.38)

maxdb->use_result — 結果セットの取得を開始する

説明

手続き型

resource **maxdb_use_result** (resource $link)

オブジェクト指向型（メソッド）

maxdb
resource **use_result** (void)

maxdb_use_result() は何の動作もしません。

返り値

結果、あるいはエラー時に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```php
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query  = "SELECT * FROM DUAL";

/* 複数クエリを実行します */
if ($maxdb->multi_query($query)) {
    do {
        /* 最初の結果セットを保存します */
        if ($result = $maxdb->use_result()) {
            while ($row = $result->fetch_row()) {
                printf("%s¥n", $row[0]);
            }
            $result->close();
        }
        /* 区切りを表示します */
        if ($maxdb->more_results()) {
            printf("-----------------¥n");
        }
    } while ($maxdb->next_result());
}

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```php
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$query  = "SELECT * FROM DUAL";

/* 複数クエリを実行します */
if (maxdb_multi_query($link, $query)) {
    do {
        /* 最初の結果セットを保存します */
        if ($result = maxdb_use_result($link)) {
            while ($row = maxdb_fetch_row($result)) {
                printf("%s¥n", $row[0]);
            }
            maxdb_free_result($result);
        }
        /* 区切りを表示します */
        if (maxdb_more_results($link)) {
```

```
            printf("-----------------¥n");
        }
    } while (maxdb_next_result($link));
}

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
a
```

参考

- maxdb_real_query()
- maxdb_store_result()

---

maxdb_warning_count

maxdb->warning_count

(PECL maxdb:1.0-7.6.00.38)

maxdb->warning_count — 指定したリンクの直近のクエリで発生した警告の数を返す

説明

手続き型

int **maxdb_warning_count** (resource $link)

オブジェクト指向型（プロパティ）

maxdb
int$warning_count;

maxdb_warning_count() は、 link で表される接続の、 直近のクエリから返される警告の数を返します。

返り値

警告の数、あるいは警告が発生しなかった場合にはゼロを返します。

例

Example#1 オブジェクト指向型

```
<?php
$maxdb = new maxdb("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

$maxdb->query("CREATE TABLE temp.mycity LIKE hotel.city");

/* ウェールズにある、珍しい地名です */
$query = "INSERT INTO temp.mycity (zip, name) VALUES('11111',
          'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

$maxdb->query($query);

printf ("警告の数: %d\n", $maxdb->warning_count);

/* 接続を閉じます */
$maxdb->close();
?>
```

Example#2 手続き型

```
<?php
$link = maxdb_connect("localhost", "MONA", "RED", "DEMODB");

/* 接続を調べます */
if (maxdb_connect_errno()) {
   printf("接続に失敗しました: %s\n", maxdb_connect_error());
    exit();
}

maxdb_query($link, "CREATE TABLE temp.mycity LIKE hotel.city");

/* ウェールズにある、とても長い名前の市です */
$query = "INSERT INTO temp.mycity (zip, name) VALUES('11111',
          'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

maxdb_query($link, $query);
```

```
printf ("警告の数:  %d\n", maxdb_warning_count($link));

/* 接続を閉じます */
maxdb_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Warning: maxdb_query(): -8004 POS(62) Constant must be compatible with column type and length <...>
警告の数: 0
```

参考

- maxdb_errno()
- maxdb_error()
- maxdb_sqlstate()

目次

- maxdb_init — MaxDB を初期化し、maxdb_real_connect で使用するリソースを返す
- maxdb_insert_id — 直近のクエリで使用した、自動生成 ID を返す
- maxdb_kill — MaxDB サーバから切断する
- maxdb_master_query — マスタ/スレーブ構成において、クエリをマスタ側で実行することを強制する
- maxdb_more_results — 複数クエリの結果の中に結果セットがまだあるかどうかを調べる
- maxdb_multi_query — データベース上でクエリを実行する
- maxdb_next_result — multi_query の、次の結果を準備する
- maxdb_num_fields — 結果のフィールド数を取得する
- maxdb_num_rows — 結果の行数を取得する
- maxdb_options — オプションを設定する
- maxdb_param_count — maxdb_stmt_param_count のエイリアス
- maxdb_ping — サーバとの接続を確認し、接続が確立されていない場合は再接続を試みる
- maxdb_prepare — 後で実行するための SQL 文を準備する
- maxdb_query — データベース上でクエリを実行する
- maxdb_real_connect — MaxDB サーバへの接続をオープンする
- maxdb_real_escape_string — 現在の接続の文字セットを考慮したうえで、 SQL 文で使用される文字列中の特殊文字をエスケープする
- maxdb_real_query — SQL クエリを実行する
- maxdb_report — 内部のレポート関数を有効あるいは無効にする
- maxdb_rollback — 現在のトランザクションをロールバックする
- maxdb_rpl_parse_enabled — RPL のパースが有効かどうかを調べる
- maxdb_rpl_probe — RPL を調べる
- maxdb_rpl_query_type — RPL クエリ型を返す
- maxdb_select_db — データベースクエリ用のデフォルトデータベースを選択する
- maxdb_send_long_data — maxdb_stmt_send_long_data のエイリアス
- maxdb_send_query — クエリを送信する
- maxdb_server_end — 埋め込みサーバをシャットダウンする
- maxdb_server_init — 埋め込みサーバを初期化する
- maxdb_set_opt — maxdb_options のエイリアス
- maxdb_sqlstate — 直近の MaxDB 操作の SQLSTATE エラーを返します
- maxdb_ssl_set — SSL を使用したセキュアな接続を確立するために使用する
- maxdb_stat — 現在のシステム状態を取得する
- maxdb_stmt_affected_rows — 直近のステートメントによって変更、削除あるいは挿入された行の数を返す
- maxdb_stmt_bind_param — プリペアドステートメントに、変数をパラメータとしてバインドする
- maxdb_stmt_bind_result — 結果を保存するために、変数をプリペアドステートメントにバインドする
- maxdb_stmt_close_long_data — maxdb_stmt_send_long_data のシーケンスを終了する
- maxdb_stmt_close — プリペアドステートメントを閉じる
- maxdb_stmt_data_seek — ステートメントの結果セットの、任意の行に移動する
- maxdb_stmt_errno — 直近のステートメントコール時のエラーコードを返す
- maxdb_stmt_error — 直近のステートメントコール時のエラー文字列を返す
- maxdb_stmt_execute — プリペアドクエリを実行する
- maxdb_stmt_fetch — プリペアドステートメントの結果を取得し、バインド変数に格納する
- maxdb_stmt_free_result — 指定したステートメントハンドルの結果を保存しているメモリを開放する
- maxdb_stmt_init — ステートメントを初期化し、maxdb_stmt_prepare で使用するリソースを返す
- maxdb_stmt_num_rows — ステートメントの結果セットの行数を返す
- maxdb_stmt_param_count — 指定したステートメントのパラメータ数を返す
- maxdb_stmt_prepare — 後で実行するための SQL 文を準備する
- maxdb_stmt_reset — プリペアドステートメントをリセットする
- maxdb_stmt_result_metadata — プリペアドステートメントから、結果セットのメタデータを返す
- maxdb_stmt_send_long_data — データを複数ブロックで送信する
- maxdb_stmt_sqlstate — 事前のステートメントの捜査からの SQLSTATE エラーを返す
- maxdb_stmt_store_result — プリペアドステートメントから結果を転送する
- maxdb_store_result — 直近のクエリから結果セットを転送する
- maxdb_thread_id — 現在の接続のスレッド ID を返す
- maxdb_thread_safe — スレッドセーフであるかどうかを返す
- maxdb_use_result — 結果セットの取得を開始する
- maxdb_warning_count — 指定したリンクの直近のクエリで発生した警告の数を返す

MCAL 関数

導入

MCAL は、Modular Calendar Access Library を意味します。

libmcal は、カレンダーアクセス用の C ライブラリです。このライブラリは、 モジュール化されており、後から追加しやすいドライバとして書かれ

ています。 MCAL は、メールボックス用の IMAP モジュールのカレンダー版です。

mcal サポート機能により、カレンダーストリームを IMAP サポート機能による メールボックスストリームと全く同様にオープンすることができます。 カレンダーは、ローカルファイルに保存したり、リモートの ICAP サーバ としたり、mcal ライブラリによりサポートされる他のフォーマットとする ことができます。

カレンダーイベントは、発生、検索、保存することができます。カレンダーの トリガー(アラーム)および定期的なイベントもサポートされます。

libmcal により、メインのカレンダーサーバーをアクセス、使用することが できます。この場合、特定のデータベースおよびローカルファイルに依存した プログラミングは必要ありません。

ほとんどの関数は、ストリームごとに固有の内部イベント構造体を使用します。 これにより、大きなオブジェクトを関数の間で渡すことが容易になります。 イベント構造体の値を設定、初期化、取得する便利な関数があります。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.0.0.
>
> **注意**: PHP には ICAP 拡張モジュールが以前ありましたが、元のライブラリと この PHP 拡張モジュールはもうサポートされていません。推奨される 代替品は MCAL です。
>
> **注意**: この拡張モジュールは Windows 環境では利用できません。

要件

mcal ライブラリがインストールされている必要があります。最新版を » http://mcal.chek.com/ から取得し、コンパイル、 インストールしてください。

インストール手順

mcal ライブラリをインストールした後、これらの関数を動作させるには、 --with-mcal[=DIR] を付けて PHP を コンパイルする必要があります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

MCAL_SUNDAY (integer)
MCAL_MONDAY (integer)
MCAL_TUESDAY (integer)
MCAL_WEDNESDAY (integer)
MCAL_THURSDAY (integer)
MCAL_FRIDAY (integer)
MCAL_SATURDAY (integer)
MCAL_JANUARY (integer)
MCAL_FEBRUARY (integer)
MCAL_MARCH (integer)
MCAL_APRIL (integer)
MCAL_MAY (integer)
MCAL_JUNE (integer)
MCAL_JULY (integer)
MCAL_AUGUST (integer)
MCAL_SEPTEMBER (integer)
MCAL_OCTOBER (integer)
MCAL_NOVEMBER (integer)
MCAL_DECEMBER (integer)
MCAL_RECUR_NONE (integer)
MCAL_RECUR_DAILY (integer)
MCAL_RECUR_WEEKLY (integer)
MCAL_RECUR_MONTHLY_MDAY (integer)
MCAL_RECUR_MONTHLY_WDAY (integer)
MCAL_RECUR_YEARLY (integer)
MCAL_M_SUNDAY (integer)
MCAL_M_MONDAY (integer)
MCAL_M_TUESDAY (integer)
MCAL_M_WEDNESDAY (integer)
MCAL_M_THURSDAY (integer)
MCAL_M_FRIDAY (integer)
MCAL_M_SATURDAY (integer)
MCAL_M_WEEKDAYS (integer)
MCAL_M_WEEKEND (integer)
MCAL_M_ALLDAYS (integer)

---

mcal_append_event

(PHP 4)

mcal_append_event — MCALカレンダーに新規イベントを保存する

説明

int **mcal_append_event** (int $mcal_stream)

mcal_append_event()はグローバルイベントを 指定したストリームのMCALカレンダーに保存します。
新規に挿入されたイベントのuidを返します。

---

mcal_close

(PHP 4)

mcal_close — MCAL ストリームを閉じる

説明

bool **mcal_close** (int $mcal_stream [, int $flags])
指定した mcal ストリームを閉じます。

---

mcal_create_calendar

(PHP 4)

mcal_create_calendar — 新規に MCAL カレンダを作成する

説明

bool **mcal_create_calendar** (int $stream , string $calendar)
calendar という名前で新規のカレンダを 作成します。

---

mcal_date_compare

(PHP 4)

mcal_date_compare — 二つの日付を比較する

説明

int **mcal_date_compare** (int $a_year , int $a_month , int $a_day , int $b_year , int $b_month , int $b_day)
mcal_date_compare() は、二つの日付を比較し、 a<b , a==b, a>b の場合に、それぞれ、<0, 0, >0 を返します。

---

mcal_date_valid

(PHP 4)

mcal_date_valid — 指定した年月日が有効な日付である場合に **TRUE** を返す

説明

bool **mcal_date_valid** (int $year , int $month , int $day)
mcal_date_valid()は、指定した年月日が有効な日 付である場合に **TRUE** を、そうでない場合に **FALSE** を返します。

---

mcal_day_of_week

(PHP 4)

mcal_day_of_week — 指定した日の曜日を返す

説明

int **mcal_day_of_week** (int $year , int $month , int $day)
mcal_day_of_week() は、指定した日の曜日を返します。 返される値の範囲は、日曜日を意味する 0 から土曜日を意味する 6 までです。

---

mcal_day_of_year

(PHP 4)

mcal_day_of_year — 指定した日の年間積算日を返す

説明

int **mcal_day_of_year** (int $year , int $month , int $day)

mcal_day_of_year() は、 指定した日の年間積算日を返します。

---

mcal_days_in_month

(PHP 4)

mcal_days_in_month — 月の日数を返す

説明

int **mcal_days_in_month** (int $month , int $leap_year)

mcal_days_in_month() は、 指定した年が閏年であるかどうかを考慮しつつ month 月の日数を返します。

---

mcal_delete_calendar

(PHP 4)

mcal_delete_calendar — MCAL カレンダを削除する

説明

bool **mcal_delete_calendar** (int $stream , string $calendar)

calendar という名前のカレンダを削除します。

---

mcal_delete_event

(PHP 4)

mcal_delete_event — MCAL カレンダーからイベントを削除する

説明

bool **mcal_delete_event** (int $mcal_stream , int $event_id)

mcal_delete_event() は event_id で指定した カレンダーイベントを削除します。

TRUE を返します。

---

mcal_event_add_attribute

(PHP 4)

mcal_event_add_attribute — グローバルイベント構造体ストリームに属性およびその値を追加する

説明

bool **mcal_event_add_attribute** (int $stream , string $attribute , string $value)

mcal_event_add_attribute() は、ストリームの グローバルイベント構造体に、"value" という値の属性を追加します。

---

mcal_event_init

(PHP 4)

mcal_event_init — グローバルイベント構造体のストリームを初期化する

説明

void **mcal_event_init** (int $stream)

mcal_event_init() は、グローバルイベント構造体 ストリームを初期化します。これにより、構造体の全ての要素に効率的に 0 またはデフォルト値が設定されます。

---

mcal_event_set_alarm

(PHP 4)

mcal_event_set_alarm — グローバルイベント構造体ストリームのアラームを設定する

説明

void ***mcal_event_set_alarm*** *(int $stream , int $alarm)*

mcal_event_set_alarm() *は、グローバルイベント構造体 ストリームのアラームをイベントの分単位の指定時間前に設定します。*

---

mcal_event_set_category

(PHP 4)

mcal_event_set_category — グローバルイベント構造体ストリームのカテゴリを設定する

説明

void ***mcal_event_set_category*** *(int $stream , string $category)*

mcal_event_set_category() *は、グローバルイベント 構造体ストリームのカテゴリを指定した文字列に設定します。*

---

mcal_event_set_class

(PHP 4)

mcal_event_set_class — グローバルイベント構造体ストリームのクラスを設定する

説明

void ***mcal_event_set_class*** *(int $stream , int $class)*

mcal_event_set_class() *は、グローバルイベント 構造体ストリームのクラスを指定した値に設定します。class は、 public の場合 1、private の場合 0 とします。*

---

mcal_event_set_description

(PHP 4)

mcal_event_set_description — グローバルイベント構造体ストリームの説明を設定する

説明

void ***mcal_event_set_description*** *(int $stream , string $description)*

mcal_event_set_description() *は、グローバル イベント構造体ストリームの説明を、指定した文字列に設定します。*

---

mcal_event_set_end

(PHP 4)

mcal_event_set_end — グローバルイベント構造体ストリームの終了日を設定する

説明

void ***mcal_event_set_end*** *(int $stream , int $year , int $month , int $day [, int $hour [, int $min [, int $sec]]])*

mcal_event_set_end() *は、グローバルイベント 構造体ストリームの終了日を指定した値に設定します。*

---

mcal_event_set_recur_daily

(PHP 4)

mcal_event_set_recur_daily — グローバルイベント構造体ストリームの反復を設定する

説明

void ***mcal_event_set_recur_daily*** *(int $stream , int $year , int $month , int $day , int $interval)*

mcal_event_set_recur_daily() *は、 グローバルイベント構造体ストリームが次に発生する日を 指定した値に設定します。イベントが、 指定した日まで毎日発生するものとして設定します。*

---

mcal_event_set_recur_monthly_mday

(PHP 4)

mcal_event_set_recur_monthly_mday — グローバルイベント構造体ストリームの繰り返しを設定する

説明

void ***mcal_event_set_recur_monthly_mday*** *(int $stream , int $year , int $month , int $day , int $interval)*

mcal_event_set_recur_monthly_mday() *は、 グローバルイベント構造体ストリームが次に発生する日を 指定した値に設定します。イベントが、指定した日まで毎月指定日に発生するものとして設定します。*

---

mcal_event_set_recur_monthly_wday

(PHP 4)

mcal_event_set_recur_monthly_wday — グローバルイベント構造体ストリームの繰り返しを設定する

説明

void ***mcal_event_set_recur_monthly_wday*** *(int $stream , int $year , int $month , int $day , int $interval)*

mcal_event_set_recur_monthly_wday() *は、 グローバルイベント構造体ストリームが次に発生する日を 指定した値に設定します。イベントが、指定した日まで毎月指定曜日に発生するものとして設定します。*

---

mcal_event_set_recur_none

(PHP 4)

mcal_event_set_recur_none — グローバルイベント構造体ストリームの反復を設定する

説明

void ***mcal_event_set_recur_none*** *(int $stream)*

mcal_event_set_recur_none() *は、 グローバルイベント構造体ストリームの反復が行われないことを 指定します(event->recur_type は MCAL_RECUR_NONE に設定されます)。*

---

mcal_event_set_recur_weekly

(PHP 4)

mcal_event_set_recur_weekly — グローバルイベント構造体ストリームの繰り返しを設定する

説明

void ***mcal_event_set_recur_weekly*** *(int $stream , int $year , int $month , int $day , int $interval , int $weekdays)*

mcal_event_set_recur_weekly() *は、 グローバルイベント構造体ストリームが次に発生する日を 指定した値に設定します。イベントが、 指定した日まで毎週指定曜日に発生するものとして設定します。*

---

mcal_event_set_recur_yearly

(PHP 4)

mcal_event_set_recur_yearly — グローバルイベント構造体ストリームの繰り返しを設定する

説明

void ***mcal_event_set_recur_yearly*** *(int $stream , int $year , int $month , int $day , int $interval)*

mcal_event_set_recur_yearly() *は、 グローバルイベント構造体ストリームが次に発生する日を 指定した値に設定します。イベントが、 指定した日まで毎年指定日に発生するものとして設定します。*

---

mcal_event_set_start

(PHP 4)

mcal_event_set_start — グローバルイベント構造体ストリームの開始日を設定する

説明

void ***mcal_event_set_start*** *(int $stream , int $year , int $month , int $day [, int $hour [, int $min [, int $sec]]])*

mcal_event_set_start() *は、グローバルイベント 構造体ストリームの開始日を指定した値に設定します。*

---

mcal_event_set_title

(PHP 4)

mcal_event_set_title — グローバルイベント構造体ストリームの表題を設定する

説明

void **mcal_event_set_title** (int $stream , string $title)

mcal_event_set_title() は、グローバルイベント 構造体ストリームの表題を指定した文字列に設定します。

---

mcal_expunge

(PHP 4)

mcal_expunge — 削除マークを付けられた全てのイベントを削除する

説明

bool **mcal_expunge** (int $stream)

mcal_expunge() は、過去に削除マークを付けられた 全てのイベントを削除します。

---

mcal_fetch_current_stream_event

(PHP 4)

mcal_fetch_current_stream_event — 現在のイベント構造体ストリームを有するオブジェクトを返す

説明

object **mcal_fetch_current_stream_event** (int $stream)

mcal_fetch_current_stream_event() は、次の要素を有する現在のイベント構造ストリームを有する オブジェクトを返します。

- int id - イベントID
- int public - イベントが public の場合に **TRUE**、private の場合に **FALSE**。
- string category - イベントの Category 文字列。
- string title - イベントの Title 文字列。
- string description - イベントの Description 文字列。
- int alarm - アラーム・リマインダを送るイベントまでの分単位の時間。
- object start - 開始 datetime エントリを有するオブジェクト。
- object end - 終了 datetime エントリを有するオブジェクト。
- int recur_type - 繰り返し型
- int recur_interval - 繰り返し間隔
- datetime recur_enddate - 繰り返し終了日
- int recur_data - 繰り返しデータ

全ての datetime エントリは、次の要素を有するオブジェクトから 構成されます。

- int year - 年
- int month - 月
- int mday - 日
- int hour - 時
- int min - 分
- int sec - 秒
- int alarm - イベントを送信するまでの分単位の時間

---

mcal_fetch_event

(PHP 4)

mcal_fetch_event — カレンダーストリームからイベントを取得する

説明

object **mcal_fetch_event** (int $mcal_stream , int $event_id [, int $options])

mcal_fetch_event() は、 id で指定されたカレンダーストリームから イベントを取得します。

以下の要素からなるイベントオブジェクトを返します。

- int id - イベント ID
- int public - イベントが public な場合は **TRUE**、private な場合は **FALSE**
- string category - イベントのカテゴリを表す文字列
- string title - イベントのタイトルを表す文字列
- string description - イベントの内容を表す文字列

- int alarm - アラーム・リマインダイベントが送られるまでの分単位 の時間
- object start - datetime エントリを有するオブジェクト
- object end - datetime エントリを有するオブジェクト
- int id - イベント ID
- int public - イベントが public な場合は TRUE、private な場合は FALSE
- string category - イベントのカテゴリを表す文字列
- string title - イベントのタイトルを表す文字列
- string description - イベントの内容を表す文字列
- int alarm - アラーム・リマインダイベントが送られるまでの分単位 の時間
- object start - datetime エントリを有するオブジェクト
- object end - datetime エントリを有するオブジェクト
- int recur_type - 繰り返しの型
- int recur_interval - 繰り返し周期
- datetime recur_enddate - 繰り返しが終わる日
- int recur_data - 繰り返しデータ

全ての datetime エントリは、次の要素を有するオブジェクトから 構成されます。

- int year - 年
- int month - 月
- int mday - 日
- int hour - 時間
- int min - 分
- int sec - 秒
- int alarm - アラームを送るイベントまでの分単位の時間

recur_type で使用可能な値を以下に示します。

- 0 - このイベントが繰り返さないことを示す
- 1 - このイベントが一日毎に繰り返す
- 2 - このイベントが1週間毎に繰り返す
- 3 - このイベントは指定した日に毎月発生する（例 毎月十日）
- 4 - このイベントは月単位の週の曜日に発生する（例 第３土曜日）
- 5 - このイベントは年単位で発生する

---

mcal_is_leap_year

(PHP 4)

mcal_is_leap_year — 指定した年が閏年であるかどうかを返す

説明

bool **mcal_is_leap_year** (int $year)

mcal_is_leap_year() は指定した年が 閏年である場合に 1、そうでない場合に 0 を返します。

---

mcal_list_alarms

(PHP 4)

mcal_list_alarms — 指定した datetime までにアラームを発生するイベントのリストを返す

説明

array **mcal_list_alarms** (int $mcal_stream [, int $begin_year], int $begin_month , int $begin_day , int $end_year , int $end_month , int $end_day)

開始日から終了日までの間にアラームがオフとなるイベントのIDの配列が 返されます。ストリームだけが指定された場合、グローバルイベント構造体の 開始日および終了日を使用します。

mcal_list_events() オプションでカレンダーストリームの 開始日と終了日をとります。指定した日または内部イベントの 間にあるイベントの ID の配列が返されます。

---

mcal_list_events

(PHP 4)

mcal_list_events — 日付または日付の範囲に関して ID のリストを返す

説明

array **mcal_list_events** (int $mcal_stream [, int $begin_year], int $begin_month , int $begin_day , int $end_year , int $end_month , int $end_day)

開始日および終了日の間のイベントのIDの配列を返します。ストリームだけを 指定した場合、グローバルイベント構造体の開始日および終了日を使用します。

mcal_list_events() は、カレンダーストリームの 開始日およびオプションで終了日をとります。指定した日または内部イベントの 間にあるイベントの ID の配列が返されます。

---

mcal_next_recurrence

(PHP 4)

mcal_next_recurrence — イベントが次に発生する日を返す

説明

int **mcal_next_recurrence** (int $stream , int $weekstart , array $next)

mcal_next_recurrence() は、指定した日以降に 次のイベントが発生する日を有するオブジェクトを返します。 イベントが発生しないか、何か問題がある場合は空の date フィールドを 返します。weekstart により週の初めとなる曜日を指定することができます。

---

mcal_open

(PHP 4)

mcal_open — MCAL 接続をオープンする

説明

int **mcal_open** (string $calendar , string $username , string $password [, int $options])

成功時に MCAL ストリーム、エラー時に **FALSE** を返します。

mcal_open() は、指定した calendar 保存ファイルへの MCAL 接続を オープンします。 オプション options が指定された場合、 options もメールボックスに渡されます。 ストリームの内部イベント構造体も接続時に初期化されます。

---

mcal_popen

(PHP 4)

mcal_popen — 永続的な MCAL 接続をオープンする

接続

int **mcal_popen** (string $calendar , string $username , string $password [, int $options])

成功時に MCAL ストリームを、エラー時に **FALSE** を返します。

mcal_popen() は、指定した calendar が保存しているMCAL接続をオープン します。オプションの options が指定された場合、 options をメールボックスにも渡します。 内部イベント構造体のストリームも接続時に初期化されます。

---

mcal_rename_calendar

(PHP 4)

mcal_rename_calendar — MCAL カレンダの名前を変更する

説明

bool **mcal_rename_calendar** (int $stream , string $old_name , string $new_name)

カレンダ名 old_name を new_name に変更します。

---

mcal_reopen

(PHP 4)

mcal_reopen — MCAL 接続を再オープンする

説明

bool **mcal_reopen** (int $mcal_stream , string $calendar [, int $options])

MCAL ストリームを新規カレンダー用に再オープンします。

mcal_reopen()は、指定した calendar が保存したMCAL接続を再オープンします。 オプションの options が指定された場合、 options をメール

ボックスにも渡します。

---

mcal_snooze

(PHP 4)

mcal_snooze — イベントのアラームをオフにする

説明

bool **mcal_snooze** (int $stream_id , int $event_id)

mcal_snooze() は stream_id および event_id で指定したカレンダーイベントの アラームをオフにします。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

mcal_store_event

(PHP 4)

mcal_store_event — MCAL カレンダーの既存のイベントを修正する

説明

int **mcal_store_event** (int $mcal_stream)

mcal_store_event() は、修正したイベントを 指定したストリームの現在のグローバルイベントに保存します。

成功時に修正したイベントのイベント ID、エラー時に **FALSE** を返します。

---

mcal_time_valid

(PHP 4)

mcal_time_valid — 指定した時間、分、秒が有効な時間である場合に **TRUE** を返す

説明

bool **mcal_time_valid** (int $hour , int $minutes , int $seconds)

mcal_time_valid() は、指定した時間、分、秒が 有効な時間である場合に **TRUE** を、そうでない場合に **FALSE** を 返します。

---

mcal_week_of_year

(PHP 4)

mcal_week_of_year — 指定した日付の週番号を返す

説明

int **mcal_week_of_year** (int $day , int $month , int $year)

mcal_week_of_year() は、指定した日付の 週番号を返します。

目次

- mcal_event_set_description — グローバルイベント構造体ストリームの説明を設定する
- mcal_event_set_end — グローバルイベント構造体ストリームの終了日を設定する
- mcal_event_set_recur_daily — グローバルイベント構造体ストリームの反復を設定する
- mcal_event_set_recur_monthly_mday — グローバルイベント構造体ストリームの繰り返しを設定する
- mcal_event_set_recur_monthly_wday — グローバルイベント構造体ストリームの繰り返しを設定する
- mcal_event_set_recur_none — グローバルイベント構造体ストリームの反復を設定する
- mcal_event_set_recur_weekly — グローバルイベント構造体ストリームの繰り返しを設定する
- mcal_event_set_recur_yearly — グローバルイベント構造体ストリームの繰り返しを設定する
- mcal_event_set_start — グローバルイベント構造体ストリームの開始日を設定する
- mcal_event_set_title — グローバルイベント構造体ストリームの表題を設定する
- mcal_expunge — 削除マークを付けられた全てのイベントを削除する
- mcal_fetch_current_stream_event — 現在のイベント構造体ストリームを有するオブジェクトを返す
- mcal_fetch_event — カレンダーストリームからイベントを取得する
- mcal_is_leap_year — 指定した年が閏年であるかどうかを返す
- mcal_list_alarms — 指定した datetime までにアラームを発生するイベントのリストを返す
- mcal_list_events — 日付または日付の範囲に関して ID のリストを返す
- mcal_next_recurrence — イベントが次に発生する日を返す
- mcal_open — MCAL 接続をオープンする
- mcal_popen — 永続的な MCAL 接続をオープンする
- mcal_rename_calendar — MCAL カレンダの名前を変更する
- mcal_reopen — MCAL 接続を再オープンする
- mcal_snooze — イベントのアラームをオフにする
- mcal_store_event — MCAL カレンダーの既存のイベントを修正する
- mcal_time_valid — 指定した時間、分、秒が有効な時間である場合に TRUE を返す
- mcal_week_of_year — 指定した日付の週番号を返す

Mcrypt 暗号化関数

導入

この関数は、CBC, OFB, CFB, ECB 暗号モードの DES, TripleDES, Blowfish (デフォルト), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 , GOST のような広範なブロックアルゴリズムをサポートする mcrypt ライブラリへのインターフェースです。加えて、"フリーではない" とされている RC6 および IDEA もサポートします。

要件

ここで示す関数は、» mcrypt を 使用して動作します。この拡張モジュールを使用するには、» http://mcrypt.sourceforge.net/ から libmcrypt-x.x.tar.gz をダウンロードし、含まれているインストール用の指示に従ってください。 Windows ユーザは、コンパイル済みの mcrypt バイナリを » http://files.edin.dk/php/win32/mcrypt/ から入手することが可能です。

PHP 5.0.0 以降では、libmcrypt のバージョン 2.5.6 以降が必要です。

libmcrypt 2.4.x とリンクした場合、加えてブロックアルゴリズム: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT および次のストリーム暗号: ENIGMA (crypt), PANAMA, RC4, WAKE がサポートされます。 libmcrypt 2.4.x を使用した場合、暗号モード nOFB もサポートされます。

インストール手順

この拡張機能を利用可能にするためには、 PHP を --with-mcrypt パラメータを 付けてコンパイルする必要があります。DIR は mcrypt のインストール ディレクトリです。必ず、 --disable-posix-threadsを付けて libmcrypt をコンパイルするようにしてください。

実行時設定

php.ini の設定により動作が変化します。

Mcrypt 設定オプション

名前	デフォルト	変更の可否	変更履歴
mcrypt.algorithms_dir	NULL	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。
mcrypt.modes_dir	NULL	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

mcrypt_module_open() は暗号記述子を返します。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

mcryptは 4 つのブロック暗号モード(CBC, OFB, CFB, ECB)で実行可能です。 libmcrypt-2.4.x 以降にリンクした場合、ブロック暗号モード nOFG と STREAM モードでも実行可能です。MCRYPT_MODE_mode 形式を関数で使用する 際には、いくつかの制約があります。ここで、これらの各

モードの通常の 使用法の概要を示します。詳細なリファレンスおよび議論に関しては、 Applied Cryptography by Schneier (ISBN 0-471-11709-9) を参照ください。

- MCRYPT_MODE_ECB (electronic codebook) は、他のキーを暗号化すると いったランダムデータに適しています。出力データが短くランダムであると いう ECB の短所は、都合の良い逆の効果を持っています。
- MCRYPT_MODE_CBC (cipher block chaining)は、特に、ECB よりも著しく 高いセキュリティでファイルを暗号化する用途に適しています。
- MCRYPT_MODE_CFB (cipher feedback) は、1 バイト毎に暗号化する必要がある バイトストリームを暗号化する際に最も適したモードです。
- MCRYPT_MODE_OFB (output feedback、8 ビット形式) はCFBと互換性が ありますが、エラーの伝播が許容されないアプリケーションに使用する ことが可能です。このモードは(8 ビットモードで処理を行うため)安全では なく、使用は推奨されません。
- MCRYPT_MODE_NOFB (output feedback, nビット形式) は OFB と互換ですが、 アルゴリズムのブロックサイズを変更可能なため、より安全です。
- MCRYPT_MODE_STREAM は、WAKE や RC4 のようないくつかのストリーム アルゴリズムを読み込む追加のモードです。

他のモードおよびランダムデバイス定数:

MCRYPT_ENCRYPT (integer)
MCRYPT_DECRYPT (integer)
MCRYPT_DEV_RANDOM (integer)
MCRYPT_DEV_URANDOM (integer)
MCRYPT_RAND (integer)

Mcrypt 暗号

以下に、mcrypt 拡張モジュールにより現在サポートされている暗号のリストを 示します。サポートされる暗号の完全なリストについては、mcrypt.h の最後にある define を参照ください。 mcrypt-2.2.x API に関する一般的な規則は、 MCRYPT_暗号名で PHP から暗号をアクセス可能であるということです。 mcrypt-2.4.x および mcrypt-2.5.x の API についてもこれらの定数は 使用できますが、mcrypt_module_open() を コールする際に文字列で暗号名を指定することも可能です。

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt > 2.4.x のみ)
- MCRYPT_ARCFOUR (libmcrypt > 2.4.x のみ)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x only)
- MCRYPT_ENIGMA (libmcrypt > 2.4.x のみ、 MCRYPT_CRYPTへのエイリアス)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt > 2.4.x のみ)
- MCRYPT_MARS (libmcrypt > 2.4.xのみ, non-free)
- MCRYPT_PANAMA (libmcrypt > 2.4.xのみ)
- MCRYPT_RIJNDAEL_128 (libmcrypt > 2.4.xのみ)
- MCRYPT_RIJNDAEL_192 (libmcrypt > 2.4.xのみ)
- MCRYPT_RIJNDAEL_256 (libmcrypt > 2.4.xのみ)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.xのみ)
- MCRYPT_RC6 (libmcrypt > 2.4.xのみ)
- MCRYPT_RC6_128 (libmcrypt 2.2.xのみ)
- MCRYPT_RC6_192 (libmcrypt 2.2.xのみ)
- MCRYPT_RC6_256 (libmcrypt 2.2.xのみ)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt > 2.4.xのみ)
- MCRYPT_SERPENT(libmcrypt > 2.4.xのみ)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.xのみ)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.xのみ)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.xのみ)
- MCRYPT_SKIPJACK (libmcrypt > 2.4.xのみ)
- MCRYPT_TEAN (libmcrypt 2.2.xのみ)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt > 2.4.xのみ)
- MCRYPT_TWOFISH (mcrypt 2.x より古いバージョン、 またはmcrypt > 2.4.xの場合)
- MCRYPT_TWOFISH128 (TWOFISHxxx は 2.x の新しいバージョンで使用可能ですが、2.4.x では使用できません)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt > 2.4.xのみ)
- MCRYPT_XTEA (libmcrypt > 2.4.xのみ)

(CFB および OFB モードでは)それぞれの暗号関数に初期化ベクトル(IV) を指定する必要があり、(CBC モードでは)IV を指定することが可能です。

IV はユニークである必要があり、暗号化/復号化の際に同じである 必要があります。暗号化されて保存されたデータの場合、関数の出力を (ファイル名の MD5 キーのように) 保存されたデータの位置を表す インデックスとして使用することができます。もしくは、暗号化された データと共に IV を渡すことができます(このトピックに関する議論については、 Applied Cryptography by Schneier (ISBN 0-471-11709-9) の 9.3 章を参照ください)。

例

mcrypt は、上に示した暗号を用いて暗号化および復号化を 行うことが可能です。libmcrypt-2.2.xとリンクした場合、4つの重要な mcrypt コマンド (mcrypt_cfb(), mcrypt_cbc(),mcrypt_ecb(), mcrypt_ofb()) は、MCRYPT_ENCRYPT および MCRYPT_DECRYPT という 2つのモードの両方で実行可能です。

Example#1 入力値を 2.2.x において ECB モードの TripleDES で暗号化する

```
<?php
$key = "this is a secret key";
$input = "Let us meet at 9 o'clock at the secret place.";

$encrypted_data = mcrypt_ecb (MCRYPT_3DES, $key, $input, MCRYPT_ENCRYPT);
?>
```

この例では、$encrypted_data に文字列として 暗号化されたデータが返されます。

libmcrypt 2.4.x または 2.5.x とリンクした場合、上記の関数も利用可能 ですが、新しい関数を使用されることを推奨します。

Example#2 2.4.x 以降において ECB モードで TripleDES により入力を暗号化する

```
<?php
    $key = "this is a secret key";
    $input = "Let us meet at 9 o'clock at the secret place.";

    $td = mcrypt_module_open('tripledes', '', 'ecb', '');
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size($td), MCRYPT_RAND);
    mcrypt_generic_init($td, $key, $iv);
    $encrypted_data = mcrypt_generic($td, $input);
    mcrypt_generic_deinit($td);
    mcrypt_module_close($td);
?>
```

この例は、$encrypted_data に文字列として 暗号化されたデータを取得します。詳細な例については、 mcrypt_module_open() を参照してください。

---

mcrypt_cbc

(PHP 4, PHP 5)

mcrypt_cbc — CBC モードでデータを暗号化/復号化する

説明

```
string mcrypt_cbc ( int $cipher , string $key , string $data , int $mode [, string $iv ] )
string mcrypt_cbc ( string $cipher , string $key , string $data , int $mode [, string $iv ] )
```

最初のプロトタイプは libmcrypt 2.2.x とリンクした場合、2 番目は、 libmcrypt 2.4.x とリンクした場合のものです。 mode は **MCRYPT_ENCRYPT** あるいは **MCRYPT_DECRYPT** のいずれかです。

この関数は使用すべきではありません。代替となる関数については mcrypt_generic() および mdecrypt_generic() を参照ください。

---

mcrypt_cfb

(PHP 4, PHP 5)

mcrypt_cfb — CFB モードでデータを暗号化/復号化する

説明

```
string mcrypt_cfb ( int $cipher , string $key , string $data , int $mode , string $iv )
string mcrypt_cfb ( string $cipher , string $key , string $data , int $mode [, string $iv ] )
```

最初のプロトタイプは libmcrypt 2.2.x とリンクした場合、2 番目は、 libmcrypt 2.4.x とリンクした場合のものです。 mode は **MCRYPT_ENCRYPT** あるいは **MCRYPT_DECRYPT** のいずれかです。

この関数は使用すべきではありません。代替となる関数については mcrypt_generic() および mdecrypt_generic() を参照ください。

---

mcrypt_create_iv

(PHP 4, PHP 5)

mcrypt_create_iv — 乱数ソースから初期化ベクトル(IV)を生成する

説明

```
string mcrypt_create_iv ( int $size [, int $source ] )
```

mcrypt_create_iv() は、IV を生成するために 使用されます。

パラメータ size で IV のサイズを、また パラメータ source (デフォルトはランダムな値) で IV のソースを指定します。

source には、**MCRYPT_RAND** (システムの乱数生成器)、 **MCRYPT_DEV_RANDOM** (/dev/random からデータを読む)および **MCRYPT_DEV_URANDOM**

(/dev/urandom からデータを読む) を指定できます。 Windows でサポートされているのは **MCRYPT_RAND** のみです。なぜなら、Windows には(当然) /dev/random あるいは /dev/urandom が存在しないからです。

> **注意**: **MCRYPT_RAND** を使用する場合、乱数生成器を初期化するために、 必ず **mcrypt_create_iv()** の前に srand() をコールしてください。rand() のように、自動的に 初期化されるわけではありません。

Example#1 mcrypt_create_iv() の例

```
<?php
    $size = mcrypt_get_iv_size(MCRYPT_CAST_256, MCRYPT_MODE_CFB);
    $iv = mcrypt_create_iv($size, MCRYPT_DEV_RANDOM);
?>
```

IV は、単に暗号化ルーチンに異なる初期値を与えるためだけのものです。 この IV は、たとえ要求されていたとしても秘密にしておく必要はありません。 暗号化したデータとともに IV を送信したとしても、セキュリティを損ねる ことはありません。

このトピックについてのより詳細な情報は » http://www.ciphersbyritter.com/GLOSSARY.HTM#IV、» http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/co0409.htm および Applied Cryptography by Schneier (ISBN 0-471-11709-9) の 9.3 節にあります。

---

mcrypt_decrypt

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_decrypt — 指定したパラメータで暗号化されたテキストを復号化する

説明

string **mcrypt_decrypt** (string $cipher , string $key , string $data , string $mode [, string $iv])

mcrypt_decrypt() はデータを復号化し、 復号化されたデータを返します。

cipher には、アルゴリズム名を表す定数 MCRYPT_暗号名 の一つを文字列で指定します。

key は、データを暗号化する際のキーです。 必要なキー長よりも短い場合には、'¥0' で埋められます。

data は、指定した暗号およびモードで復号化 されます。データの大きさが n * blocksize でない場合、データは、 '¥0' で埋められます。

mode には、"ecb", "cbc", "cfb", "ofb", "nofb" ,"stream" のどれかを用いて定数 MCRYPT_MODE_モード名を 指定します。

パラメータ IV は、CBC, CFB, OFB モードおよび STREAM モードのいくつかのアルゴリズムの初期化の際に使用されます。 アルゴリズムで必要とする IV を指定しない場合、この関数は警告を発生し、 全てのバイトを '\0' に設定した IV を使用します。

---

mcrypt_ecb

(PHP 4, PHP 5)

mcrypt_ecb — 非推奨: ECB モードでデータを暗号化/復号化する

説明

string **mcrypt_ecb** (int $cipher , string $key , string $data , int $mode)
string **mcrypt_ecb** (string $cipher , string $key , string $data , int $mode [, string $iv])

最初のプロトタイプは libmcrypt 2.2.x とリンクした場合、2 番目は libmcrypt 2.4.x とリンクした場合です。 mode は **MCRYPT_ENCRYPT** あるいは **MCRYPT_DECRYPT** のいずれかとなります。

この関数は非推奨であり、使用すべきではありません。代替機能については mcrypt_generic() および mdecrypt_generic() を参照ください。

---

mcrypt_enc_get_algorithms_name

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_algorithms_name — オープンされたアルゴリズムの名前を返す

説明

string **mcrypt_enc_get_algorithms_name** (resource $td)

この関数はアルゴリズム名を返します。

Example#1 mcrypt_enc_get_algorithms_name() の例

```
<?php
$td = mcrypt_module_open(MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
echo mcrypt_enc_get_algorithms_name($td). "¥n";

$td = mcrypt_module_open('cast-256', '', MCRYPT_MODE_CFB, '');
echo mcrypt_enc_get_algorithms_name($td). "¥n";
?>
```

上の例の出力は以下となります。

```
CAST-256
CAST-256
```

---

mcrypt_enc_get_block_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_block_size — オープンされたアルゴリズムのブロックサイズを返す

説明

int **mcrypt_enc_get_block_size** (resource $td)

この関数は、暗号化記述子 td で指定したアルゴリズムの ブロック長をバイト単位で返します。

---

mcrypt_enc_get_iv_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_iv_size — オープンされたアルゴリズムの IV の大きさを返す

説明

int **mcrypt_enc_get_iv_size** (resource $td)

この関数は、暗号化記述子で指定したアルゴリズムの iv の大きさを バイト単位で返します。この関数が '0' を返す場合、IV は そのアルゴリズムでは無視されます。IV は、cbc、cfb、ofbモード およびストリームモードのいくつかのアルゴリズムで使用されます。

---

mcrypt_enc_get_key_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_key_size — オープンされたモードでサポートされる最大キー長を返す

説明

int **mcrypt_enc_get_key_size** (resource $td)

この関数は、暗号化記述子 td で指定した アルゴリズムでサポートされる最大キー長をバイト単位で返します。

---

mcrypt_enc_get_modes_name

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_modes_name — オープンされたモードの名前を返す

説明

string **mcrypt_enc_get_modes_name** (resource $td)

この関数はモード名を返します。

Example#1 mcrypt_enc_get_modes_name() の例

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
    echo mcrypt_enc_get_modes_name($td). "¥n";

    $td = mcrypt_module_open ('cast-256', '', 'ecb', '');
    echo mcrypt_enc_get_modes_name($td). "¥n";
?>
```

出力は以下のようになります。

```
CFB
ECB
```

---

mcrypt_enc_get_supported_key_sizes

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_get_supported_key_sizes — オープンされたアルゴリズムでサポートされるキー長を配列にして返す

説明

array **mcrypt_enc_get_supported_key_sizes** (resource $td)

暗号化記述子で指定したアルゴリズムでサポートされるキー長を配列と して返します。返された配列が空の場合、1 と
mcrypt_enc_get_key_size() の値の間のすべての キー長がアルゴリズムでサポートされます。

Example#1 mcrypt_enc_get_supported_key_sizes() の例

```
<?php
    $td = mcrypt_module_open('rijndael-256', '', 'ecb', '');
    var_dump(mcrypt_enc_get_supported_key_sizes($td));
?>
```

出力は以下のようになります。

```
array(3) {
  [0]=>
  int(16)
  [1]=>
  int(24)
  [2]=>
  int(32)
}
```

---

mcrypt_enc_is_block_algorithm_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_is_block_algorithm_mode — オープンされたモードの暗号がブロックモードで動作するかどうかを調べる

説明

bool **mcrypt_enc_is_block_algorithm_mode** (resource $td)

この関数は、モードがブロックアルゴリズムを使用している場合に **TRUE**、 その他の場合に **FALSE** を返します(例えば、streamでは **FALSE**、 cbc, cfb, ofb では **TRUE**)。

---

mcrypt_enc_is_block_algorithm

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_is_block_algorithm — オープンされたモードの暗号がブロックアルゴリズムであるかどうかを調べる

説明

bool **mcrypt_enc_is_block_algorithm** (resource $td)

この関数は、アルゴリズムがブロックアルゴリズムである場合に **TRUE**、 ストリームアルゴリズムである場合に **FALSE** を返します。

---

mcrypt_enc_is_block_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_is_block_mode — オープンされたモードがブロック出力を行うかどうかを調べる

説明

bool **mcrypt_enc_is_block_mode** (resource $td)

この関数は、バイトブロックを出力するモードの場合に **TRUE**、 バイト出力を行うモードの場合に **FALSE** を返します(例: cbc と ecb の場合に **TRUE**、cfb と stream の場合に **FALSE**)。

---

mcrypt_enc_self_test

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_enc_self_test — オープンしたモジュールのセルフテストを実行する

説明

int **mcrypt_enc_self_test** (resource $td)

この関数は、記述子 td で指定したアルゴリズムの セルフテストを実行します。セルフテストが成功した場合は **FALSE** を返します。エラーの場合は **TRUE** を返します。

---

mcrypt_encrypt

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_encrypt — 指定したパラメータでプレーンテキストを暗号化する

説明

string **mcrypt_encrypt** (string $cipher , string $key , string $data , string $mode [, string $iv])

mcrypt_encrypt() はデータを暗号化し、暗号化さ れたデータを返します。

cipher には、アルゴリズム名を表す定数 MCRYPT_暗号名の一つを文字列で指定します。

key は、データを暗号化する際のキーです。必 要なキー長よりも小さい場合には、'\0' で埋められます。キーにはASCII 文字列を使わない方が良いです。文字列からキーを生成するにはmhash関 数を使用することが推奨されます。

data は、指定した暗号およびモードで暗号化 されます。データの大きさが n * blocksize でない場合、データは、 '\0'で埋められます。返される暗号化されたテキストは、 data んで指定したデータの大きさよりも大きく なる可能性があります。

mode には、"ecb", "cbc", "cfb", "ofb", "nofb" ,"stream" のどれかを用いて定数 MCRYPT_MODE_モード名を 指定します。

パラメータ IV は、CBC, CFB, OFB モードおよ びSREAMモードのいくつかのアルゴリズムの初期化の際に使用されます。 アルゴリズムで必要とするIVを指定しない場合、この関数は警告を発生 し、全てのバイトを'\0'に設定したIVを使用します。

Example#1 mcrypt_encrypt() の例

```
<?php
$iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MCRYPT_RAND);
$key = "This is a very secret key";
$text = "Meet me at 11 o'clock behind the monument.";
echo strlen ($text)."¥n";

$crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
echo strlen ($crypttext)."¥n";
?>
```

上記の例の出力は次のようになります。

```
42
64
```

---

mcrypt_generic_deinit

(PHP 4 >= 4.0.7, PHP 5)

mcrypt_generic_deinit — 暗号化モジュールを終了する

説明

bool **mcrypt_generic_deinit** (resource $td)

この関数は、暗号化記述子(td)で指定した 暗号化を終了します。すべてのバッファを消去しますが、モジュールは 閉じません。
mcrypt_module_close() をコールする 必要があります(しかし、スクリプトの終了時に PHP が自動でこれを 行います)。エラー時に **FALSE**、成功時に **TRUE** を返します。

mcrypt_module_open() の例、および mcrypt_generic_init() も参照ください。

---

mcrypt_generic_end

(PHP 4 >= 4.0.2, PHP 5 <= 5.1.6)

mcrypt_generic_end — 暗号処理を終了する

説明

bool **mcrypt_generic_end** (resource $td)

警告

この関数は非推奨です。かわりに mcrypt_generic_deinit() を使用してください。 mcrypt_module_close() とともに使用すると、 複数バッファの開放によりクラッシュを引き起こすことがあります。

この関数は、暗号化記述子(td) で指定した暗号化処理を終了します。 この関数は全てのバッファをクリアし、使用した全てのモジュールを クローズします。エラーの場合に **FALSE**、成功時に **TRUE** を返します。

---

mcrypt_generic_init

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_generic_init — 暗号化に必要な全てのバッファを初期化する

説明

int **mcrypt_generic_init** (resource $td , string $key , string $iv)

キーの最大長は、mcrypt_enc_get_key_size() を コールした際に得られるキー長であり、この値より小さな値は全て有効です。 IV は通常そのアルゴリズムのブロック長と同じ大きさですが、 mcrypt_enc_get_iv_size() をコールする ことにより、その大きさを得る必要があります。IV は ECB では 無視されます。IV は CFB、CBC、STREAM、nOFB では必須です。 IV はランダムで一意である必要があります(しかし、秘密ではある必要は ありません)。暗号化と復号化で同じ IV を使用する必要があります。 IV を使用したくない場合にはゼロに設定しますが、これは推奨されません。

この関数は、エラー時に負の値を返します。キー長が不正な場合に -3、メモリの確保に問題があった場合に -4、そしてそれ以外の返り値は その他のエラーとなります。エラーが警告を発生させた場合、それが 表示されます。間違ったパラメータが渡された場合には FALSE が返されます。

mcrypt_generic() または mdecrypt_generic() をコールする前に、 常にこの関数をコールする必要があります。

mcrypt_module_open() の例を参照ください。

---

mcrypt_generic

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_generic — データを暗号化する

説明

string **mcrypt_generic** (resource $td , string $data)

この関数は、データを暗号化します。データ長を n * blocksize とする ために、データは "¥0" で埋められます。 この関数は、暗号化されたデータを返します。 データのパディングが行われるため、返される文字列の長さは 入力よりも長いことがあることに注意してください。

暗号化したデータをデータベースに格納する場合は、mcrypt_generic が返す文字列全てを格納することを忘れないでください。さもないと、 文字列を適切に復号化できなくなります。もとの文字列が 10 文字で ブロックサイズが 8 (ブロックサイズを調べるには mcrypt_enc_get_block_size() を使用します) だった場合、データベースのフィールドには少なくとも 16 文字が 必要です。mdecrypt_generic() が返す文字列も 16 文字となることに注意しましょう。埋められた文字を取り除くには... rtrim()($str, "\0") を使用します。

もし MySQL データベースにデータを格納する場合は、 varchar フィールドに値を挿入する際に末尾のスペースが取り除かれることを 覚えておきましょう。暗号化されたデータの最後にスペース(ASCII 32)が 含まれていた場合、この処理によってデータが破壊されてしまいます。 かわりに tinyblob/tinytext (あるいはより大きな) フィールドを 使用してください。

この関数をコールする前には、常にキーと IV を用いて mcrypt_generic_init() で暗号化ハンドルを 初期化しておくべきです。暗号化が終了したら、 mcrypt_generic_deinit() をコールして 暗号化バッファを開放すべきです。使用例は mcrypt_module_open() を参照ください。

mdecrypt_generic()、 mcrypt_generic_init() および mcrypt_generic_deinit() も参照ください。

---

mcrypt_get_block_size

(PHP 4, PHP 5)

mcrypt_get_block_size — 指定した暗号のブロックサイズを得る

説明

int **mcrypt_get_block_size** (int $cipher)
int **mcrypt_get_block_size** (string $cipher , string $module)

最初のプロトタイプは libmcrypt 2.2.x とリンクした場合であり、2 番目 のものは libmcrypt 2.4.x あるいは 2.5.x とリンクした場合です。

mcrypt_get_block_size()は、指定した cipher のブロック長を返します (暗号化モードと組み合わせます)。

mcrypt_get_block_size()は、一つまたは二つの引数、 つまり、cipher と module を引数とし、長さをバイト数で返します。

mcrypt_get_key_size()も参照ください。

---

mcrypt_get_cipher_name

(PHP 4, PHP 5)

mcrypt_get_cipher_name — 指定した暗号の名前を得る

説明

string **mcrypt_get_cipher_name** (int $cipher)
string **mcrypt_get_cipher_name** (string $cipher)

mcrypt_get_cipher_name() は、指定した暗号(cipher) の名前を得るために使用されます。

mcrypt_get_cipher_name() は、暗号の番号 (libmcrypt 2.2.x)または暗号名(libmcrypt 2.4.x 以降)を引数とし、 暗号名を返します。指定した暗号が存在しない場合は FALSE を返します。

Example#1 mcrypt_get_cipher_name() の例

```
<?php
    $cipher = MCRYPT_TripleDES;

    echo mcrypt_get_cipher_name($cipher);
?>
```

上の例の出力は以下となります。

3DES

---

mcrypt_get_iv_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_get_iv_size — 指定した暗号/モードの組み合わせに属する IV の大きさを返す

説明

int **mcrypt_get_iv_size** (string $cipher , string $mode)

mcrypt_get_iv_size() は初期化ベクトル(IV)の 大きさをバイト数で返します。エラーの際には **FALSE** を返します。IV が指定した暗号/モードで無視される場合には、ゼロが返されます。

cipher には、アルゴリズム名を表す定数 MCRYPT_暗号名の一つを文字列で指定します。

mode は "ecb"、"cbc"、"cfb"、"ofb"、 "nofb"、"stream" のどれか、あるいは定数 MCRYPT_MODE_モード名です。 ECB モードでは IV は要求されないため、IV は無視されます。 暗号化と復号化の際には、同じ IV(開始位置)を必要とします。さもないと 暗号化処理は失敗します。

mcrypt_enc_get_iv_size() 関数を使用するほうが より有用です。これは mcrypt_module_open() が返すリソースを使用します。

Example#1 mcrypt_get_iv_size() の例

```
<?php
    echo mcrypt_get_iv_size(MCRYPT_CAST_256, MCRYPT_MODE_CFB) . "¥n";

    echo mcrypt_get_iv_size('des', 'ecb') . "¥n";
?>
```

mcrypt_get_block_size()、 mcrypt_enc_get_iv_size() および mcrypt_create_iv() も参照ください。

---

mcrypt_get_key_size

(PHP 4, PHP 5)

mcrypt_get_key_size — 指定した暗号のキーの長さを得る

説明

int **mcrypt_get_key_size** (int $cipher)
int **mcrypt_get_key_size** (string $cipher , string $module)

最初のプロトタイプは libmcrypt 2.2.x とリンクした場合であり、2 番目の ものは libmcrypt 2.4.x あるいは 2.5.x とリンクした場合です。

mcrypt_get_key_size() は、 cipher で(暗号化モードとあわせて)指定した キーの長さを得るために使用されます。

この例では、libmcrypt 2.4.x および 2.5.x とリンクした場合の この関数の使用法を示します。 mcrypt_enc_get_key_size() 関数を使用するほうが より有用です。これは mcrypt_module_open() が返すリソースを使用します。

Example#1 mcrypt_get_block_size() の例

```
<?php
    echo mcrypt_get_key_size('tripledes', 'ecb');
?>

Prints:
24
```

mcrypt_get_block_size()、 **mcrypt_end_get_key_size()** および mcrypt_encrypt() も参照ください。

---

mcrypt_list_algorithms

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_list_algorithms — サポートされる全ての暗号を配列として取得する

説明

array **mcrypt_list_algorithms** ([string $lib_dir])

mcrypt_list_algorithms() は、 lib_dir にあるサポートされる暗号を全て 取得するために使用します。

mcrypt_list_algorithms() はオプションの パラメータとして lib_dir をとり、全ての アルゴリズムがある場所のディレクトリを指定することが可能です。 指定されない場合には、php.ini ディレクティブ mcrypt.algorithms_dir の値が使用されます。

Example#1 mcrypt_list_algorithms() の例

```
<?php
    $algorithms = mcrypt_list_algorithms("/usr/local/lib/libmcrypt");

    foreach ($algorithms as $cipher) {
        echo "$cipher<br />¥n";
```

```
    }
?>
```

上記の例は、"/usr/local/lib/libmcrypt" ディレクトリにある全ての サポートされるアルゴリズムの一覧を表示します。

---

mcrypt_list_modes

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_list_modes — サポートされる全てのモードの配列を取得する

説明

array **mcrypt_list_modes** ([string $lib_dir])

mcrypt_list_modes() は、 lib_dir にあるサポートされる全てのモードを 取得するために使用されます。

mcrypt_list_modes() はオプションのパラメータと して全てのモードがある場所をとります。指定されない場合には、 php.ini ディレクティブの mcrypt.modes_dir の値が使用されます。

Example#1 mcrypt_list_modes() の例

```
<?php
     $modes = mcrypt_list_modes();

     foreach ($modes as $mode) {
         echo "$mode <br />¥n";
     }
?>
```

上の例は、デフォルトのモードディレクトリにあるサポートされる 全てのアルゴリズムの一覧を出力します。ini ディレクティブの mcrypt.modes_dir でこれが設定されていない場合、mcrypt の デフォルトディレクトリ(/usr/local/lib/libmcrypt)が使用されます。

---

mcrypt_module_close

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_close — mcrypt モジュールを閉じる

説明

bool **mcrypt_module_close** (resource $td)

この関数は、指定した暗号化ハンドルを閉じます。

使用例は mcrypt_module_open() を参照ください。

---

mcrypt_module_get_algo_block_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_get_algo_block_size — 指定したアルゴリズムのブロック長を返す

説明

int **mcrypt_module_get_algo_block_size** (string $algorithm [, string $lib_dir])

この関数は、指定したアルゴリズムでサポートされるブロック長を バイト単位で返します。オプションのパラメータ lib_dir により、システム上での モードモジュールの位置を指定することが可能です。

---

mcrypt_module_get_algo_key_size

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_get_algo_key_size — オープンされたモードでサポートされる最大キー長を返す

説明

int **mcrypt_module_get_algo_key_size** (string $algorithm [, string $lib_dir])

この関数は、指定したアルゴリズムでサポートされる最大キー長を バイト単位で返します。オプションのパラメータ lib_dir により、システム上での モードモジュールの位置を指定することが可能です。

---

mcrypt_module_get_supported_key_sizes

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_get_supported_key_sizes — オープンされたアルゴリズムでサポートされるキーのサイズを配列として返す

説明

array **mcrypt_module_get_supported_key_sizes** (string $algorithm [, string $lib_dir])

指定したアルゴリズムでサポートされるキーのサイズを配列で返します。 この関数が空の配列を返した場合、1 から mcrypt_module_get_algo_key_size() の返り値の間のすべてのサイズのキーがアルゴリズムでサポートされます。 オプションのパラメータ lib_dir により、 システムで mode モジュールがある場所を指定することが可能です。

オープン暗号化モジュールで使用される mcrypt_enc_get_supported_key_sizes() も参照ください。

---

mcrypt_module_is_block_algorithm_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_is_block_algorithm_mode — 指定したモジュールがブロックアルゴリズムであるかどうかを返す

説明

bool **mcrypt_module_is_block_algorithm_mode** (string $mode [, string $lib_dir])

この関数は、mode がブロックアルゴリズムを使用する場合に **TRUE**、 その他の場合に **FALSE** を返します。(例: stream の場合に **FALSE**、 cbc、cfb、ofb の場合に **TRUE**) オプションのパラメータ lib_dir により、 システム上での mode モジュールの位置を指定することが可能です。

---

mcrypt_module_is_block_algorithm

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_is_block_algorithm — 指定したアルゴリズムがブロックアルゴリズムであるかを調べる

説明

bool **mcrypt_module_is_block_algorithm** (string $algorithm [, string $lib_dir])

この関数は、指定したアルゴリズムがブロックアルゴリズムの場合に **TRUE**、ストリームアルゴリズムの場合に **FALSE** を返します。 オプションのパラメータ lib_dir により、 システム上での algorithm モジュールの位置を指定することが可能です。

---

mcrypt_module_is_block_mode

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_is_block_mode — 指定したモードがブロック出力を行うかどうかを返す

説明

bool **mcrypt_module_is_block_mode** (string $mode [, string $lib_dir])

この関数は、ブロック出力を行うモードの場合に **TRUE**、バイトのみを 出力する場合に **FALSE** を返します(例: cbc と ecb の場合に **TRUE**、 cfb と streamの場合に **FALSE**)。オプションのパラメータ lib_dir により、システム上での mode モジュールの 位置を指定することが可能です。

---

mcrypt_module_open

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_open — 使用するアルゴリズムおよびモードのモジュールをオープンする

説明

resource **mcrypt_module_open** (string $algorithm , string $algorithm_directory , string $mode , string $mode_directory)

この関数は、使用するアルゴリズムおよびモードのモジュールをオープン します。アルゴリズム名は、"twofish" または定数 MCRYPT_暗号名により algorithm で指定します。 ライブラリは mcrypt_module_close() をコールする ことによりクローズされます。通常、この関数は暗号化記述子を返し、 エラーの際に **FALSE** を返します。

暗号化モジュールの位置を指定する際には algorithm_directory および mode_directory が使用されます。 ディレクトリ名を指定した場合には、これが使用されます。 これらの一つに空の文字列("")を指定した場合、ini ディレクティブの mcrypt.algorithms_dir または mcrypt.modes_dir に設定された値が使用されます。 これらが設定されていない場合、libmcrypt をコンパイルした際の デフォルトのディレクトリ (通常は/usr/local/lib/libmcrypt)が 使用されます。

Example#1 mcrypt_module_open() の例

```
<?php
    $td = mcrypt_module_open(MCRYPT_DES, '',
        MCRYPT_MODE_ECB, '/usr/lib/mcrypt-modes');

    $td = mcrypt_module_open('rijndael-256', '', 'ofb', '');
?>
```

上の例の最初の行では、デフォルトのディレクトリから DES 暗号化を、 そして /usr/lib/mcrypt-modes から EBC モードの オープンを試みます。2 行目の例では暗号化形式とモードを文字列で 指定しています。これは、libmcrypt 2.4.x あるいは 2.5.x に対して 拡張モジュールがリンクされている場合にのみ使用可能です。

Example#2 mcrypt_module_open() を暗号化で使用する

```
<?php
    /* 暗号モジュールをオープンします */
    $td = mcrypt_module_open('rijndael-256', '', 'ofb', '');

    /* IV を作成し、キー長を定義します。Windows では、かわりに
     * MCRYPT_RAND を使用します */
    $iv = mcrypt_create_iv(mcrypt_enc_get_iv_size($td), MCRYPT_DEV_RANDOM);
    $ks = mcrypt_enc_get_key_size($td);

    /* キーを作成します */
    $key = substr(md5('very secret key'), 0, $ks);

    /* 暗号化処理を初期化します */
    mcrypt_generic_init($td, $key, $iv);

    /* データを暗号化します */
    $encrypted = mcrypt_generic($td, 'This is very important data');

    /* 暗号化ハンドラを終了します */
    mcrypt_generic_deinit($td);

    /* 復号化用の暗号モジュールを初期化します */
    mcrypt_generic_init($td, $key, $iv);

    /* 暗号化された文字列を復号します */
    $decrypted = mdecrypt_generic($td, $encrypted);

    /* 復号化ハンドルを終了し、モジュールを閉じます */
    mcrypt_generic_deinit($td);
    mcrypt_module_close($td);

    /* 文字列を表示します */
    echo trim($decrypted) . "¥n";
?>
```

mcrypt_module_close()、 mcrypt_generic()、 mdecrypt_generic()、 mcrypt_generic_init() および mcrypt_generic_deinit() も参照ください。

---

mcrypt_module_self_test

(PHP 4 >= 4.0.2, PHP 5)

mcrypt_module_self_test — 指定したモジュールのセルフテストを実行する

説明

bool **mcrypt_module_self_test** (string $algorithm [, string $lib_dir])

この関数は、指定したアルゴリズムのセルフテストを実行します。 オプションのパラメータ lib_dir により、 システム上での algorithm モジュールの位置を指定することが可能です。

この関数は、セルフテストが成功した場合 **TRUE**、失敗した場合に **FALSE** を返します。

Example#1 mcrypt_module_self_test() の例

```
<?php
var_dump(mcrypt_module_self_test(MCRYPT_RIJNDAEL_128)) . "¥n";
var_dump(mcrypt_module_self_test(MCRYPT_BOGUS_CYPHER));
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(false)
```

---

mcrypt_ofb

(PHP 4, PHP 5)

mcrypt_ofb — OFB モードでデータを暗号化/復号化する

説明

string **mcrypt_ofb** (int $cipher , string $key , string $data , int $mode , string $iv)
string **mcrypt_ofb** (string $cipher , string $key , string $data , int $mode [, string $iv])

最初のプロトタイプは libmcrypt 2.2.x とリンクした場合、2 番目は libmcrypt 2.4.x 以降とリンクした場合です。 mode は、**MCRYPT_ENCRYPT** あるいは **MCRYPT_DECRYPT** のいずれかです。

この関数はもはや使用すべきではありません。代替関数として mcrypt_generic() および mdecrypt_generic() を参照ください。

---

mdecrypt_generic

(PHP 4 >= 4.0.2, PHP 5)

mdecrypt_generic — データを復号化する

説明

string **mdecrypt_generic** (resource $td , string $data)

この関数は、データを復号化します。データのパディングが行われるため、 返される文字列の長さは暗号化前の文字列よりも長くなる可能性があることに 注意してください。

Example#1 mdecrypt_generic() の例

```
<?php
    /* データ */
    $key = 'this is a very long key, even too long for the cipher';
    $plain_text = 'very important data';

    /* モジュールをオープンし、IV を作成します */
    $td = mcrypt_module_open('des', '', 'ecb', '');
    $key = substr($key, 0, mcrypt_enc_get_key_size($td));
    $iv_size = mcrypt_enc_get_iv_size($td);
    $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);

    /* 暗号化ハンドルを初期化します */
    if (mcrypt_generic_init($td, $key, $iv) != -1) {

        /* データを暗号化します */
        $c_t = mcrypt_generic($td, $plain_text);
        mcrypt_generic_deinit($td);

        /* 復号化のため、バッファを再度初期化します */
        mcrypt_generic_init($td, $key, $iv);
        $p_t = mdecrypt_generic($td, $c_t);

        /* 後始末をします */
        mcrypt_generic_deinit($td);
        mcrypt_module_close($td);
    }

    if (strncmp($p_t, $plain_text, strlen($plain_text)) == 0) {
        echo "ok¥n";
    } else {
        echo "error¥n";
    }
?>
```

上記の例は、暗号化前のデータと復号化したデータと同じであるかどうかを 調べる方法を示すものです。データを復号化する前に mcrypt_generic_init() で暗号化バッファを再度 初期化しておくことが重要です。

この関数をコールする前に、常に mcrypt_generic_init() でキーと IV を使用して 復号化ハンドルを初期化しておくべきです。暗号化が終了した際には mcrypt_generic_deinit() をコールして暗号化 バッファを開放すべきです。 mcrypt_module_open() の例を参照ください。

mcrypt_generic()、 mcrypt_generic_init() および mcrypt_generic_deinit() も参照ください。

目次

- mcrypt_module_close — mcrypt モジュールを閉じる
- mcrypt_module_get_algo_block_size — 指定したアルゴリズムのブロック長を返す
- mcrypt_module_get_algo_key_size — オープンされたモードでサポートされる最大キー長を返す
- mcrypt_module_get_supported_key_sizes — オープンされたアルゴリズムでサポートされるキーのサイズを配列として返す
- mcrypt_module_is_block_algorithm_mode — 指定したモジュールがブロックアルゴリズムであるかどうかを返す
- mcrypt_module_is_block_algorithm — 指定したアルゴリズムがブロックアルゴリズムであるかを調べる
- mcrypt_module_is_block_mode — 指定したモードがブロック出力を行うかどうかを返す
- mcrypt_module_open — 使用するアルゴリズムおよびモードのモジュールをオープンする
- mcrypt_module_self_test — 指定したモジュールのセルフテストを実行する
- mcrypt_ofb — OFB モードでデータを暗号化/復号化する
- mdecrypt_generic — データを復号化する

MCVE (Monetra) 支払い関数

導入

以下の関数は MCVE (Monetra) API (libmonetra、以前は libmcve と呼ばれていました) のインターフェイスで、PHP スクリプトから MCVE/Monetra を直接使用することができるようになります。 MCVE/Monetra は Main Street Softworks のソリューションであり、Linux/Unix/MacOSX/Windows 上で クレジットカード/デビットカード/ギフトカード を直接処理することが可能です (» http://www.mainstreetsoftworks.com/)。 MCVE/Monetra は、*nix マシン、モデムまたはインターネット接続により、 直接クレジットカードを処理することを可能とします (Authorize.Net や Pay Flow Pro のような付加的なサービスは不要と なります)。 PHP で MCVE/Monetra モジュールを使用することにより、PHP スクリプトで MCVE/Monetra を通じて直接クレジットカード処理が可能となります。 以下のリファレンスに処理の概要を示します。

> **注意**: MCVE/Monetra は、RedHat の CCVS を置き換えるものです。MCVEは 2001 年末に RedHat に採用され、全ての既存システムを MCVE プラットフォームに移行することになりました。
>
> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 5.1.0.
>
> **注意**: この拡張モジュールは Windows 環境では利用できません。

インストール手順

MCVE (Monetra) サポートを PHP で有効にするには、まず、システムの LibMonetra (以前は libmcve と呼ばれていました) インストールディレクトリを 調べてください。MVCE/Monetra サポートを直接 PHP に組み込むには、 --with-mcve オプションを 指定して PHP の configure を実行する必要があります。 libmonetra のインストール場所へのパスを指定せずにこのオプションを 使用した場合、PHP は デフォルトの LibMonetra インストールパス (/usr/local) を探します。Monetra (MCVE) が標準以外の場所にある場合、 --with-mcve=$mcve_path を指定して configure を実行してください。ただし、$mcve_path は MCVE/Monetra を インストールしたパスです。MCVE/Monetra サポートは、$mcve_path/lib および $mcve_path/include が存在し、include ディレクトリの下に mcve.h あるいは monetra.h、 lib ディレクトリの下にlibmcve.so や libmcve.a や libmonetra.so や libmonetra.a が存在しなければならないことに 注意してください。

MCVE/Monetra サポートをモジュールとしてインストールしたい場合は、 もし PEAR のバージョン 1.4.0 以降が稼動しているなら、 PECL リポジトリを利用して 'pecl install mcve' コマンドを発行します。

MCVE/Monetra はサーバ/クライアントを完全に分離しているため、 PHP で MCVE サポートを実行する際に追加するべきものはありません。 PHP で MCVE/Monetra 拡張モジュールをテストするには、PHP API を使用し、 IP の場合は testbox.monetra.com のポート 8333、SSL の場合はポート 8444 に接続します。ユーザ名として 'vitale'、 パスワードとして 'test' を使用してください。 テスト機能に関する詳細な情報については » http://www.mainstreetsoftworks.com/ を参照ください。

参考

MCVE/Monetra の PHP API に関する追加のドキュメントについては、 » http://www.mainstreetsoftworks.com/documentation.html にあります。 Main Streetのドキュメントは完全なものであり、関数のリファレンスは まず最初にここを参照するべきです。

リソース型

この拡張モジュールでは MCVE_CONN リソースを定義しています。これは m_initconn() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

M_PENDING (integer)
M_DONE (integer)
M_ERROR (integer)
M_FAIL (integer)
M_SUCCESS (integer)

---

m_checkstatus

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_checkstatus — トランザクションが完了したかどうかを確かめる

説明

int **m_checkstatus** (resource $conn , int $identifier)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_completeauthorizations

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_completeauthorizations — キューの中の認証済み件数を数え、その ID の配列を返す

説明

int **m_completeauthorizations** (resource $conn , int &$array)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

array

Its description

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

m_connect

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_connect — MCVE との接続を確立する

説明

int **m_connect** (resource $conn)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

返り値

---

m_connectionerror

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_connectionerror — 接続が失敗した理由をテキストで取得する

説明

string **m_connectionerror** (resource $conn)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

返り値

---

m_deletetrans

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_deletetrans — MCVE_CONN 構造体から、指定したトランザクションを削除する

説明

bool **m_deletetrans** (resource $conn , int $identifier)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_destroyconn

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_destroyconn — 接続および MCVE_CONN 構造体を破壊する

説明

bool **m_destroyconn** (resource $conn)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

返り値

TRUE を返します。

---

m_destroyengine

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_destroyengine — IP/SSL 接続に関連付けられたメモリを開放する

説明

void **m_destroyengine** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

返り値

値を返しません。

---

m_getcell

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_getcell — カンマ区切りの応答から、カラム名を指定してセルを取得する

説明

string **m_getcell** (resource $conn , int $identifier , string $column , int $row)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

column

row

返り値

---

m_getcellbynum

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_getcellbynum — カンマ区切りの応答から、カラム番号を指定してセルを取得する

説明

string **m_getcellbynum** (resource $conn , int $identifier , int $column , int $row)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

column

row

返り値

---

m_getcommadelimited

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_getcommadelimited — MCVE が返すデータを、もとのカンマ区切り形式のままで取得する

説明

string **m_getcommadelimited** (resource $conn , int $identifier)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_getheader

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_getheader — カンマ区切りの応答から、カラム名を取得する

説明

string **m_getheader** (resource $conn , int $identifier , int $column_num)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

column_num

返り値

---

m_initconn

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_initconn — MCVE_CONN 構造体を作成し、初期化する

説明

resource ***m_initconn*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

MCVE_CONN リソースを返します。

参考

- *m_destroyconn()*

---

m_initengine

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_initengine — IP/SSL 通信のためのクライアントの準備をする

説明

int ***m_initengine*** *(string $location)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

location

返り値

---

m_iscommadelimited

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_iscommadelimited — 応答がカンマ区切りかどうかを調べる

説明

int ***m_iscommadelimited*** *(resource $conn , int $identifier)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() *が返す MCVE_CONN リソース。*

identifier

返り値

---

m_maxconntimeout

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_maxconntimeout — API が MCVE への接続を試みる時間の最大値

説明

bool **m_maxconntimeout** (resource $conn , int $secs)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

secs

返り値

---

m_monitor

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_monitor — MCVE との通信（データの送受信）を非ブロックモードで行う

説明

int **m_monitor** (resource $conn)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

返り値

---

m_numcolumns

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_numcolumns — カンマ区切りの応答の中のカラム数を返す

説明

int **m_numcolumns** (resource $conn , int $identifier)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_numrows

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_numrows — カンマ区切りの応答の中の行数を返す

説明

int **m_numrows** (resource $conn , int $identifier)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

m_parsecommadelimited

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_parsecommadelimited — カンマ区切りの応答をパースし、m_getcell などが動作するようにする

説明

int **m_parsecommadelimited** (resource $conn , int $identifier)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_responsekeys

(PHP 5 >= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_responsekeys — このトランザクションの応答パラメータとして使用することが可能な キーを表す文字列の配列を返す

説明

array **m_responsekeys** (resource $conn , int $identifier)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_responseparam

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_responseparam — カスタム応答パラメータを取得する

説明

string **m_responseparam** (resource $conn , int $identifier , string $key)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

key

返り値

---

m_returnstatus

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_returnstatus — トランザクションが成功したかどうかを確かめる

説明

int ***m_returnstatus*** *(resource $conn , int $identifier)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_setblocking

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_setblocking — 接続モードを、ブロックモードあるいは非ブロックモードに設定する

説明

int ***m_setblocking*** *(resource $conn , int $tf)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

tf

返り値

---

m_setdropfile

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_setdropfile — Drop-File への接続方法を設定する

説明

int ***m_setdropfile*** *(resource $conn , string $directory)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

directory

返り値

---

m_setip

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_setip — IP での接続方法を設定する

説明

int ***m_setip*** *(resource $conn , string $host , int $port)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

host

port

返り値

---

m_setssl_cafile

(PHP 5 >= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_setssl_cafile — サーバ証明書を検証するための SSL CA (Certificate Authority) ファイルを設定する

説明

int ***m_setssl_cafile*** *(resource $conn , string $cafile)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

cafile

返り値

---

m_setssl_files

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_setssl_files — サーバがクライアント証明書による検証を要求している場合に、証明書のキーファイルを設定する

説明

int ***m_setssl_files*** *(resource $conn , string $sslkeyfile , string $sslcertfile)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

sslkeyfile

sslcertfile

返り値

---

m_setssl

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_setssl — SSL での接続方法を設定する

説明

int ***m_setssl*** *(resource $conn , string $host , int $port)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

host

port

返り値

---

m_settimeout

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_settimeout — （トランザクション単位の）最大の時間を設定する

説明

int **m_settimeout** (resource $conn , int $seconds)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

seconds

返り値

---

m_sslcert_gen_hash

(PECL mcve:5.2.0-5.2.2)

m_sslcert_gen_hash — SSL クライアント証明書の検証のためのハッシュを作成する

説明

string **m_sslcert_gen_hash** (string $filename)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

filename

返り値

---

m_transactionssent

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_transactionssent — 送信バッファが空かどうかを確かめる

説明

int **m_transactionssent** (resource $conn)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

返り値

---

m_transinqueue

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_transinqueue — クライアントキューの中のトランザクション数を返す

説明

int **m_transinqueue** (resource $conn)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

返り値

---

m_transkeyval

(PHP 5 >= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_transkeyval — トランザクションにキー/値のペアを追加する。transparam() の代替関数

説明

int **m_transkeyval** (resource $conn , int $identifier , string $key , string $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

key

value

返り値

---

m_transnew

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_transnew — 新しいトランザクションを開始する

説明

int **m_transnew** (resource $conn)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

返り値

---

m_transsend

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_transsend — トランザクションを終了し、送信する

説明

int **m_transsend** (resource $conn , int $identifier)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

identifier

返り値

---

m_uwait

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_uwait — x マイクロ秒だけ待つ

説明

int **m_uwait** (int $microsecs)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

microsecs

返り値

---

m_validateidentifier

(PHP 5 >= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_validateidentifier — 指定したトランザクションについて、ID の検証を行うかどうか

説明

int **m_validateidentifier** (resource $conn , int $tf)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

tf

返り値

---

m_verifyconnection

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_verifyconnection — 接続の検証を行うために PING を行うかどうかを設定する

説明

bool **m_verifyconnection** (resource $conn , int $tf)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

tf

返り値

---

m_verifysslcert

(PHP 4 >= 4.3.9, PHP 5 <= 5.0.5, PECL mcve:1.0.0-5.2.2)

m_verifysslcert — サーバの ssl 証明書を検証するかどうかを設定する

説明

bool **m_verifysslcert** (resource $conn , int $tf)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

conn

m_initengine() が返す MCVE_CONN リソース。

tf

返り値

目次

Memcache 関数

導入

Memcache モジュールは、memcached に対する手続き型および オブジェクト指向のインターフェイスを提供します。これは非常に効率的な キャッシュデーモンで、動的な web アプリケーションでの データベースの読み込み量を減らすように設計されています。

Memcache モジュールは、セッション ハンドラ (memcache) も提供します。

memcached についてのより詳細な情報は » http://www.danga.com/memcached/ にあります。

要件

このモジュールは、その場でのデータ圧縮機能をサポートするために » zlib の関数を使用しています。 このモジュールをインストールするには Zlib が必要となります。

memcache 拡張モジュールを使用するには、PHP 4.3.3 以降が必要です。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/memcache.

これらの関数を使用するには、 --enable-memcache[=DIR] オプションを 指定し、Memcache のサポートを有効にして PHP をコンパイルする必要があります。 オプションで、memcache セッションハンドラのサポートを無効にすることもできます。 その場合は --disable-memcache-session を指定します。

Windows ユーザがこれらの関数を使用するには、php.ini の中で php_memcache.dll を有効にします。 この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

php.ini の設定により動作が変化します。

Memcache 設定オプション

名前	デフォルト	変更の可否	変更履歴
memcache.allow_failover	"1"	PHP_INI_ALL	memcache 2.0.2 以降で使用可能です
memcache.max_failover_attempts	"20"	PHP_INI_ALL	memcache 2.1.0 以降で使用可能です
memcache.chunk_size	"8192"	PHP_INI_ALL	memcache 2.0.2 以降で使用可能です
memcache.default_port	"11211"	PHP_INI_ALL	memcache 2.0.2 以降で使用可能です
memcache.hash_strategy	"standard"	PHP_INI_ALL	memcache 2.2.0 以降で使用可能です
memcache.hash_function	"crc32"	PHP_INI_ALL	memcache 2.2.0 以降で使用可能です
session.save_handler	"files"	PHP_INI_ALL	memcache 2.1.2 以降で使用可能です
session.save_path	""	PHP_INI_ALL	memcache 2.1.2 以降で使用可能です

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

memcache.allow_failover boolean

エラー時に、透過的なフェイルオーバーを行うかどうかを指定します。

memcache.max_failover_attempts integer

データの設定や取得を試みるサーバの数を指定します。 memcache.allow_failover を指定した場合にのみ使用します。

memcache.chunk_size integer

データは、ここで指定した大きさに分割されます。 この値を小さくすると、ネットワークに対する書き込みが多くなります。 不可解な速度低下が発生する場合は、この値を 32768 まで大きくしてください。

memcache.default_port string

memcached サーバに接続する際に、 デフォルトで使用される TCP ポート番号。

memcache.hash_strategy string

キーをサーバと関連づけるために使用する方式を制御します。この値を consistent にすると、一貫したハッシュを使用します。 これにより、サーバを追加したり削除したりした際にキーの再マッピングの必要がなくなります。 この値を standard にすると、以前の方法を使用します。

memcache.hash_function string

キーをサーバに関連づける際に使用するハッシュ関数を制御します。 crc32 は標準の CRC32 ハッシュを、そして fnv は FNV-1a を使用します。

session.save_handler string

memcache をセッションハンドラとして使用するには、この値を memcache と設定します。

session.save_path string

セッションを格納するためのサーバの URL を、カンマ区切りで指定します。たとえば "tcp://host1:11211, tcp://host2:11211" のようになります。

個々の URL には、そのサーバ用のパラメータを含めることができます。これは Memcache::addServer() メソッドと同じ形式です。たとえば "tcp://host1:11211?persistent=1&weight=1&timeout=1&retry_interval=15" のようになります。

リソース型

memcache モジュールで使用されるリソースは 1 種類で、それは キャッシュサーバとの接続を指す ID です。

定義済み定数

Memcache 定数

名前	説明
MEMCACHE_COMPRESSED (integer)	Memcache::set()、 Memcache::add() そして Memcache::replace() を実行する際に、同時にデータの圧縮を行います。
MEMCACHE_HAVE_SESSION (integer)	この Memcache セッションハンドラが有効な場合に 1、それ以外の場合に 0 となります。

例

Example#1 memcache 拡張モジュールの概要

この例では、オブジェクトをキャッシュに保存した後に、改めて取得しなおします。 オブジェクトやその他の非スカラー型のデータは、保存される前に シリアライズされます。そのため（接続 ID などの）リソース型を 保存することはできません。

```
<?php

$memcache = new Memcache;
$memcache->connect('localhost', 11211) or die ("接続できませんでした");

$version = $memcache->getVersion();
echo "サーバのバージョン: ".$version."<br/>¥n";

$tmp_object = new stdClass;
$tmp_object->str_attr = 'test';
$tmp_object->int_attr = 123;

$memcache->set('key', $tmp_object, false, 10) or die ("データをサーバに保存できませんでした");
echo "データをキャッシュに保存します (データの有効期限は 10 秒です)<br/>\n";

$get_result = $memcache->get('key');
echo "キャッシュから取得したデータ:<br/>\n";

var_dump($get_result);

?>
```

Example#2 memcache セッションハンドラの使用例

```
<?php

$session_save_path = "tcp://$host:$port?persistent=1&weight=2&timeout=2&retry_interval=10,    ,tcp://$host:$port    ";
ini_set('session.save_handler', 'memcache');
ini_set('session.save_path', $session_save_path);

?>
```

Memcache::add

(PECL memcache:0.2-2.1.2)

Memcache::add — サーバに項目を追加する

説明

bool **Memcache::add** (string $key , mixed $var [, int $flag [, int $expire]])

Memcache::add() は、サーバに同名のキーが存在しない 場合に限り、key というキーで 値 var を格納します。 **memcache_add()** 関数を使用することも可能です。

パラメータ

key

項目に関連付けられたキー。

var

格納する値。文字列および整数値はそのままの形式で、それ以外の型は シリアライズされて格納されます。

flag

項目を圧縮して格納する場合に **MEMCACHE_COMPRESSED** を使用します (zlib を使用します)。

expire

項目の有効期限。ゼロの場合は有効期限なし（いつまでも有効）となります。Unix タイムスタンプ形式、あるいは現在からの 秒数で指定することが可能ですが、後者の場合は秒数が 2592000 (30 日) を超えることはできません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 もし同名のキーが既に存在する場合は **FALSE** を返します。 それ以外は、**Memcache::add()** の振る舞いは **Memcache::set()** と同じです。

例

Example#1 Memcache::add() の例

```
<?php

$memcache_obj = memcache_connect("localhost", 11211);

/* 手続き型の API */
memcache_add($memcache_obj, 'var_key', 'test  variable', false, 30);

/* オブジェクト指向の API */
$memcache_obj->add('var_key', 'test  variable', false, 30);

?>
```

参考

- **Memcache::set()**
- **Memcache::replace()**

Memcache::addServer

(No version information available, might be only in CVS)

Memcache::addServer — コネクションプールに memcached サーバを追加する

説明

bool **Memcache::addServer** (string $host [, int $port [, bool $persistent [, int $weight [, int $timeout [, int $retry_interval [, bool $status [, callback $failure_callback]]]]]]])

Memcache::addServer() は、コネクションプールに サーバを追加します。 実際の接続は、最初に使用する際に確立されます。 **Memcache::addServer()** を使用してオープンされた 接続は、スクリプトの実行終了時に自動的に閉じられます。 **Memcache::close()** を使用して閉じることも可能です。 **memcache_add_server()** 関数を使用することも可能です。

(**Memcache::connect()**() および **Memcache::pconnect()**() ではなく) このメソッドを使用すると、ネットワーク接続は それが実際に必要となるときまで確率されません。 つまり、大量のサーバをプールに追加した場合に、 それらすべてが使用されることはないとしてもオーバーヘッドが発生しないということです。

他のサーバが使用可能である場合、あらゆるメソッドのあらゆる段階について ユーザが意識しないままにフェイルオーバー処理が行われます。 ソケットあるいは Memcaches サーバレベルで発生したあらゆるエラー (ただし out-of-memory は除く) に対してフェイルオーバーが動作します。 既存のキーを追加しようとしたなどの通常のクライアントエラーの場合は、 フェイルオーバー処理は起動しません。

> **注意**: この関数は、Memcache バージョン 2.0.0 で追加されました。

パラメータ

host

memcached が接続を待ち受けるホストを指定します。 このパラメータには別のトランスポート層を指定することもできます。たとえば unix:///path/to/memcached.sock のようにすると Unix ドメインソケットを使用できます。この場合、 port は 0 を指定しなければなりません。

port

memcached が接続を待ち受けるポートを指定します。 このパラメータはオプションで、デフォルト値は 11211 です。 Unix ドメインソケットを使用する場合は、このパラメータを 0 とします。

persistent

持続的な接続を使用するかどうかを指定します。 デフォルトは **TRUE** です。

weight

このサーバに対して割り当てる容器の数を指定します。これは、 このサーバが選択される可能性を左右します。選択される可能性は、 すべてのサーバの weight の合計に対するこのパラメータの割合で 決まります。

timeout

デーモンへの接続の際に使用する値 (秒単位) です。 デフォルト値を 1 秒でも変更する前には十分注意してください。 接続が遅くなってしまうと、キャッシュ処理のメリットが なくなってしまいます。

retry_interval

サーバとの接続が失敗した際に再試行を行う頻度を設定します。 デフォルト値は 15 秒です。このパラメータを -1 にすると、 自動的な再試行を行いません。 dl() を使用してこの拡張モジュールが動的に 読み込まれている場合は、このパラメータおよび persistent は何の効果も及ぼしません。

失敗した接続構造体は、個々にタイムアウト値を持っており、 タイムアウト時間が経過するまでは、バックエンドから新たな要求が来ても その構造体はスキップされます。時間が経過すると、 その接続が無事再接続されるか、あるいはさらに retry_interval 秒の間、接続失敗と記録されます。 典型的な効果は、ウェブサーバの各子プロセスがページを送り出す際に retry_interval 秒ごとに接続を再試行することです。

status

サーバがオンライン状態であるかどうかを制御します。このパラメータを **FALSE** にし、retry_interval を -1 と設定すると、失敗したサーバもコネクションプールに残します。 これにより、キー配布アルゴリズムに影響を与えないようにします。 このサーバへのリクエストは、フェイルオーバーされるか失敗します。 どちらになるかは memcache.allow_failover の設定によって決まります。デフォルトは **TRUE** で、 サーバがオンライン状態であることを意味します。

failure_callback

エラーが発生した際に実行されるコールバック関数を指定できるようにします。 コールバック関数は、フェイルオーバー処理の前に実行されます。 この関数は、ふたつの引数 (ホスト名、失敗したサーバのポート) を受け取ります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::addServer() の例

```php
<?php

/* オブジェクト指向の API */

$memcache = new Memcache;
$memcache->addServer('memcache_host', 11211);
$memcache->addServer('memcache_host2', 11211);

/* 手続き型の API */

$memcache_obj = memcache_connect('memcache_host', 11211);
```

```
memcache_add_server($memcache_obj, 'memcache_host2', 11211);

?>
```

参考

- **Memcache::connect()**
- **Memcache::pconnect()**
- **Memcache::close()**
- **Memcache::setServerParams()**
- **Memcache::getServerStatus()**

Memcache::close

(PECL memcache:0.4-2.1.2)

Memcache::close — memcached サーバとの接続を閉じる

説明

bool **Memcache::close** (void)

Memcache::close() は、memcached サーバとの接続を閉じます。 この関数は、持続的な接続については閉じません。持続的な接続が 閉じられるのは、Web サーバのシャットダウン/再起動のときだけです。 **memcache_close()** 関数を使用することも可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::close() の例

```
<?php

/* 手続き型の API */
$memcache_obj = memcache_connect('memcache_host', 11211);
/*
ここで何かを行います ..
*/
memcache_close($memcache_obj);

/* オブジェクト指向の API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
/*
ここで何かを行います ..
*/
$memcache_obj->close();

?>
```

参考

- **Memcache::connect()**
- **Memcache::pconnect()**

Memcache::connect

(PECL memcache:0.2-2.1.2)

Memcache::connect — memcached サーバへの接続をオープンする

説明

bool **Memcache::connect** (string $host [, int $port [, int $timeout]])

Memcache::connect() は、memcached サーバへの接続を 確立します。 **Memcache::connect()** を使用してオープンされた接続は、 スクリプトの実行終了時に自動的に閉じられます。 **Memcache::close()** を使用して閉じることも可能です。 **memcache_connect()** 関数を使用することも可能です。

パラメータ

host

memcached が接続を待ち受けるホストを指定します。 このパラメータには別のトランスポート層を指定することもできます。たとえば unix:///path/to/memcached.sock のようにすると Unix ドメインソケットを使用できます。この場合、 port は 0 を指定しなければなりません。

port

memcached が接続を待ち受けるポートを指定します。 Unix ドメインソケットを使用する場合は、このパラメータを 0 とします。

timeout

デーモンへの接続の際に使用する値（秒単位）です。 デフォルト値を 1 秒でも変更する前には十分注意してください。 接続が遅くなってしまうと、キャッシュ処理のメリットが なくなってしまいます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::connect() の例

```
<?php

/* 手続き型の API */

$memcache_obj = memcache_connect('memcache_host', 11211);

/* オブジェクト指向の API */

$memcache = new Memcache;
$memcache->connect('memcache_host', 11211);

?>
```

参考

- **Memcache::pconnect()**
- **Memcache::close()**

---

memcache_debug

(PECL memcache:0.2-2.1.2)

memcache_debug — デバッグ出力のオン/オフを切り替える

説明

bool **memcache_debug** (bool $on_off)

memcache_debug() は、パラメータ on_off が **TRUE** の場合にデバッグ出力を有効にし、 **FALSE** の場合には無効にします。

> **注意**: **memcache_debug()** は、PHP が --enable-debug オプションをつけてビルドされている場合にのみ使用可能で、 この場合は常に **TRUE** を返します。そうでない場合は、この関数は 何も行わずに常に **FALSE** を返します。

パラメータ

on_off

TRUE にすると、デバッグ出力をオンにします。 **FALSE** にすると、デバッグ出力をオフにします。

返り値

PHP が --enable-debug オプションをつけてビルドされている場合に **TRUE**、それ以外の場合に **FALSE** を返します。

Memcache::decrement

(PECL memcache:0.2-2.1.2)

Memcache::decrement — 項目の値を減らす

説明

int **Memcache::decrement** (string $key [, int $value])

Memcache::decrement() は、項目の値を value だけ減らします。 **Memcache::increment()** と同様、項目の現在の値が まず数値に変換されてから value を引きます。

> **注意**: 項目の新しい値をゼロより小さくすることはできません。
>
> **注意**: 圧縮して格納されている項目に対して **Memcache::decrement()** を使用しないでください。 なぜなら、それ以降の **Memcache::get()** のコールが 失敗してしまうからです。

Memcache::decrement() は、指定した項目が 存在しない場合に項目を作成することは ありません。 **memcache_decrement()** 関数を使用することも可能です。

パラメータ

key

値を減らす項目のキー。

value

項目の値を value だけ減らします。 オプションのパラメータで、デフォルトは 1。

返り値

項目の新しい値か、失敗した場合は **FALSE** を返します。

例

Example#1 Memcache::decrement() の例

```
<?php

/* 手続き型の API */
$memcache_obj = memcache_connect('memcache_host', 11211);
/* 項目の値を 2 減らします */
$new_value = memcache_decrement($memcache_obj, 'test_item', 2);

/* オブジェクト指向の API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
/* 項目の値を 3 減らします */
$new_value = $memcache_obj->decrement('test_item', 3);
?>
```

参考

- **Memcache::increment()**
- **Memcache::replace()**

Memcache::delete

(PECL memcache:0.2-2.1.2)

Memcache::delete — サーバから項目を削除する

説明

bool **Memcache::delete** (string $key [, int $timeout])

Memcache::delete() は、 key に対応する項目を削除します。 パラメータ timeout が指定されている場合は、 その項目は timeout 秒が経過した後に 期限切れとなります。 **memcache_delete()** 関数を使用することも可能です。

パラメータ

key

timeout

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::delete() の例

```
<?php

/* 手続き型の API */
$memcache_obj = memcache_connect('memcache_host', 11211);

/* 10 秒後に、項目がサーバから削除されます */
memcache_delete($memcache_obj, 'key_to_delete', 10);

/* オブジェクト指向の API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);

$memcache_obj->delete('key_to_delete', 10);

?>
```

参考

- **Memcache::set()**
- **Memcache::replace()**

Memcache::flush

(PECL memcache:1.0-2.1.2)

Memcache::flush — サーバ上のすべての既存項目を消去する

説明

bool **Memcache::flush** (void)

Memcache::flush() は、すべての既存項目を直ちに 無効にします。**Memcache::flush()** は 実際にリソースを開放するわけではなく、単にすべての項目に 有効期限切れのマークをつけるだけです。それらの項目が使用していた メモリは、新しい項目で上書きされるようになります。
memcache_flush() 関数を使用することも可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::flush() の例

```php
<?php

/* 手続き型の API */
$memcache_obj = memcache_connect('memcache_host', 11211);

memcache_flush($memcache_obj);

/* オブジェクト指向の API */

$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);

$memcache_obj->flush();

?>
```

Memcache::get

(PECL memcache:0.2-2.1.2)

Memcache::get — サーバから項目を取得する

説明

```
string Memcache::get ( string $key [, int &$flags ] )
array Memcache::get ( array $keys [, array &$flags ] )
```

Memcache::get() は、その時点でサーバ上に key というキーのデータが存在する場合に、 格納されているデータを返します。

Memcache::get() にキーの配列を渡すことにより、 値の配列を取得することができます。この配列には、サーバ上で見つかった キーと値のペアのみが含まれます。

パラメータ

key

取得したいキー（あるいはキーの配列）。

flags

存在した場合は、値とともに取得したフラグをここに書き込みます。 これらのフラグは、たとえば **Memcache::set()** に渡すものと同じです。int の最下位バイトは pecl/memcache で内部的に使用するために予約されています（たとえば圧縮やシリアライズに関する状態を表します）。

返り値

key に関連付けられた文字列を返します。 取得に失敗したり key が見つからなかったりした 場合には **FALSE** を返します。

例

Example#1 Memcache::get() の例

```php
<?php

/* 手続き型の API */
$memcache_obj = memcache_connect('memcache_host', 11211);
$var = memcache_get($memcache_obj, 'some_key');

/* オブジェクト指向の API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
$var = $memcache_obj->get('some_key');

/*
キーの配列をパラメータとして使用することもできます。
もしキーに対応する項目がサーバ上で見つからなければ、
結果の配列の中にはそのキーは含まれません。
*/

/* 手続き型の API */
$memcache_obj = memcache_connect('memcache_host', 11211);
$var = memcache_get($memcache_obj, Array('some_key', 'another_key'));

/* オブジェクト指向の API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
$var = $memcache_obj->get(Array('some_key', 'second_key'));

?>
```

Memcache::getExtendedStats

(No version information available, might be only in CVS)

Memcache::getExtendedStats — プール内のすべてのサーバの統計情報を取得する

説明

array **Memcache::getExtendedStats** ([string $type [, int $slabid [, int $limit]]])

Memcache::getExtendedStats() は、サーバの 統計情報を含む二次元の配列を返します。配列のキーが各サーバの host:port に対応し、その値として個々のサーバの統計情報を保持します。 取得に失敗したサーバは、値に **FALSE** が設定されます。 **memcache_get_extended_stats()** 関数を使用することも可能です。

> **注意**: この関数は、Memcache バージョン 2.0.0 で追加されました。

パラメータ

type

取得する統計情報の型。使用可能な値は {reset, malloc, maps, cachedump, slabs, items, sizes} のいずれかです。 memcached プロトコルの仕様によると、これらの追加の引数は 「memcache の開発者の都合により、変更される可能性があります」 ということです。

slabid

type を cachedump と設定した場合に使用し、ダンプを取得する slab を指定します。 cachedump コマンドはサーバと結びついており、デバッグ目的でのみ使用します。

limit

type を cachedump と設定した場合に使用し、ダンプするエントリの数を制限します。 デフォルト値は 100 です。

返り値

サーバの統計情報を含む二次元の配列を返します。失敗した場合は **FALSE** を返します。

例

Example#1 Memcache::getExtendedStats() の例

```
<?php
     $memcache_obj = new Memcache;
    $memcache_obj->addServer('memcache_host', 11211);
    $memcache_obj->addServer('failed_host', 11211);

    $stats = $memcache_obj->getExtendedStats();
    print_r($stats);
?>
```

上の例の出力は以下となります。

```
Array
(
      [memcache_host:11211] => Array
         (
               [pid] => 3756
               [uptime] => 603011
               [time] => 1133810435
               [version] => 1.1.12
               [rusage_user] => 0.451931
               [rusage_system] => 0.634903
               [curr_items] => 2483
               [total_items] => 3079
               [bytes] => 2718136
               [curr_connections] => 2
               [total_connections] => 807
               [connection_structures] => 13
               [cmd_get] => 9748
               [cmd_set] => 3096
               [get_hits] => 5976
               [get_misses] => 3772
               [bytes_read] => 3448968
               [bytes_written] => 2318883
               [limit_maxbytes] => 33554432
         )

      [failed_host:11211] => false
)
```

参考

- **Memcache::getVersion()**
- **Memcache::getStats()**

Memcache::getServerStatus

(No version information available, might be only in CVS)

Memcache::getServerStatus — サーバの状態を返す

説明

int **Memcache::getServerStatus** (string $host [, int $port])

Memcache::getServerStatus() は、サーバがオンライン/オフラインのどちらであるかを返します。 **memcache_get_server_status()** 関数を使用することも可能です。

> **注意**: この関数は、Memcache バージョン 2.1.0 で追加されました。

パラメータ

host

memcached が接続を待ち受けるホストを指定します。

port

memcached が接続を待ち受けるポートを指定します。 このパラメータはオプションで、デフォルト値は 11211 です。

返り値

サーバの状態を返します。サーバに接続できなかった場合に 0、 それ以外の場合にはゼロ以外の値を返します。

例

Example#1 Memcache::getServerStatus() の例

```
<?php

/* オブジェクト指向の API */
$memcache = new Memcache;
$memcache->addServer('memcache_host', 11211);
echo $memcache->getServerStatus('memcache_host', 11211);

/* 手続き型の API */
$memcache = memcache_connect('memcache_host', 11211);
echo memcache_get_server_status($memcache, 'memcache_host', 11211);

?>
```

参考

- **Memcache::addServer()**
- **Memcache::setServerParams()**

Memcache::getStats

(No version information available, might be only in CVS)

Memcache::getStats — サーバの統計情報を取得する

説明

array **Memcache::getStats** ([string $type [, int $slabid [, int $limit]]])

Memcache::getStats() は、サーバの統計情報を含む 連想配列を返します。配列のキーが統計情報パラメータの名前、そして 配列の値がパラメータの値に対応します。 **memcache_get_stats()** 関数を使用することも可能です。

パラメータ

type

取得する統計情報の型。使用可能な値は {reset, malloc, maps, cachedump, slabs, items, sizes} のいずれかです。 memcached プロトコルの仕様によると、これらの追加の引数は 「memcache の開発者の都合により、変更される可能性があります」 ということです。

slabid

type を cachedump と設定した場合に使用し、ダンプを取得する slab を指定します。 cachedump コマンドはサーバと結びついており、デバッグ目的でのみ使用します。

limit

type を cachedump と設定した場合に使用し、ダンプするエントリの数を制限します。 デフォルト値は 100 です。

返り値

サーバの統計情報を含む連想配列を返します。失敗した場合は **FALSE** を返します。

参考

- **Memcache::getVersion()**
- **Memcache::getExtendedStats()**

Memcache::getVersion

(No version information available, might be only in CVS)

Memcache::getVersion — サーバのバージョンを返す

説明

string **Memcache::getVersion** (void)

Memcache::getVersion() は、サーバのバージョン番号を 文字列で返します。 **memcache_get_version()** 関数を使用することも可能です。

返り値

バージョン番号を表す文字列を返します。 失敗した場合は **FALSE** を返します。

例

Example#1 Memcache::getVersion() の例

```
<?php

/* オブジェクト指向の API */
$memcache = new Memcache;
$memcache->connect('memcache_host', 11211);
echo $memcache->getVersion();

/* 手続き型の API */
$memcache = memcache_connect('memcache_host', 11211);
echo memcache_get_version($memcache);

?>
```

参考

- **Memcache::getExtendedStats()**
- **Memcache::getStats()**

Memcache::increment

(PECL memcache:0.2-2.1.2)

Memcache::increment — 項目の値を増やす

説明

int **Memcache::increment** (string $key [, int $value])

Memcache::increment() は、項目の値を value だけ増やします。 もし key に対応する値が数値ではなく、かつ 数値に変換できなかった場合は、その新しい値は value となります。 **Memcache::increment()** は、指定した項目が 存在しない場合に項目を作成することは ありません。

> **注意**: 圧縮して格納されている項目に対して **Memcache::increment()** を使用しないでください。 なぜなら、それ以降の **Memcache::get()** のコールが 失敗してしまうからです。

memcache_increment() 関数を使用することも可能です。

パラメータ

key

値を増やす項目のキー。

value

項目の値を value だけ増やします。 オプションのパラメータで、デフォルト値は 1 です。

返り値

項目の新しい値か、失敗した場合には **FALSE** を返します。

例

Example#1 Memcache::increment() の例

```
<?php

/* 手続き型の API */
$memcache_obj = memcache_connect('memcache_host', 11211);
/* カウンタを 2 増やします */
$current_value = memcache_increment($memcache_obj, 'counter', 2);

/* オブジェクト指向の API */
$memcache_obj = new Memcache;
$memcache_obj->connect('memcache_host', 11211);
/* カウンタを 3 増やします */
$current_value = $memcache_obj->increment('counter', 3);

?>
```

参考

- Memcache::decrement()
- Memcache::replace()

Memcache::pconnect

(PECL memcache:0.4-2.1.2)

Memcache::pconnect — memcached サーバへの持続的な接続をオープンする

説明

bool Memcache::pconnect (string $host [, int $port [, int $timeout]])

Memcache::pconnect() は Memcache::connect() とほぼ同じですが、 確立する接続が持続的なものであるという点が違います。 この接続はスクリプトの実行が終了しても閉じられることはなく、 Memcache::close() 関数を使用することで閉じられます。 memcache_pconnect() 関数を使用することも可能です。

パラメータ

host

memcached が接続を待ち受けるホストを指定します。 このパラメータには別のトランスポート層を指定することもできます。たとえば unix:///path/to/memcached.sock のようにすると Unix ドメインソケットを使用できます。この場合、 port は 0 を指定しなければなりません。

port

memcached が接続を待ち受けるポートを指定します。 Unix ドメインソケットを使用する場合は、このパラメータを 0 とします。

timeout

デーモンへの接続の際に使用する値（秒単位）です。 デフォルト値を 1 秒でも変更する前には十分注意してください。 接続が遅くなってしまうと、キャッシュ処理のメリットが なくなってしまいます。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 Memcache::pconnect() の例

```
<?php

/* 手続き型の API */
$memcache_obj = memcache_pconnect('memcache_host', 11211);

/* オブジェクト指向の API */

$memcache_obj = new Memcache;
$memcache_obj->pconnect('memcache_host', 11211);

?>
```

参考

- Memcache::connect()

Memcache::replace

(PECL memcache:0.2-2.1.2)

Memcache::replace — 既存項目の値を置換する

説明

bool Memcache::replace (string $key , mixed $var [, int $flag [, int $expire]])

Memcache::replace() は、key に対応する既存項目の値を置換するために使用します。 指定したキーに対応する項目がない場合は、Memcache::replace() は FALSE を返します。 それ以外の点では、Memcache::replace() の振る舞いは Memcache::set() と同じです。memcache_replace() 関数を使用することも可能です。

パラメータ

key

項目に関連付けられたキー。

var

格納する値。文字列および整数値はそのままの形式で、それ以外の型は シリアライズされて格納されます。

flag

項目を圧縮して格納する場合に **MEMCACHE_COMPRESSED** を使用します (zlib を使用します)。

expire

項目の有効期限。ゼロの場合は有効期限なし (いつまでも有効) となります。Unix タイムスタンプ形式、あるいは現在からの 秒数で指定することが可能ですが、後者の場合は秒数が 2592000 (30 日) を超えることはできません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::replace() の例

```
<?php

$memcache_obj = memcache_connect('memcache_host', 11211);

/* 手続き型の API */
memcache_replace($memcache_obj, "test_key", "some variable", false, 30);

/* オブジェクト指向の API */
$memcache_obj->replace("test_key", "some variable", false, 30);

?>
```

参考

- **Memcache::set()**
- **Memcache::add()**

Memcache::set

(PECL memcache:0.2-2.1.2)

Memcache::set — データをサーバに格納する

説明

bool **Memcache::set** (string $key , mixed $var [, int $flag [, int $expire]])

Memcache::set() は、キー key に var という値を 関連付け、memcached サーバに格納します。パラメータ expire は、データの有効期限を秒単位で指定します。もし 0 を指定した場合は その項目が期限切れになることはありません (これは、その項目のデータが memcached サーバ上にずっと残り続けることを保証するものではありません。 他の項目をキャッシュするための場所を確保するためにサーバから 削除されてしまうこともあります)。 (zlib を使用して) その場でのデータの圧縮を行いたい場合は、 flag の値として、定数 **MEMCACHE_COMPRESSED** を指定します。

> **注意**: リソース型の変数 (たとえばファイル記述子や接続記述子など) はキャッシュに保存できないことを覚えておきましょう。これは、 シリアライズした状態ではそれらのデータを適切に表すことが できないためです。

memcache_set() 関数を使用することも可能です。

パラメータ

key

項目に関連付けられたキー。

var

格納する値。文字列および整数値はそのままの形式で、それ以外の型は シリアライズされて格納されます。

flag

項目を圧縮して格納する場合に **MEMCACHE_COMPRESSED** を使用します (zlib を使用します)。

expire

項目の有効期限。ゼロの場合は有効期限なし (いつまでも有効) となります。Unix タイムスタンプ形式、あるいは現在からの 秒数で指定することが可能ですが、後者の場合は秒数が 2592000 (30 日) を超えることはできません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::set() の例

```
<?php
/* 手続き型の API */

/* memcached サーバに接続します */
$memcache_obj = memcache_connect('memcache_host', 11211);

/*
キー 'var_key' の項目の値を設定します。
flag の値として 0 を使用し、圧縮は使用しません。
有効期限は 30 秒です。
*/
```

```
memcache_set($memcache_obj, 'var_key', 'some  variable', 0, 30);

echo memcache_get($memcache_obj, 'var_key');

?>
```

Example#2 Memcache::set() の例

```
<?php
/* オブジェクト指向の API */

$memcache_obj  = new Memcache;

/* memcached サーバに接続します */
$memcache_obj->connect('memcache_host', 11211);

/*
キー 'var_key' に対応する値を設定します。その際、データの圧縮を行います。
有効期限は 50 秒です。
*/
$memcache_obj->set('var_key', 'some  really  big  variable', MEMCACHE_COMPRESSED, 50);

echo $memcache_obj->get('var_key');

?>
```

参考

- **Memcache::add()**
- **Memcache::replace()**

Memcache::setCompressThreshold

(No version information available, might be only in CVS)

Memcache::setCompressThreshold — 大きな値に対する自動圧縮処理を有効にする

説明

bool **Memcache::setCompressThreshold** (int $threshold [, float $min_savings])

Memcache::setCompressThreshold() は、 大きな値に対して自動圧縮処理を有効にします。 **memcache_set_compress_threshold()** 関数を使用することも可能です。

> **注意**: この関数は、Memcache バージョン 2.0.0 で追加されました。

パラメータ

threshold

自動圧縮を試みる値の長さの最小値を指定します。

min_saving

値を圧縮する際の、最小の圧縮率を指定します。 値は 0 から 1 までの間で指定する必要があります。 デフォルト値は 0.2 で、最低 20% の圧縮を指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::setCompressThreshold() の例

```
<?php

/* オブジェクト指向の API */

$memcache_obj  = new Memcache;
$memcache_obj->addServer('memcache_host', 11211);
$memcache_obj->setCompressThreshold(20000, 0.2);

/* 手続き型の API */

$memcache_obj  = memcache_connect('memcache_host', 11211);
memcache_set_compress_threshold($memcache_obj, 20000, 0.2);

?>
```

Memcache::setServerParams

(No version information available, might be only in CVS)

Memcache::setServerParams — サーバのパラメータおよび状態を、実行時に変更する

説明

bool **Memcache::setServerParams** (string $host [, int $port [, int $timeout [, int $retry_interval [, bool $status [, callback $failure_callback]]]]])

Memcache::setServerParams() は、 サーバのパラメータを実行時に変更します。 **memcache_set_server_params()** 関数を使用することも可能です。

注意: この関数は、Memcache バージョン 2.1.0 で追加されました。

パラメータ

host

memcached が接続を待ち受けるホストを指定します。

port

memcached が接続を待ち受けるポートを指定します。 このパラメータはオプションで、デフォルト値は 11211 です。

timeout

デーモンへの接続の際に使用する値（秒単位）です。 デフォルト値を 1 秒でも変更する前には十分注意してください。 接続が遅くなってしまうと、キャッシュ処理のメリットが なくなってしまいます。

retry_interval

サーバとの接続が失敗した際に再試行を行う頻度を設定します。 デフォルト値は 15 秒です。このパラメータを -1 にすると、 自動的な再試行を行いません。 dl() を使用してこの拡張モジュールが動的に 読み込まれている場合は、このパラメータおよび persistent は何の効果も及ぼしません。

status

サーバがオンライン状態であるかどうかを制御します。このパラメータを **FALSE** にし、retry_interval を -1 と設定すると、失敗したサーバもコネクションプールに残します。 これにより、キー配布アルゴリズムに影響を与えないようにします。 このサーバへのリクエストは、フェイルオーバーされるか失敗します。 どちらになるかは memcache.allow_failover の設定によって決まります。デフォルトは **TRUE** で、 サーバがオンライン状態であることを意味します。

failure_callback

エラーが発生した際に実行されるコールバック関数を指定できるようにします。 コールバック関数は、フェイルオーバー処理の前に実行されます。 この関数は、ふたつの引数（ホスト名、失敗したサーバのポート）を受け取ります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Memcache::setServerParams() の例

```php
<?php

function _callback_memcache_failure($host, $port) {
    print "memcache '$host:$port' failed";
}

/* オブジェクト指向の API */

$memcache = new Memcache;

// サーバをオフラインモードで追加します
$memcache->addServer('memcache_host', 11211, false, 1, 1, -1, false);

// サーバをオンラインに変更します
$memcache->setServerParams('memcache_host', 11211, 1, 15, true, '_callback_memcache_failure');

/* 手続き型の API */

$memcache_obj = memcache_connect('memcache_host', 11211);
memcache_set_server_params($memcache_obj, 'memcache_host', 11211, 1, 15, true, '_callback_memcache_failure');

?>
```

参考

- **Memcache::addServer()**
- **Memcache::getServerStatus()**

目次

- Memcache::getExtendedStats — プール内のすべてのサーバの統計情報を取得する
- Memcache::getServerStatus — サーバの状態を返す
- Memcache::getStats — サーバの統計情報を取得する
- Memcache::getVersion — サーバのバージョンを返す
- Memcache::increment — 項目の値を増やす
- Memcache::pconnect — memcached サーバへの持続的な接続をオープンする
- Memcache::replace — 既存項目の値を置換する
- Memcache::set — データをサーバに格納する
- Memcache::setCompressThreshold — 大きな値に対する自動圧縮処理を有効にする
- Memcache::setServerParams — サーバのパラメータおよび状態を、実行時に変更する

Mhash 関数

導入

以下の関数は、» mhash と組み合わせて 動作することを前提としています。mhashは、チェックサム、メッセージ ダイジェスト、メッセージ認証コード等を作成するために使用することが できます。

この関数は、mhash ライブラリへのインターフェースです。 mhash は、MD5, SHA1, GOST や他の多くの方法といった広範なハッシュ アルゴリズムをサポートします。サポートされるハッシュの全一覧に ついては、mhash のドキュメントを参照してください。一般的な規則として、 特定のハッシュアルゴリズムは、PHP から定数「MHASH_ハッシュ名」で アクセス可能です。例えば、TIGER の場合、PHP 定数 MHASH_TIGER を 使用します。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 5.3.0. この拡張モジュールの後継版は Hash です。

要件

mhash を使用するには、mhash の配布ファイルを » mhash の Web サイト から ダウンロードし、その中のインストール用の指示に従ってください。

インストール手順

この拡張機能を使用するには、PHP に --with-mhash[=DIR] パラメータを付けて コンパイルする必要があります。DIR は mhash インストールディレクトリです。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

以下に現在 mhash によりサポートされているハッシュの一覧を示します。 mhash にサポートされているハッシュがこのリストにない場合は、 このドキュメントが古いと考えてください。

- **MHASH_ADLER32**
- **MHASH_CRC32**
- **MHASH_CRC32B**
- **MHASH_GOST**
- **MHASH_HAVAL128**
- **MHASH_HAVAL160**
- **MHASH_HAVAL192**
- **MHASH_HAVAL256**
- **MHASH_MD4**
- **MHASH_MD5**
- **MHASH_RIPEMD160**
- **MHASH_SHA1**
- **MHASH_SHA256**
- **MHASH_TIGER**
- **MHASH_TIGER128**
- **MHASH_TIGER160**

例

Example#1 MD5 ダイジェストと hmac を計算し、16 進数で出力する

```
<?php
$input = "what do ya want for nothing?";
```

```
$hash = mhash(MHASH_MD5, $input);
echo "The hash is " . bin2hex($hash) . "<br />¥n";
$hash = mhash(MHASH_MD5, $input, "Jefe");
echo "The hmac is " . bin2hex($hash) . "<br />¥n";
?>
```

この例の出力は次のようになります。

```
The hash is d03cb659cbf9192dcd066272249f8412
The hmac is 750c783e6ab0b503eaa86e310a5db738
```

---

mhash_count

(PHP 4, PHP 5)

mhash_count — 利用可能なハッシュ ID の最大値を得る

説明

int **mhash_count** (void)

利用可能なハッシュ ID の最大値を取得します。

返り値

利用可能なハッシュ ID の最大値を返します。 ハッシュは、0 からこのハッシュ ID までの数字となります。

例

Example#1 全ハッシュをループする

```
<?php

$nr = mhash_count();

for ($i = 0; $i <= $nr; $i++) {
    echo sprintf("The blocksize of %s is %d¥n",
        mhash_get_hash_name($i),
        mhash_get_block_size($i));
}
?>
```

---

mhash_get_block_size

(PHP 4, PHP 5)

mhash_get_block_size — 指定したハッシュのブロックサイズを得る

説明

int **mhash_get_block_size** (int $hash)

指定した hash のブロックサイズを取得します。

パラメータ

hash

ハッシュ ID。MHASH_XXX 定数のいずれかを指定します。

返り値

サイズをバイト数で返します。hash が存在しない場合は **FALSE** を返します。

例

Example#1 mhash_get_block_size() の例

```
<?php

echo mhash_get_block_size(MHASH_MD5); // 16

?>
```

---

mhash_get_hash_name

(PHP 4, PHP 5)

mhash_get_hash_name — 指定したハッシュの名前を得る

説明

string ***mhash_get_hash_name*** *(int $hash)*

指定した hash の名前を取得します。

パラメータ

hash

ハッシュ ID。MHASH_XXX 定数のいずれかを指定します。

返り値

ハッシュの名前を返します。そのハッシュが存在しない場合は、 ***FALSE*** *を返します。*

例

Example#1 mhash_get_hash_name() の例

```
<?php

echo mhash_get_hash_name(MHASH_MD5); // MD5

?>
```

---

mhash_keygen_s2k

(PHP 4 >= 4.0.4, PHP 5)

mhash_keygen_s2k — キーを生成する

説明

string ***mhash_keygen_s2k*** *(int $hash , string $password , string $salt , int $bytes)*

hash にユーザが指定した password を用いてキーを生成します。

OpenPGP のドキュメント (» RFC 2440) で規定されている、Salted S2k アルゴリズムを使用します。

ユーザが指定したパスワードは、暗号化アルゴリズムのキーとしては 適切ではないということを覚えておきましょう。ユーザが選択するのは、 通常はキーボードから入力可能なキーだからです。 これらのパスワードは、文字当たり 6 から 7 ビットのみ(もしくはそれ以下) しか使用していません。ユーザの指定したキーに対して、(この関数のような) ある種の変換を行うことを強く推奨します。

パラメータ

hash

キーの作成に使用するハッシュ ID。MHASH_XXX 定数のいずれかを指定します。

password

ユーザが指定したパスワード。

salt

異なったキーを生成するため、キーを生成するたびに、 十分にランダムな異なる値となる必要があります。 キーを調べる際に salt を知っている必要があるため、キーを salt に付加するというのは 良い発想です。salt は 8 バイト固定長で、これより少ない場合はゼロで 埋められます。

bytes

キーの長さをバイト数で指定します。

返り値

生成されたキーを表す文字列、あるいはエラー時に ***FALSE*** *を返します。*

---

mhash

(PHP 4, PHP 5)

mhash — ハッシュ値を計算する

説明

string ***mhash*** *(int $hash , string $data [, string $key])*

mhash() *は、 hash で指定したハッシュ関数を data に適用します。*

パラメータ

hash

ハッシュ ID。MHASH_XXX 定数のいずれかを指定します。

data

ユーザが入力した文字列。

key

指定した場合は、この関数は結果の HMAC を返します。 HMAC は、メッセージ認証時のキーに基づくハッシュ、 あるいは指定したキーに基づく単なるメッセージダイジェストです。 mhash でサポートしている全てのアルゴリズムが HMAC モードで使用できるわけではありません。

返り値

*結果のハッシュ（あるいはダイジェストとも言います）あるいは HMAC を表す文字列を返します。 エラー時は **FALSE** を返します。*

目次

Mimetype 関数

導入

警告

この拡張モジュールは廃止予定です。PECL の Fileinfo 拡張モジュールが、同等の機能（に加えてそれ以上の機能）を よりきれいな方法で提供しています。

このモジュールの関数は、ファイル内の特定の位置にある magic バイトシーケンスを探すことにより ファイルの content type とエンコーディングの推定を試みます。 この手法は失敗する可能性がないわけではありませんが、経験上 非常に良い結果が得られます。

この拡張モジュールは、Apache の mod_mime_magic から派生したものです。 mod_mime_magic は、Ian F. Darwin により管理されている file コマンドに基づいています。 より詳細な履歴や著作権についてはソースコードを参照してください。

要件

外部ライブラリを必要としません。

インストール手順

mime-type 関数をサポートするためには、configure スイッチ --with-mime-magic を付けて PHP をコンパイルする必要があります。この拡張モジュールは、 Apache httpd と共に配布されているシンプル版の magic ファイルのコピーを必要とします。

> ***注意**: configure オプションは、PHP 4.3.2 以降 --enable-mime-magic から --with-mime-magic に変更されています。*
>
> ***注意**: この拡張モジュールには、付加機能が多くついた magic ファイルを処理する機能はありません（この magic ファイルは、標準的なLinuxディストリビューションに付属しており、 最近のバージョンの file コマンドで使用されている ようです）。*
>
> ***注意**: **Win32 ユーザへの注意** このモジュールを Windows 環境で使用するには、バンドルされた magic.mime へのパスを php.ini に設定する必要 があります。*
>
> ***Example#1 magic.mime のパスの設定例***
>
> ```
> mime_magic.magicfile = "$PHP_INSTALL_DIR¥magic.mime"
> ```
>
> *上の例の $PHP_INSTALL_DIR を実際の PHP のパス、例えば、c:¥php に置き換えてください。*

実行時設定

php.ini の設定により動作が変化します。

Mimetype 設定オプション

名前	デフォルト	変更の可否	変更履歴
mime_magic.debug	"0"	PHP_INI_SYSTEM	PHP 5.0.0 以降で利用可能
mime_magic.magicfile	"/path/to/php/magic.mime"	PHP_INI_SYSTEM	PHP 4.3.0 以降で利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

mime_magic.debug bool

デバッグを有効/無効にする

mime_magic.magicfile string

magic.mime ファイルへのパス

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

mime_content_type

(PHP 4 >= 4.3.0, PHP 5)

mime_content_type — ファイルの MIME Content-type を検出する (非推奨)

説明

string **mime_content_type** (string $filename)

magic.mime ファイルの情報を用いて、 ファイルの MIME content type を返します。

パラメータ

filename

調べるファイルへのパス。

返り値

Content type を MIME 形式で返します。 たとえば text/plain あるいは application/octet-stream のような形式です。

注意

警告

この関数は非推奨です。PECL 拡張モジュール Fileinfo が、同等の機能 (それ以上のもの) をもっときれいな方法で提供しています。

例

Example#1 mime_content_type() の例

```
<?php
echo mime_content_type('php.gif') . "¥n";
echo mime_content_type('test.php');
?>
```

上の例の出力は以下となります。

```
image/gif
text/plain
```

参考

- Fileinfo を代わりに使用してください。

Flash 用 Ming 関数

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

導入

Ming は省略語ではありません。Ming は、 SWF ("Flash") フォーマットのムービーを作成するための オープンソース (LGPL) のライブラリです。Ming は、 シェープ、グラディエント、ビットマップ (PNG および JPEG)、モーフィング ("変形")、 テキスト、ボタン、アクション、スプライト ("ムービークリップ")、 mp3 のストリーム出力、色変換といった Flash 4 の機能のほとんど全てをサポートしています。 現在未サポートなのは、サウンド関係のイベントのみです。

長さ、距離、大きさ等を指定する値は全て "twips" つまり、20 ユニット /ピクセル 単位であることに注意してください。しかし、実際には、Flash プレイヤーがムービーを embed/object タグで指定したピクセルサイズまたは embed されていない場合はフレーム全体にスケーリングするため、 任意のサイズになります。

Ming は、既存の PHP/libswf モジュール に対して多くの点で優れています。 Ming は、そのコードをコンパイルできる環境でならどこでも使用することが可能です。一方、 libswf のソースコードは公開されておらず、 ごくわずかなプラットフォームでのみしか利用できません。 Windows は、libswf でサポートされるプラットフォームには入っていません。 Ming は、ムービーの要素を PHP オブジェクト内に隠蔽することにより、 SWF ファイルフォーマットの細部を直接取り扱うことを回避しています。 また、Ming はメンテナンスが継続されています。 使用したい機能がある場合には、我々、 » http://ming.sourceforge.net/ まで知らせてください。

Ming は、PHP 4.0.5 で追加されました。

要件

Ming を PHP で使用するには、まず、Ming ライブラリを構築し、 インストールする必要があります。ソースコードとインストール手引が、 Ming のホームページ » http://ming.sourceforge.net/ から入手可能です。ここには、例や簡単なチュートリアル、最新のニュースもあります。

ming のアーカイブをダウンロードし、展開してください。Ming ディレクトリに移動し、make、make install を実行してください。

これにより libming.so が構築され、 /usr/lib/ にインストールされます。また、 ming.h が /usr/include/ にコピーされます。インストールされるディレクトリを変更するには、 Makefile の PREFIX= の行を編集してください。

インストール手順

Example#1 PHP に組み込む(Unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure --with-ming <other config options>
```

PHP を通常と同様に構築、インストールしてください。必要ならば Web サーバを再起動してください。

ここで、extension=php_ming.so を php.ini ファイルに追加するか、 dl('php_ming.so'); を全ての Ming スクリプトの 先頭に追加してください。

実行時設定

設定ディレクティブは定義されていません。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

MING_NEW (integer)
MING_ZLIB (integer)
SWFBUTTON_HIT (integer)
SWFBUTTON_DOWN (integer)
SWFBUTTON_OVER (integer)
SWFBUTTON_UP (integer)
SWFBUTTON_MOUSEUPOUTSIDE (integer)
SWFBUTTON_DRAGOVER (integer)
SWFBUTTON_DRAGOUT (integer)
SWFBUTTON_MOUSEUP (integer)
SWFBUTTON_MOUSEDOWN (integer)
SWFBUTTON_MOUSEOUT (integer)
SWFBUTTON_MOUSEOVER (integer)
SWFFILL_RADIAL_GRADIENT (integer)
SWFFILL_LINEAR_GRADIENT (integer)
SWFFILL_TILED_BITMAP (integer)
SWFFILL_CLIPPED_BITMAP (integer)
SWFTEXTFIELD_HASLENGTH (integer)
SWFTEXTFIELD_NOEDIT (integer)
SWFTEXTFIELD_PASSWORD (integer)
SWFTEXTFIELD_MULTILINE (integer)
SWFTEXTFIELD_WORDWRAP (integer)
SWFTEXTFIELD_DRAWBOX (integer)
SWFTEXTFIELD_NOSELECT (integer)
SWFTEXTFIELD_HTML (integer)
SWFTEXTFIELD_ALIGN_LEFT (integer)
SWFTEXTFIELD_ALIGN_RIGHT (integer)
SWFTEXTFIELD_ALIGN_CENTER (integer)
SWFTEXTFIELD_ALIGN_JUSTIFY (integer)
SWFACTION_ONLOAD (integer)
SWFACTION_ENTERFRAME (integer)
SWFACTION_UNLOAD (integer)
SWFACTION_MOUSEMOVE (integer)
SWFACTION_MOUSEDOWN (integer)
SWFACTION_MOUSEUP (integer)
SWFACTION_KEYDOWN (integer)
SWFACTION_KEYUP (integer)
SWFACTION_DATA (integer)

定義済みクラス

以下のクラスが定義されています。この拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

Classes

- SWFAction
- SWFBitmap
- SWFButton
- SWFDisplayItem
- SWFFill
- SWFFont
- SWFFontChar
- SWFGradient
- SWFMorph

SWFAction

(No version information available, might be only in CVS)

SWFAction — SWFAction クラス

説明

スクリプトの文法は C 言語をもとにしていますが、多くの機能が 省略されています - SWF バイトコードマシンは、より単純であることを 志向しています。例えば、相当手の込んだ細工をしなければ、関数のコールを 実装することもできません。なぜなら、バイトコードの jump 命令は ハードコードされたオフセット値を使用しているからです。 呼び出し元のアドレスをスタックに格納したりはしません - すべての関数は、戻ってくる場所を正確に知っている必要があるのです。

結局、どんな機能が残っているのでしょう? コンパイラが理解できるトークンは 以下のとおりです。

- *break*
- *for*
- *continue*
- *if*
- *else*
- *do*
- *while*

データ型は存在しません。SWF アクションマシンにおいては、すべての値は 文字列として扱われます。以下の関数が使用可能です。

time()
ムービーが開始してからの経過時間を、ミリ秒 (?) で返します。
random(seed)
0 から seed までの範囲の擬似乱数を返します。
length(expr)
指定した式の長さを返します。
int(number)
指定した数値を、一番近い整数に切り下げた値を返します。
concat(expr, expr)
指定した式を連結して返します。
ord(expr)
指定した文字の ASCII コードを返します。
chr(num)
指定した ASCII コードに対応する文字を返します。
substr(string, location, length)
指定した文字列 string の、 位置 location から始まる 長さ length の部分文字列を返します。

さらに、以下のコマンドも使用できるでしょう。

duplicateClip(clip, name, depth)
指定したムービークリップ (またの名をスプライト) clip を複製します。 新しいムービークリップの名前は name で、 深度は depth となります。
removeClip(expr)
指定したムービークリップを削除します。
trace(expr)
指定した式をトレースログに書き込みます。ブラウザのプラグインが これをきちんと扱ってくれるかは疑わしいものです。
startDrag(target, lock, [left, top, right, bottom])
*ムービークリップ target のドラッグを開始します。 引数 lock で、マウスをロックするかどうか (?) を指定します - 0 (**FALSE**) あるいは 1 (**TRUE**) を指定します。 オプションのパラメータでは、ドラッグする範囲のを指定します。*
stopDrag()
つかれきった心を落ち着かせます。そしてムービークリップの ドラッグも修了させます。
callFrame(expr)
指定したフレームを関数としてコールします。
getURL(url, target, [method])
指定した URL を読み込みます。引数 target は、 (たとえば "_top" や "_blank" のような) HTML ドキュメントの target に対応します。オプションの引数 method は、 サーバに変数を返したい場合に POST あるいは GET を指定します。
loadMovie(url, target)
指定した URL を読み込みます。引数 target は、 (おそらく) フレームの名前か あるいは特別な値 "_level0" (現在のムービーを置き換える)、"_level1" (現在のムービーの前面に新しいムービーを表示する) のうちのいずれかです。
nextFrame()
次のフレームに移動します。
prevFrame()
直前の (あるいは一つ前の) フレームに移動します。
play()
ムービーの再生を開始します。
stop()
ムービーの再生を停止します。
toggleQuality()
高品質/低品質を切り替えます。
stopSounds()
音声の再生を停止します。
gotoFrame(num)
フレーム番号 num に移動します。 フレーム番号は 0 からはじまります。

gotoFrame(name)
name という名前のフレームに移動します。 これは便利です。というのもまだフレームのラベルを追加していないからです。
setTarget(expr)
アクションのコンテキストといわれるものを設定します。 これが何をするものなのかは実際のところよくわかりません。

そしてもうひとつ変なものがあります。if 文や while ループの中で、 指定したフレーム番号が読み込まれているかどうかを調べるために、 式 frameLoaded(num) が使用可能です。ええ。そのようにいわれています。 しかし私は実際にこれをテストしたことがなく、実際に動作するのか 疑問に思っています。かわりに /:framesLoaded を使用するとよいでしょう。

ムービークリップ（さぁみなさんご一緒に - またの名をスプライト）はプロパティをひじしています。すべてのプロパティが読み込み可能で、 そのうちのいくつかには値を設定することも可能です。プロパティの 一覧は以下のとおりです。

- x
- y
- xScale
- yScale
- currentFrame - (読み込み専用)
- totalFrames - (読み込み専用)
- alpha - 透明度
- visible - 1=on, 0=off (?)
- width - (読み込み専用)
- height - (読み込み専用)
- rotation
- target - (読み込み専用) (???)
- framesLoaded - (読み込み専用)
- name
- dropTarget - (読み込み専用) (???)
- url - (読み込み専用) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

ということで、スプライトの x 位置を指定するには単に /box.x = 100; とすればよいわけです。なぜ box の前のスラッシュがあるのでしょう? これは、flash がムービー内のスプライトを Unix ファイルシステム風に 管理しているからです - つまり、この場合 box がトップレベルに存在することになります。box という名前のスプライトがその中に biff という名前の スプライトを保持している場合、その x 位置を指定するには /box/biff.x = 100; とします。すくなくとも私はそう思っています。 もし間違っていたら指摘してください。

クラスのメンバ

メソッド

- SWFAction->__construct()

例

この単純な例は、赤い四角形がウィンドウを横切るものです。

Example#1 swfaction() の例

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500, -500);
  $s->drawLineTo(500, -500);
  $s->drawLineTo(500, 500);
  $s->drawLineTo(-500, 500);
  $s->drawLineTo(-500, -500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for ($n=0; $n<5; ++$n) {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000, 4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

この単純な例は、画面上のマウスを追いかけます。

Example#2 swfaction() の例

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* マウスを追いかけるスプライト - 空ですが、マウスを追いかけるために
     その x 座標と y 座標を取得することが可能です */

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
     startDrag('/mouse', 1); /* '1' は、スプライトをマウスにロックします */
  "));

  /* これらは単なる四角形なので、アンチエイリアスを off にしたほうがよいでしょう */

  $m->add(new SWFAction("
     this.quality = 0;
  "));

  /* 変形ボックス */
  $r = new SWFMorph();
  $s = $r->getShape1();

  /* これは通常の図形から背景よりになることに注意しましょう。なぜかはわかりません */
  $s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
  $s->movePenTo(-40, -40);
  $s->drawLine(80, 0);
  $s->drawLine(0, 80);
  $s->drawLine(-80, 0);
  $s->drawLine(0, -80);

  $s = $r->getShape2();

  $s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
  $s->movePenTo(-1, -1);
  $s->drawLine(2, 0);
  $s->drawLine(0, 2);
  $s->drawLine(-2, 0);
  $s->drawLine(0, -2);

  /* 変形ボックスのためのスプライトコンテナ
     これは、単にボックスを変形させる際の時間軸にすぎません */

  $box = new SWFSprite();
  $box->add(new SWFAction("
     stop();
  "));
  $i = $box->add($r);

  for ($n=0; $n<=20; ++$n) {
    $i->setRatio($n/20);
    $box->nextFrame();
  }

  /* このスプライトにより、同じアクションコードを何度も実行させることが可能となります */

  $cell = new SWFSprite();
  $i = $cell->add($box);
  $i->setName('box');

  $cell->add(new SWFAction("

      setTarget('box');

     /* ...x は、親の x 座標を意味します。すなわち (..).x ということです */
      dx = (/mouse.x + random(6)-3 - ...x)/5;
      dy = (/mouse.y + random(6)-3 - ...y)/5;
      gotoFrame(int(dx*dx + dy*dy));

  "));

  $cell->nextFrame();
  $cell->add(new SWFAction("

     gotoFrame(0);
     play();

  "));

  $cell->nextFrame();

  /* 最後に、セルをムービーに追加します */

  for ($x=0; $x<12; ++$x) {
     for ($y=0; $y<8; ++$y) {
       $i = $m->add($cell);
       $i->moveTo(100*$x+50, 100*$y+50);
     }
  }

  $m->nextFrame();

  $m->add(new SWFAction("
```

```
    gotoFrame(1);
    play();

  "));

  header('Content-type:  application/x-shockwave-flash');
  $m->output();
?>
```

上と同じですが、きれいな色のボールを使用します...

Example#3 swfaction() の例

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // マウスを追いかけるスプライト
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient を指定した図形
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // 複数の色を使用するため、スプライトにする必要があります
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // 図形をここに入れ、各フレームで別の色を使用します
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1);  stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // 最後に、アクションコードにこれを含めます
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
```

```
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame();  play();"));
  $p->nextFrame();

  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if (num == 11)
  num = 1;

removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame();  play();"));

  header('Content-type:  application/x-shockwave-flash');
  $m->output();
?>
```

SWFAction->__construct()

(PHP 5)

SWFAction->__construct() — 新しい SWFAction を作成する

説明

SWFAction
SWFAction **__construct** (string $script)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい SWFAction を作成し、その内部の script をコンパイルします。

パラメータ

script

この SWFAction に関連付ける ActionScript。 詳細は SWFAction を参照ください。

SWFBitmap

(No version information available, might be only in CVS)

SWFBitmap — SWFBitmap クラス

説明

クラスのメンバ

メソッド

- SWFBitmap->__construct()
- SWFBitmap->getHeight()
- SWFBitmap->getWidth()

SWFBitmap->__construct()

(PHP 5)

SWFBitmap->__construct() — ビットマップオブジェクトを読み込む

説明

SWFBitmap
SWFBitmap **__construct** (mixed $file [, mixed $alphafile])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい SWFBitmap オブジェクトを、指定した file から作成します。

パラメータ

どちらのパラメータについても、 fopen() が返すファイルポインタか 画像データのバイナリ文字列のいずれかを指定できます。

file

注意: baseline (frame 0) jpeg にのみ対応しています。baseline を最適化した jpeg やプログレッシブ jpeg には対応していません!

png 画像を直接読み込むことはできません。png2dbl を使用して、png から dbl ("define bits lossless") ファイルを作成する必要があります。 この理由は、ming に png ライブラリの依存性を持ち込みたくないからです。 これは autoconf が解決すべき問題ですが、いまのところ解決されていません。

alphafile

JPEG 画像のアルファマスクとして使用する MSK ファイル。

例

Example#1 DBL ファイルのインポート

```
<?php
$s = new SWFShape();
$f = $s->addFill(new SWFBitmap(file_get_contents("image.dbl")));
$s->setRightFill($f);

$s->drawLine(32, 0);
$s->drawLine(0, 32);
$s->drawLine(-32, 0);
$s->drawLine(0, -32);

$m = new SWFMovie();
$m->setDimension(32, 32);
$m->add($s);

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

Example#2 アルファマスクの使用法

```
<?php

$s = new SWFShape();

// .msk ファイルは "gif2mask" ユーティリティで作成します
$f = $s->addFill(new SWFBitmap(file_get_contents("alphafill.jpg"), file_get_contents("alphafill.msk")));
$s->setRightFill($f);

$s->drawLine(640, 0);
$s->drawLine(0, 480);
$s->drawLine(-640, 0);
$s->drawLine(0, -480);

$c = new SWFShape();
$c->setRightFill($c->addFill(0x99, 0x99, 0x99));
$c->drawLine(40, 0);
$c->drawLine(0, 40);
$c->drawLine(-40, 0);
$c->drawLine(0, -40);

$m = new SWFMovie();
$m->setDimension(640, 480);
$m->setBackground(0xcc, 0xcc, 0xcc);

// チェッカー盤の背景を描きます
for ($y=0; $y<480; $y+=40) {
   for ($x=0; $x<640; $x+=80) {
     $i = $m->add($c);
     $i->moveTo($x, $y);
   }

   $y+=40;

   for ($x=40; $x<640; $x+=80) {
     $i = $m->add($c);
     $i->moveTo($x, $y);
   }
}

$m->add($s);

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

SWFBitmap->getHeight()

(PHP 4 >= 4.0.5)

SWFBitmap->getHeight() — ビットマップの高さを返す

説明

SWFBitmap
float **getHeight** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ビットマップの高さを返します。

返り値

ビットマップの高さをピクセルで返します。

参考

- SWFBitmap->getWidth()

SWFBitmap->getWidth()

(PHP 4 >= 4.0.5)

SWFBitmap->getWidth() — ビットマップの幅を返す

説明

SWFBitmap
float **getWidth** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ビットマップの幅を返します。

返り値

ビットマップの幅をピクセルで返します。

参考

- SWFBitmap->getHeight()

SWFButton

(No version information available, might be only in CVS)

SWFButton — SWFButton クラス

説明

クラスのメンバ

メソッド

- SWFButton->__construct()
- SWFButton->addAction()
- SWFButton->addASound()
- SWFButton->addShape()
- SWFButton->setAction()
- SWFButton->setDown()
- SWFButton->setHit()
- SWFButton->setMenu()
- SWFButton->setOver()
- SWFButton->setUp()

SWFButton->__construct()

(PHP 5)

SWFButton->__construct() — 新しいボタンを作成する

説明

SWFButton
SWFButton **__construct** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しいボタンを作成します。

例

この例では、ボタンの通常の動作 (rollover, rollon, mouseup, mousedown, noaction) を示します。

Example#1 ボタンの通常の動作

```
<?php

$f = new SWFFont("_serif");

$p = new SWFSprite();

function label($string)
{
   global $f;

   $t = new SWFTextField();
   $t->setFont($f);
   $t->addString($string);
   $t->setHeight(200);
   $t->setBounds(3200, 200);
   return $t;
}

function addLabel($string)
{
   global $p;

   $i = $p->add(label($string));
   $p->nextFrame();
   $p->remove($i);
}

$p->add(new SWFAction("stop();"));
addLabel("NO  ACTION");
addLabel("SWFBUTTON_MOUSEUP");
addLabel("SWFBUTTON_MOUSEDOWN");
addLabel("SWFBUTTON_MOUSEOVER");
addLabel("SWFBUTTON_MOUSEOUT");
addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
addLabel("SWFBUTTON_DRAGOVER");
addLabel("SWFBUTTON_DRAGOUT");

function rect($r, $g, $b)
{
   $s = new SWFShape();
   $s->setRightFill($s->addFill($r, $g, $b));
   $s->drawLine(600, 0);
   $s->drawLine(0, 600);
   $s->drawLine(-600, 0);
   $s->drawLine(0, -600);

   return $s;
}

$b = new SWFButton();
$b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP  | SWFBUTTON_HIT);
$b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
$b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

$b->addAction(new SWFAction("setTarget('/label');  gotoFrame(1);"),
           SWFBUTTON_MOUSEUP);

$b->addAction(new SWFAction("setTarget('/label');  gotoFrame(2);"),
       SWFBUTTON_MOUSEDOWN);

$b->addAction(new SWFAction("setTarget('/label');  gotoFrame(3);"),
       SWFBUTTON_MOUSEOVER);

$b->addAction(new SWFAction("setTarget('/label');  gotoFrame(4);"),
       SWFBUTTON_MOUSEOUT);

$b->addAction(new SWFAction("setTarget('/label');  gotoFrame(5);"),
       SWFBUTTON_MOUSEUPOUTSIDE);

$b->addAction(new SWFAction("setTarget('/label');  gotoFrame(6);"),
       SWFBUTTON_DRAGOVER);

$b->addAction(new SWFAction("setTarget('/label');  gotoFrame(7);"),
       SWFBUTTON_DRAGOUT);

$m = new SWFMovie();
$m->setDimension(4000, 3000);

$i = $m->add($p);
$i->setName("label");
$i->moveTo(400, 1900);

$i = $m->add($b);
$i->moveTo(400, 900);
```

```
header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

この例では、ウィンドウ上で大きな赤いボタンをドラッグさせます。 drag-and-drop ではなく、単に移動させるだけです。

Example#2 ドラッグの例

```
<?php

$s = new SWFShape();
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->drawLine(1000,0);
$s->drawLine(0,1000);
$s->drawLine(-1000,0);
$s->drawLine(0,-1000);

$b = new SWFButton();
$b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

$b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mouse
      SWFBUTTON_MOUSEDOWN);

$b->addAction(new SWFAction("stopDrag();"),
      SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

$p = new SWFSprite();
$p->add($b);
$p->nextFrame();

$m = new SWFMovie();
$i = $m->add($p);
$i->setName('test');
$i->moveTo(1000,1000);

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

SWFButton->addAction()

(PHP 4 >= 4.0.5)

SWFButton->addAction() — アクションを追加する

説明

SWFButton
void **addAction** (SWFAction $action , int $flags)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した action を、 条件を指定してボタンに追加します。

パラメータ

action

SWFAction->__construct() が返す SWFAction。

flags

以下のいずれかを指定します。 **SWFBUTTON_MOUSEOVER**、 **SWFBUTTON_MOUSEOUT**、 **SWFBUTTON_MOUSEUP**、 **SWFBUTTON_MOUSEUPOUTSIDE**、 **SWFBUTTON_MOUSEDOWN**、 **SWFBUTTON_DRAGOUT** および **SWFBUTTON_DRAGOVER**。

返り値

値を返しません。

参考

- SWFButton->addShape()
- SWFAction

SWFButton->addASound()

(PHP 5)

SWFButton->addASound() — ボタンに音を関連付ける

説明

SWFButton
SWFSoundInstance **addASound** (SWFSound $sound , int $flags)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFButton->addShape()

(PHP 4 >= 4.0.5)

SWFButton->addShape() — ボタンに図形を追加する

説明

SWFButton
void **addShape** (SWFShape $shape , int $flags)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した shape をボタンに追加します。

パラメータ

shape

SWFShape のインスタンス。

flags

以下のいずれかを指定します。 **SWFBUTTON_UP**、 **SWFBUTTON_OVER**、 **SWFBUTTON_DOWN** および **SWFBUTTON_HIT**。

SWFBUTTON_HIT は表示されず、 ボタンのヒット領域を定義します。つまり、 その図形が描画された範囲が、ボタンの "押せる" 部分となります。

返り値

値を返しません。

SWFButton->setAction()

(PHP 4 >= 4.0.5)

SWFButton->setAction() — アクションを設定する

説明

SWFButton
void **setAction** (SWFAction $action)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ボタンがクリックされたときに実行されるアクションを設定します。

これは、SWFButton->addAction() にフラグ **SWFBUTTON_MOUSEUP** を指定してコールするのと同じです。

パラメータ

action

SWFAction->__construct() が返す SWFAction。

返り値

値を返しません。

参考

- SWFButton->addAction()
- SWFAction

SWFButton->setDown()

(PHP 4 >= 4.0.5)

SWFButton->setDown() — addShape(shape, SWFBUTTON_DOWN) のエイリアス

説明

SWFButton
void **setDown** (SWFShape $shape)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて

変更される可能性があります。この関数は自己責任で使用してください。

swfbutton->setdown() は addShape(shape, SWFBUTTON_DOWN) のエイリアスです。

返り値

値を返しません。

参考

- SWFButton->addShape()
- SWFAction

SWFButton->setHit()

(PHP 4 >= 4.0.5)

SWFButton->setHit() — addShape(shape, SWFBUTTON_HIT) のエイリアス

説明

```
SWFButton
void setHit ( SWFShape $shape )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfbutton->sethit() は addShape(shape, SWFBUTTON_HIT) のエイリアスです。

返り値

値を返しません。

参考

- SWFButton->addShape()
- SWFAction

SWFButton->setMenu()

(PHP 5)

SWFButton->setMenu() — メニューボタンとしての挙動を有効にする

説明

```
SWFButton
void setMenu ( int $flag )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

flag

このパラメータを使用して、ボタンの挙動を少々変更します。 0 (off) あるいは 1 (on) を指定します。

返り値

値を返しません。

SWFButton->setOver()

(PHP 4 >= 4.0.5)

SWFButton->setOver() — addShape(shape, SWFBUTTON_OVER) のエイリアス

説明

```
SWFButton
void setOver ( SWFShape $shape )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfbutton->setover() は addShape(shape, SWFBUTTON_OVER) のエイリアスです。

返り値

値を返しません。

参考

- SWFButton->addShape()
- SWFAction

SWFButton->setUp()

(PHP 4 >= 4.0.5)

SWFButton->setUp() — addShape(shape, SWFBUTTON_UP) のエイリアス

説明

SWFButton
void **setUp** (SWFShape $shape)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfbutton->setup() は addShape(shape, SWFBUTTON_UP) のエイリアスです。

返り値

値を返しません。

参考

- SWFButton->addShape()
- SWFAction

SWFDisplayItem

(No version information available, might be only in CVS)

SWFDisplayItem — SWFDisplayItem クラス

説明

クラスのメンバ

メソッド

- SWFDisplayItem->addAction()
- SWFDisplayItem->addColor()
- SWFDisplayItem->endMask()
- SWFDisplayItem->getRot()
- SWFDisplayItem->getX()
- SWFDisplayItem->getXScale()
- SWFDisplayItem->getXSkew()
- SWFDisplayItem->getY()
- SWFDisplayItem->getYScale()
- SWFDisplayItem->getYSkew()
- SWFDisplayItem->move()
- SWFDisplayItem->moveTo()
- SWFDisplayItem->multColor()
- SWFDisplayItem->remove()
- SWFDisplayItem->rotate()
- SWFDisplayItem->rotateTo()
- SWFDisplayItem->scale()
- SWFDisplayItem->scaleTo()
- SWFDisplayItem->setDepth()
- SWFDisplayItem->setMaskLevel()
- SWFDisplayItem->setMatrix()
- SWFDisplayItem->setName()
- SWFDisplayItem->setRatio()
- SWFDisplayItem->skewX()

- SWFDisplayItem->skewXTo()
- SWFDisplayItem->skewY()
- SWFDisplayItem->skewYTo()

SWFDisplayItem->addAction()

(PHP 4 >= 4.2.0)

SWFDisplayItem->addAction() — この SWFAction を、指定した SWFSprite インスタンスに追加する

説明

SWFDisplayItem
void **addAction** (SWFAction $action , int $flags)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

action

SWFAction->__construct() が返す SWFAction。

flags

返り値

値を返しません。

参考

- SWFAction

SWFDisplayItem->addColor()

(PHP 4 >= 4.0.5)

SWFDisplayItem->addColor() — 指定した色を、このアイテムの色変換に追加する

説明

SWFDisplayItem
void **addColor** (int $red , int $green , int $blue [, int $a])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->addcolor() は、指定した色を このアイテムの色変換に追加します。色は RGB 形式で指定します。

オブジェクトは **swfshape()**、 **swfbutton()**、**swftext()** あるいは **swfsprite()** となります。オブジェクトは **swfmovie->add()** を使用して追加されていなければ なりません。

返り値

値を返しません。

返り値

値を返しません。

SWFDisplayItem->endMask()

(PHP 5)

SWFDisplayItem->endMask() — MASK レイヤを定義するもうひとつの方法

説明

SWFDisplayItem
void **endMask** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFDisplayItem->getRot()

(No version information available, might be only in CVS)

SWFDisplayItem->getRot() —

説明

SWFDisplayItem
float ***getRot*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFDisplayItem->getX()

(No version information available, might be only in CVS)

SWFDisplayItem->getX() —

説明

SWFDisplayItem
float ***getX*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *SWFDisplayItem->getY()*

SWFDisplayItem->getXScale()

(No version information available, might be only in CVS)

SWFDisplayItem->getXScale() —

説明

SWFDisplayItem
float ***getXScale*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *SWFDisplayItem->getYScale()*

SWFDisplayItem->getXSkew()

(No version information available, might be only in CVS)

SWFDisplayItem->getXSkew() —

説明

SWFDisplayItem
float ***getXSkew*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *SWFDisplayItem->getYSkew()*

SWFDisplayItem->getY()

(No version information available, might be only in CVS)

SWFDisplayItem->getY() —

説明

SWFDisplayItem
float **getY** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFDisplayItem->getX()

SWFDisplayItem->getYScale()

(No version information available, might be only in CVS)

SWFDisplayItem->getYScale() —

説明

SWFDisplayItem
float **getYScale** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFDisplayItem->getXScale()

SWFDisplayItem->getYSkew()

(No version information available, might be only in CVS)

SWFDisplayItem->getYSkew() —

説明

SWFDisplayItem
float **getYSkew** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFDisplayItem->getXSkew()

SWFDisplayItem->move()

(PHP 4 >= 4.0.5)

SWFDisplayItem->move() — *オブジェクトを相対座標系で移動する*

説明

SWFDisplayItem
void **move** (*int $dx , int $dy*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->move() *は、現在のオブジェクトを 現在位置から (dx ,dy) まで移動します。*

オブジェクトは **swfshape()**、 **swfbutton()**、**swftext()** *あるいは* **swfsprite()** *となります。オブジェクトは* **swfmovie->add()** *を使用して追加されていなければなりません。*

返り値

値を返しません。

参考

- SWFDisplayItem->moveTo()

SWFDisplayItem->moveTo()

(PHP 4 >= 4.0.5)

SWFDisplayItem->moveTo() — グローバル座標系でオブジェクトを移動する

説明

```
SWFDisplayItem
void moveTo ( int $x , int $y )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->moveto() は、グローバル座標系で 現在のオブジェクトを (x ,y) に移動します。

オブジェクトは **swfshape()**、 **swfbutton()**、**swftext()** あるいは **swfsprite()** となります。オブジェクトは **swfmovie->add()** を使用して追加されていなければなりません。

返り値

値を返しません。

参考

- SWFDisplayItem->move()

SWFDisplayItem->multColor()

(PHP 4 >= 4.0.5)

SWFDisplayItem->multColor() — アイテムの色変換を乗算する

説明

```
SWFDisplayItem
void multColor ( int $red , int $green , int $blue [, int $a ] )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->multcolor() は、指定した値で アイテムの色変換を乗算します。

オブジェクトは **swfshape()**、 **swfbutton()**、**swftext()** あるいは **swfsprite()** となります。オブジェクトは **swfmovie->add()** を使用して追加されていなければ なりません。

パラメータ

これらのパラメータは、0 から 255 までの整数値か、あるいは 0x00 から 0xFF までの十六進値です。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

a

アルファコンポーネントの値。

返り値

値を返しません。

例

この単純な例では、あなたの写真をハロウィン風に変更しています (風景写真あるいは明るい写真を使用します)。

Example#1 swfdisplayitem->multcolor() の例

```php
<?php

$b = new SWFBitmap(file_get_contents("backyard.jpg"));
// 自分の写真を使用してください (^o^)
$s = new SWFShape();
$s->setRightFill($s->addFill($b));
$s->drawLine($b->getWidth(), 0);
$s->drawLine(0, $b->getHeight());
$s->drawLine(-$b->getWidth(), 0);
$s->drawLine(0, -$b->getHeight());

$m = new SWFMovie();
$m->setDimension($b->getWidth(), $b->getHeight());

$i = $m->add($s);

for ($n=0; $n<=20; ++$n) {
```

```
  $i->multColor(1.0-$n/10, 1.0, 1.0);
  $i->addColor(0xff*$n/20, 0, 0);
  $m->nextFrame();
}

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

SWFDisplayItem->remove()

(No version information available, might be only in CVS)

SWFDisplayItem->remove() — *オブジェクトをムービーから削除する*

説明

SWFDisplayItem
void ***remove*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->remove() *は、このオブジェクトをムービーの表示リストから削除します。*

オブジェクトは、 ***swfshape()*** *か* ***swfbutton()***、***swftext()***、 ***swfsprite()*** *のいずれかとなります。 これは、* ***swfmovie->add()*** *で追加されたものでなければなりません。*

返り値

値を返しません。

参考

- *SWFMovie->add()*

SWFDisplayItem->rotate()

(PHP 4 >= 4.0.5)

SWFDisplayItem->rotate() — *相対座標で回転させる*

説明

SWFDisplayItem
void ***rotate*** *(float $angle)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->rotate() *は、現在のオブジェクトを 現在の角度から ddegrees 度回転させます。*

オブジェクトは、 ***swfshape()*** *か* ***swfbutton()***、***swftext()***、 ***swfsprite()*** *のいずれかとなります。 これは、* ***swfmovie->add()*** *で追加されたものでなければなりません。*

返り値

値を返しません。

参考

- *SWFDisplayItem->rotateTo()*

SWFDisplayItem->rotateTo()

(PHP 4 >= 4.0.5)

SWFDisplayItem->rotateTo() — *グローバル座標で回転させる*

説明

SWFDisplayItem
void ***rotateTo*** *(float $angle)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->rotateto() *は、 グローバル座標で、現在のオブジェクトを 現在の角度から ddegrees 度回転させます。*

オブジェクトは、 ***swfshape()*** *か* ***swfbutton()***、***swftext()***、 ***swfsprite()*** *のいずれかとなります。 これは、* ***swfmovie->add()*** *で追加された*

ものでなければなりません。

返り値

値を返しません。

例

この例は、背景から前面まで 3 つの回転する文字列を表示します。 かなりよくできています。

Example#1 swfdisplayitem->rotateto() の例

```
<?php
$thetext = "ming!";

$f = new SWFFont("Bauhaus 93.fdb");

$m = new SWFMovie();
$m->setRate(24.0);
$m->setDimension(2400, 1600);
$m->setBackground(0xff, 0xff, 0xff);

// たくさんの任意引数をとる関数というのは
// 常に良い考えです! 本当です!

function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
{
  global $f, $m;

  $t = new SWFText();
  $t->setFont($f);
  $t->setColor($r, $g, $b, $a);
  $t->setHeight(960);
  $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
  $t->addString($string);

  // その名前がすでに使われていない限りは、通常の PHP 変数と
  // 同様にプロパティを追加することが可能です。
  // 例: $i->scale は設定できません。なぜならそれは関数名だからです。

  $i = $m->add($t);
  $i->x = $x;
  $i->y = $y;
  $i->rot = $rot;
  $i->s = $scale;
  $i->rotateTo($rot);
  $i->scale($scale, $scale);

  // しかし、変更内容はこの関数内で閉じています。そのため、
  // 変更したオブジェクトを関数から返す必要があります。ちょっと面倒くさい……。

  return $i;
}

function step($i)
{
  $oldrot = $i->rot;
  $i->rot = 19*$i->rot/20;
  $i->x = (19*$i->x + 1200)/20;
  $i->y = (19*$i->y + 800)/20;
  $i->s = (19*$i->s + 1.0)/20;

  $i->rotateTo($i->rot);
  $i->scaleTo($i->s, $i->s);
  $i->moveTo($i->x, $i->y);

  return $i;
}

// どうです? 読みやすいでしょう。

$i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
$i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
$i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

for ($i=1; $i<=100; ++$i) {
  $i1 = step($i1);
  $i2 = step($i2);
  $i3 = step($i3);

  $m->nextFrame();
}

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

参考

- SWFDisplayItem->rotate()

SWFDisplayItem->scale()

(PHP 4 >= 4.0.5)

SWFDisplayItem->scale() — 相対座標系でオブジェクトを拡大縮小する

説明

SWFDisplayItem
void **scale** (int $dx , int $dy)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->scale() は、現在のオブジェクトを 現在の大きさから (dx ,dy) に拡大縮小します。

オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。

返り値

値を返しません。

参考

- SWFDisplayItem->scaleTo()

SWFDisplayItem->scaleTo()

(PHP 4 >= 4.0.5)

SWFDisplayItem->scaleTo() — グローバル座標系でオブジェクトを拡大縮小する

説明

SWFDisplayItem
void **scaleTo** (int $x [, int $y])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->scaleto() は、現在のオブジェクトを グローバル座標系において (dx ,dy) に拡大縮小します。

オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。

返り値

値を返しません。

参考

- SWFDisplayItem->scale()

SWFDisplayItem->setDepth()

(PHP 4 >= 4.0.5)

SWFDisplayItem->setDepth() — z オーダーを設定する

説明

SWFDisplayItem
void **setDepth** (float $depth)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->setdepth() は、オブジェクトの z オーダーを depth に設定します。デフォルト値は インスタンスが作成された(図形/文字をムービーに追加した) 際のオーダーで、 新しく作成されたものほど前面になります。2 つのオブジェクトに同じ depth を指定した場合、後から定義したもののみが移動します。

オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。

返り値

値を返しません。

SWFDisplayItem->setMaskLevel()

(PHP 5)

SWFDisplayItem->setMaskLevel() — *指定したレベルに MASK レイヤを設定する*

説明

```
SWFDisplayItem
void setMaskLevel ( int $level )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFDisplayItem->setMatrix()

(PHP 4 >= 4.0.5)

SWFDisplayItem->setMatrix() — *アイテムの変換行列を設定する*

説明

```
SWFDisplayItem
void setMatrix ( float $a , float $b , float $c , float $d , float $x , float $y )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFDisplayItem->setName()

(PHP 4 >= 4.0.5)

SWFDisplayItem->setName() — *オブジェクトの名前を設定する*

説明

```
SWFDisplayItem
void setName ( string $name )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->setname() *はオブジェクトの名前を name に設定します。 アクションスクリプトからこの名前を使用します。 これは、スプライトに対してのみ有用です。*

*オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。*

返り値

値を返しません。

SWFDisplayItem->setRatio()

(PHP 4 >= 4.0.5)

SWFDisplayItem->setRatio() — *オブジェクトの比を設定する*

説明

```
SWFDisplayItem
void setRatio ( float $ratio )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->setratio() *は、オブジェクトの比を ratio に設定します。明らかに、これは morph に対してのみ有用です。*

*オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。*

返り値

値を返しません。

例

この単純な例では、3 つの同心円をうまく変形させます。

Example#1 swfdisplayitem->setname() の例

```
<?php

$p = new SWFMorph();

$g = new SWFGradient();
$g->addEntry(0.0, 0, 0, 0);
$g->addEntry(0.16, 0xff, 0xff, 0xff);
$g->addEntry(0.32, 0, 0, 0);
$g->addEntry(0.48, 0xff, 0xff, 0xff);
$g->addEntry(0.64, 0, 0, 0);
$g->addEntry(0.80, 0xff, 0xff, 0xff);
$g->addEntry(1.00, 0, 0, 0);

$s = $p->getShape1();
$f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
$f->scaleTo(0.05);
$s->setLeftFill($f);
$s->movePenTo(-160, -120);
$s->drawLine(320, 0);
$s->drawLine(0, 240);
$s->drawLine(-320, 0);
$s->drawLine(0, -240);

$g = new SWFGradient();
$g->addEntry(0.0, 0, 0, 0);
$g->addEntry(0.16, 0xff, 0, 0);
$g->addEntry(0.32, 0, 0, 0);
$g->addEntry(0.48, 0, 0xff, 0);
$g->addEntry(0.64, 0, 0, 0);
$g->addEntry(0.80, 0, 0, 0xff);
$g->addEntry(1.00, 0, 0, 0);

$s = $p->getShape2();
$f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
$f->scaleTo(0.05);
$f->skewXTo(1.0);
$s->setLeftFill($f);
$s->movePenTo(-160, -120);
$s->drawLine(320, 0);
$s->drawLine(0, 240);
$s->drawLine(-320, 0);
$s->drawLine(0, -240);

$m = new SWFMovie();
$m->setDimension(320, 240);
$i = $m->add($p);
$i->moveTo(160, 120);

for ($n=0; $n<=1.001; $n+=0.01) {
    $i->setRatio($n);
    $m->nextFrame();
}

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

SWFDisplayItem->skewX()

(PHP 4 >= 4.0.5)

SWFDisplayItem->skewX() — X-skew を設定する

説明

SWFDisplayItem
void **skewX** (float $ddegrees)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->skewx() は、 現在の x-skew に ddegrees を追加します。

オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。

返り値

値を返しません。

参考

- SWFDisplayItem->skewXTo()
- SWFDisplayItem->skewY()
- SWFDisplayItem->skewYTo()

SWFDisplayItem->skewXTo()

(PHP 4 >= 4.0.5)

SWFDisplayItem->skewXTo() — X-skew を設定する

説明

SWFDisplayItem
void **skewXTo** (float $degrees)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->skewxto() は、 現在の x-skew を degrees に設定します。 degrees を 1.0 にすると、前方 45 度の傾きを 意味します。それより大きくするとより前方に、小さくするとより後方に なります。

オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。

返り値

値を返しません。

参考

- SWFDisplayItem->skewX()
- SWFDisplayItem->skewY()
- SWFDisplayItem->skewYTo()

SWFDisplayItem->skewY()

(PHP 4 >= 4.0.5)

SWFDisplayItem->skewY() — Y-skew を設定する

説明

SWFDisplayItem
void **skewY** (float $ddegrees)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->skewy() は、 現在の y-skew に ddegrees を追加します。

オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。

返り値

値を返しません。

参考

- SWFDisplayItem->skewYTo()
- SWFDisplayItem->skewX()
- SWFDisplayItem->skewXTo()

SWFDisplayItem->skewYTo()

(PHP 4 >= 4.0.5)

SWFDisplayItem->skewYTo() — Y-skew を設定する

説明

SWFDisplayItem
void **skewYTo** (float $degrees)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfdisplayitem->skewyto() は、 現在の y-skew を degrees に設定します。 degrees を 1.0 にすると、前方 45 度の傾きを 意味します。それより大きくするとより前方に、小さくするとより後方に なります。

オブジェクトは、**swfshape()** か **swfbutton()**、**swftext()**、 **swfsprite()** のいずれかとなります。 これは、**swfmovie->add()** で追加されたものでなければなりません。

返り値

値を返しません。

参考

- SWFDisplayItem->skewY()
- SWFDisplayItem->skewX()
- SWFDisplayItem->skewXTo()

SWFFill

(No version information available, might be only in CVS)

SWFFill — SWFFill クラス

説明

SWFFill オブジェクトは、ビットマップや 階調の塗りつぶしを変形（拡大縮小する・傾ける・回転させる）します。

swffill オブジェクトは SWFShape->addFill() メソッドで作成します。

クラスのメンバ

メソッド

- SWFFill->moveTo()
- SWFFill->rotateTo()
- SWFFill->scaleTo()
- SWFFill->skewXTo()
- SWFFill->skewYTo()

SWFFill->moveTo()

(PHP 4 >= 4.0.5)

SWFFill->moveTo() — 塗りつぶしの原点を移動する

説明

```
SWFFill
void moveTo ( int $x , int $y )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

塗りつぶしの原点を、指定したグローバル座標に移動します。

パラメータ

x

X 座標。

y

Y 座標。

返り値

値を返しません。

SWFFill->rotateTo()

(PHP 4 >= 4.0.5)

SWFFill->rotateTo() — 塗りつぶしの回転を設定する

説明

```
SWFFill
void rotateTo ( float $angle )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

塗りつぶしの回転を degrees 度に設定します。

パラメータ

angle

回転角度。

返り値

値を返しません。

SWFFill->scaleTo()

(PHP 4 >= 4.0.5)

SWFFill->scaleTo() — 塗りつぶしの倍率を設定する

説明

SWFFill
void **scaleTo** (int $x [, int $y])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

塗りつぶしの倍率を設定します。

パラメータ

x

X 座標。

y

Y 座標。

返り値

値を返しません。

SWFFill->skewXTo()

(PHP 4 >= 4.0.5)

SWFFill->skewXTo() — 塗りつぶしの x-skew を設定する

説明

SWFFill
void **skewXTo** (float $x)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

塗りつぶしの x-skew を x に設定します。

パラメータ

x

x を 1.0 にすると、前方 45 度の傾きを意味します。 それより大きくするとより前方に、小さくするとより後方になります。

返り値

値を返しません。

参考

- SWFFill->skewYTo()

SWFFill->skewYTo()

(PHP 4 >= 4.0.5)

SWFFill->skewYTo() — 塗りつぶしの y-skew を設定する

説明

SWFFill
void **skewYTo** (float $y)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

塗りつぶしの y-skew を y に設定します。

パラメータ

y

y を 1.0 にすると、前方 45 度の傾きを意味します。 それより大きくするとより前方に、小さくするとより後方になります。

返り値

値を返しません。

参考

- SWFFill->skewXTo()

SWFFont

(No version information available, might be only in CVS)

SWFFont — SWFFont クラス

説明

SWFFont オブジェクトは、フォント定義への参照を表します。 これは SWFText->setFont() および SWFTextField->setFont() で使用します。

クラスのメンバ

メソッド

- SWFFont->__construct()
- SWFFont->getAscent()
- SWFFont->getDescent()
- SWFFont->getLeading()
- SWFFont->getShape()
- SWFFont->getUTF8Width()
- SWFFont->getWidth()

SWFFont->__construct()

(PHP 5)

SWFFont->__construct() — フォント定義を読み込む

説明

SWFFont
SWFFont **__construct** (string $filename)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

filename が、FDBファイルの名前（すなわち、 ".fdb" で終る）の場合、このファイル中のフォントの定義を読み込みます。 そうでない場合、ブラウザで定義されたフォントリファレンスを作成します。

FDB ("font definition block") は、フォントに関する詳細な情報が 記述されている SWF DefineFont2 ブロック用の非常に簡単なラッパーです。 FDB ファイルは、ming の配布ファイルの util ディレクトリにある 付属の makefdb ユーティリティにより SWF ジェネレータ テンプレートファイルから作成することが可能です。

ブラウザで定義されたフォントには、フォント名以外のフォント情報が含まれていません。 フォント定義は、ムービープレイヤーに提供されると仮定します。 フォント _serif, _sans, _typewriter は、常に使用可能です。 例えば、

```
<?php
$f = newSWFFont("_sans");
?>
```

により、標準的な sans-serif フォントが指定されます。これは、HTML で <font name="sans-serif"> と指定した場合と おそらく同じになります。

SWFFont->getAscent()

(PHP 4 >= 4.0.5)

SWFFont->getAscent() — フォントの ascent (ベースライン上部の高さ) あるいは取得できない場合は 0 を返す

説明

SWFFont
float **getAscent** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *SWFFont->getDescent()*

SWFFont->getDescent()

(PHP 4 >= 4.0.5)

SWFFont->getDescent() — フォントの descent (ベースライン下部の深さ) あるいは取得できない場合は 0 を返す

説明

SWFFont
float **getDescent** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *SWFFont->getAscent()*

SWFFont->getLeading()

(PHP 4 >= 4.0.5)

SWFFont->getLeading() — フォントの leading (行間) あるいは取得できない場合は 0 を返す

説明

SWFFont
float **getLeading** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFFont->getShape()

(PHP 5)

SWFFont->getShape() — 指定した文字のグリフを文字列で返す

説明

SWFFont
string **getShape** *(int $code)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFFont->getUTF8Width()

(PHP 5)

SWFFont->getUTF8Width() — このフォントにおける指定した文字列の幅を計算する

説明

SWFFont
float **getUTF8Width** *(string $string)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *SWFFont->getWidth()*

SWFFont->getWidth()

(PHP 4 >= 4.0.5)

SWFFont->getWidth() — 文字列の幅を返す

説明

SWFFont
float **getWidth** (string $string)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swffont->getwidth() は、フォントのデフォルトの 大きさを使用して文字列 string の幅を返します。 テキストオブジェクトの大きさを使用した、このメソッドの **swftext()** バージョンを使用したくなることもあるでしょう。

参考

- SWFFont->getUTF8Width()

SWFFontChar

(No version information available, might be only in CVS)

SWFFontChar — SWFFontChar クラス

説明

クラスのメンバ

メソッド

- SWFFontChar->addChars()
- SWFFontChar->addUTF8Chars()

SWFFontChar->addChars()

(PHP 5)

SWFFontChar->addChars() — フォントをエクスポートするために、フォントに文字を追加する

説明

SWFFontChar
void **addChars** (string $char)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

参考

- SWFFontChar->addUTF8Chars()

SWFFontChar->addUTF8Chars()

(No version information available, might be only in CVS)

SWFFontChar->addUTF8Chars() — フォントをエクスポートするために、フォントに文字を追加する

説明

SWFFontChar
void **addUTF8Chars** (string $char)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

参考

- SWFFontChar->addChars()

SWFGradient

(No version information available, might be only in CVS)

SWFGradient — SWFGradient クラス

説明

クラスのメンバ

メソッド

- SWFGradient->__construct()
- SWFGradient->addEntry()

SWFGradient->__construct()

(PHP 5)

SWFGradient->__construct() — 傾きオブジェクトを作成する

説明

SWFGradient
SWFGradient **__construct** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfgradient() は、新しい SWFGradient オブジェクトを作成します。

gradient にエントリを追加すると、**swfshape->addfill()** メソッドでの塗りつぶしに gradient を使用できるようになります。

SWFGradient は以下のメソッドを保持します。**swfgradient->addentry()**。

この単純な例では、黒から白への gradient を背景に、赤い円盤を その中央に描きます。

Example#1 swfgradient() の例

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // 最初の gradient - 黒から白
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // 2 番目の gradient - 赤から透明への放射状の gradient
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type:  application/x-shockwave-flash');
  $m->output();
?>
```

SWFGradient->addEntry()

(PHP 4 >= 4.3.0)

SWFGradient->addEntry() — 傾きリストにエントリを追加する

説明

SWFGradient
void **addEntry** (float $ratio , int $red , int $green , int $blue [, int $a])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfgradient->addentry() は、gradient のリストに エントリを追加します。ratio は 0 から 1 までの 数値で、gradient にこの色が現れる割合を示します。 この値を増加させると、割合が増加します。

red 、green 、blue は色 (RGB モード) を表します。最後のパラメータ a はオプションです。

返り値

値を返しません。

SWFMorph

(No version information available, might be only in CVS)

SWFMorph — SWFMorph クラス

説明

これらのメソッドは少々変わっています。 単に newSWFMorph(shape1, shape2); とできればわかりやすいのですが、 現在は、shape2 が変形の二番目の部分を知っている必要があります (これは、描画コマンドを受け取った時点ですぐに出力を開始するからです。 そうではなく、すべてが完了してから書き出すようにすれば、かなり楽になるのですが)。

クラスのメンバ

メソッド

- SWFMorph->__construct()
- SWFMorph->getShape1()
- SWFMorph->getShape2()

SWFMorph->__construct()

(PHP 5)

SWFMorph->__construct() — 新規に SWFMorph オブジェクトを作成する

説明

SWFMorph
SWFMorph **__construct** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい SWFMorph オブジェクトを作成します。

"shape tween" とも呼ばれます。これは、趣味の悪いねじれたもので コンピュータを息詰まらせてしまいます。なんと楽しいことだ!

例

この単純な例は、大きな赤い四角形を小さな青い黒枠の四角形に変形します。

Example#1 swfmorph() の例

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0, 0, 0, 0);

  /* Note that this is backwards from normal shapes (left instead of right).
      I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
```

```
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for ($r=0.0; $r<=1.0; $r+=0.1) {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type:  application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getShape1()

(PHP 4 >= 4.0.5)

SWFMorph->getShape1() — 最初の図形へのハンドルを取得する

説明

SWFMorph
SWFShape **getShape1** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変換の最初の図形を取得します。

返り値

SWFShape オブジェクトを返します。

参考

- SWFMorph->getShape2()

SWFMorph->getShape2()

(PHP 4 >= 4.0.5)

SWFMorph->getShape2() — 最後の図形へのハンドルを取得する

説明

SWFMorph
SWFShape **getShape2** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変換の最後の図形を取得します。

返り値

SWFShape オブジェクトを返します。

参考

- SWFMorph->getShape1()

SWFMovie

(No version information available, might be only in CVS)

SWFMovie — SWFMovie クラス

説明

SWFMovie は SWF ムービーを表すオブジェクトです。

クラスのメンバ

メソッド

SWFMovie->__construct()

(PHP 5)

SWFMovie->__construct() — SWF バージョン 4 のムービーを表すムービーオブジェクトを作成する

説明

```
SWFMovie
SWFMovie __construct ( int $version )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SWF ムービーを表す、新しいムービーオブジェクトを作成します。

パラメータ

version

必要な SWF バージョン。デフォルトは 4 です。

SWFMovie->add()

(PHP 4 >= 4.3.3)

SWFMovie->add() — 任意の型のデータをムービーに追加する

説明

```
SWFMovie
mixed add ( object $instance )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SWF オブジェクトのインスタンス instance を現在のムービーに追加します。

パラメータ

instance

SWFShape や SWFText、SWFFont などの任意の型のオブジェクトインスタンス。

返り値

表示可能な型 (shape、text、button、sprite) の場合、このメソッドは、 表示リスト内のオブジェクトへのハンドル SWFDisplayItem を返します。つまり、同じ図形を をムービーに複数回追加することができ、各インスタンスについて 異なったハンドルを得ることが可能です。

参考

- SWFMovie->remove()

SWFMovie->addExport()

(No version information available, might be only in CVS)

SWFMovie->addExport() —

説明

SWFMovie
void **addExport** (SWFCharacter $char , string $name)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFMovie->addFont()

(No version information available, might be only in CVS)

SWFMovie->addFont() —

説明

SWFMovie
mixed **addFont** (SWFFont $font)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFMovie->importChar()

(No version information available, might be only in CVS)

SWFMovie->importChar() —

説明

SWFMovie
SWFSprite **importChar** (string $libswf , string $name)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFMovie->importFont()

SWFMovie->importFont()

(No version information available, might be only in CVS)

SWFMovie->importFont() —

説明

SWFMovie
SWFFontChar **importFont** (string $libswf , string $name)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFMovie->importChar()

SWFMovie->labelFrame()

(PHP 4 >= 4.3.3)

SWFMovie->labelFrame() — フレームにラベルをつける

説明

```
SWFMovie
void labelFrame ( string $label )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFMovie->nextFrame()

(PHP 4 >= 4.3.3)

SWFMovie->nextFrame() — 動画の次のフレームに移動する

説明

```
SWFMovie
void nextFrame ( void )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

動画の次のフレームに移動します。

返り値

値を返しません。

SWFMovie->output()

(PHP 4 >= 4.3.3)

SWFMovie->output() — 作成したムービーを出力する

説明

```
SWFMovie
int output ([ int $compression ] )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SWFMovie を出力します。

この関数を使用する前には、 Content-Type HTTP ヘッダを送信し、 ムービーがブラウザで表示されるようにしておきましょう。

パラメータ

compression

圧縮レベルは 0 から 9 までの値で、 gzip と同様の形式で SWF の圧縮を指定します。

このパラメータはFlash MX (6) でのみ使用可能です。

返り値

書き込んだバイト数、あるいはエラー時に **FALSE** を返します。

例

Example#1 $movie のブラウザでの表示

```
<?php
header('Content-type:  application/x-shockwave-flash');
$movie->output();
?>
```

参考

- SWFMovie->save()
- SWFMovie->saveToFile()

SWFMovie->remove()

(PHP 5 >= 5.2.1)

SWFMovie->remove() — 表示リストからオブジェクトのインスタンスを削除する

説明

```
SWFMovie
void remove ( object $instance )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したオブジェクトのインスタンス instance を表示リストから削除します。

返り値

値を返しません。

参考

- SWFMovie->add()

SWFMovie->save()

(PHP 4 >= 4.3.3)

SWFMovie->save() — SWF ムービーをファイルに保存する

説明

```
SWFMovie
int save ( string $filename [, int $compression ] )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SWF ムービーを、指定したファイル filename に保存します。

パラメータ

filename

SWF ドキュメントの保存先へのパス。

compression

圧縮レベルは 0 から 9 までの値で、 gzip と同様の形式で SWF の圧縮を指定します。

このパラメータはFlash MX (6) でのみ使用可能です。

返り値

書き込んだバイト数、あるいはエラー時に **FALSE** を返します。

参考

- SWFMovie->output()
- SWFMovie->saveToFile()

SWFMovie->saveToFile()

(PHP 4 >= 4.3.3)

SWFMovie->saveToFile() —

説明

```
SWFMovie
int saveToFile ( stream $x [, int $compression ] )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

X

compression

圧縮レベルは 0 から 9 までの値で、 gzip と同様の形式で SWF の圧縮を指定します。

このパラメータはFlash MX (6) でのみ使用可能です。

返り値

書き込んだバイト数、あるいはエラー時に **FALSE** を返します。

参考

- SWFMovie->output()
- SWFMovie->save()

SWFMovie->setbackground()

(PHP 4 >= 4.3.3)

SWFMovie->setbackground() — 背景色を設定する

説明

SWFMovie
void **setbackground** (int $red , int $green , int $blue)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

背景色を設定します。

なで rgba 版がないのでしょうか? 考えても見てください。 HTML の背景を表示させたいこともあるでしょう。 その方法はあるのですが、IE4 でしか動作しません。詳細は » http://www.macromedia.com/ で検索してください。

パラメータ

これらのパラメータは、0 から 255 までの整数値か、あるいは 0x00 から 0xFF までの十六進値となります。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

返り値

値を返しません。

SWFMovie->setDimension()

(PHP 4 >= 4.3.3)

SWFMovie->setDimension() — ムービーの幅と高さを設定する

説明

SWFMovie
void **setDimension** (int $width , int $height)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ムービーの大きさを、指定した幅 width と高さ height に設定します。

パラメータ

width

ムービーの幅。

height

ムービーの高さ。

返り値

値を返しません。

SWFMovie->setFrames()

(PHP 4 >= 4.3.3)

SWFMovie->setFrames() — 動画の総フレーム数を設定する

説明

```
SWFMovie
void setFrames ( int $number )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

動画の総フレーム数を number に設定します。

パラメータ

number

フレームの数。

返り値

値を返しません。

SWFMovie->setRate()

(PHP 4 >= 4.3.3)

SWFMovie->setRate() — 動画のフレームレートを設定する

説明

```
SWFMovie
void setRate ( int $rate )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

動画のフレームレートを rate に設定します。

プレイヤーがフレームを処理しきれない場合は、 アニメーションの速度が低下します。ただし、ストリーミング音声が 存在する場合は、音飛びを防ぐためにフレームが間引かれます。

パラメータ

rate

フレームレート。一秒あたりのフレーム数です。

返り値

値を返しません。

SWFMovie->startSound()

(No version information available, might be only in CVS)

SWFMovie->startSound() —

説明

```
SWFMovie
SWFSoundInstance startSound ( SWFSound $sound )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFMovie->stopSound()

SWFMovie->stopSound()

(No version information available, might be only in CVS)

SWFMovie->stopSound() —

説明

SWFMovie
void **stopSound** (SWFSound $sound)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

参考

- SWFMovie->startSound()

SWFMovie->streamMP3()

(PHP 4 >= 4.3.3)

SWFMovie->streamMP3() — MP3 ファイルをストリーム再生する

説明

SWFMovie
int **streamMP3** (mixed $mp3file [, float $skip])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した MP3 ファイル mp3file をストリーム再生します。

異常なデータに対する耐性は十分ではありません（最初の ID3 タグは読み飛ばしますが、その程度です）。

ムービーはそれほど賢くないので、mp3 ストリーム全体を含めるには 必要なだけの（曲の長さ * 一秒あたりのフレーム数）フレームを用意する必要があります。

パラメータ

mp3file

fopen() が返すファイルポインタ、 あるいはバイナリ文字列で指定した MP3 データ。

skip

スキップする秒数。

返り値

フレームの数を返します。

変更履歴

バージョン	説明
5.2.0	skip が追加されました。

例

Example#1 ストリーム再生の例

```
<?php
$m = new SWFMovie();
$m->setRate(12.0);
$m->streamMp3(file_get_contents("distortobass.mp3"));
// ご自身の MP3 を使用してください

// このファイルの長さは 11.85 秒で 12.0 fps です。つまり 142 フレームが必要です
$m->setFrames(142);

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

SWFMovie->writeExports()

(No version information available, might be only in CVS)

SWFMovie->writeExports() ー

説明

SWFMovie
void **writeExports** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFPrebuiltClip

(No version information available, might be only in CVS)

SWFPrebuiltClip ー *SWFPrebuiltClip* クラス

説明

クラスのメンバ

メソッド

- *SWFPrebuiltClip->__construct()*

SWFPrebuiltClip->__construct()

(No version information available, might be only in CVS)

SWFPrebuiltClip->__construct() ー *SWFPrebuiltClip* オブジェクトを返す

説明

SWFPrebuiltClip
SWFPrebuiltClip **__construct** ([string $file])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFShape

(No version information available, might be only in CVS)

SWFShape ー *SWFShape* クラス

説明

クラスのメンバ

メソッド

- *SWFShape->__construct()*
- *SWFShape->addFill()*
- *SWFShape->drawArc()*
- *SWFShape->drawCircle()*
- *SWFShape->drawCubic()*
- *SWFShape->drawCubicTo()*
- *SWFShape->drawCurve()*
- *SWFShape->drawCurveTo()*
- *SWFShape->drawGlyph()*
- *SWFShape->drawLine()*
- *SWFShape->drawLineTo()*
- *SWFShape->movePen()*
- *SWFShape->movePenTo()*
- *SWFShape->setLeftFill()*
- *SWFShape->setLine()*
- *SWFShape->setRightFill()*

SWFShape->__construct()

(PHP 5)

SWFShape->__construct() — 新しい図形オブジェクトを作成する

説明

SWFShape
SWFShape **__construct** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい SWFShape オブジェクトを作成します。

例

この単純な例では、大きな赤い楕円の 4 分の 1 の図形を描画します。

Example#1 swfshape() の例

```
<?php
$s = new SWFShape();
$s->setLine(40, 0x7f, 0, 0);
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->movePenTo(200, 200);
$s->drawLineTo(6200, 200);
$s->drawLineTo(6200, 4600);
$s->drawCurveTo(200, 4600, 200, 200);

$m = new SWFMovie();
$m->setDimension(6400, 4800);
$m->setRate(12.0);
$m->add($s);
$m->nextFrame();

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFShape->addFill()

(PHP 4 >= 4.0.5)

SWFShape->addFill() — 塗りつぶし色を図形に追加する

説明

SWFShape
SWFFill **addFill** (int $red , int $green , int $blue [, int $a])
SWFFill **addFill** (SWFBitmap $bitmap [, int $flags])
SWFFill **addFill** (SWFGradient $gradient [, int $flags])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SWFShape->addFill() は、図形の塗りつぶし形式リストに 塗りつぶしを追加します。**SWFShape->addFill()** は、 異なる三つの形式で引数を受け付けます。

red 、green 、blue は (RGB モードの) 色です。最後のパラメータ a はオプションです。

引数 bitmap は **SWFBitmap()** オブジェクトです。引数 flags は以下の定数 SWFFILL_CLIPPED_BITMAP、SWFFILL_TILED_BITMAP、SWFFILL_LINEAR_GRADIENT あるいは SWFFILL_RADIAL_GRADIENT のうちのいずれかです。 デフォルトは、SWFBitmap の場合は SWFFILL_TILED_BITMAP、 SWFGradient の場合は SWFFILL_LINEAR_GRADIENT です。

引数 gradient は **SWFGradient()** オブジェクトです。引数 flags は以下の定数 SWFFILL_RADIAL_GRADIENT あるいは SWFFILL_LINEAR_GRADIENT のうちの いずれかです。デフォルトは SWFFILL_LINEAR_GRADIENT で、 これは確実です。

SWFShape->addFill() は、以下で説明する **SWFShape->setLeftFill()** および **SWFShape->setRightFill()** 関数で使用する **SWFFill()** オブジェクトを返します。

例

この単純な例ではビットマップ上に枠を描きます。あぁ、flash player には もうひとつおかしなところがありました - morph の場合、2 番目の図形の ビットマップの変換は考慮されないようです。そのため、この例のように ビットマップは図形にあわせて伸びてしまいます...。

Example#1 SWFShape->addFill() の例

```
<?php

$p = new SWFMorph();

$b = new SWFBitmap(file_get_contents("alphafill.jpg"));
// あなた自身のビットマップを使用します
$width = $b->getWidth();
$height = $b->getHeight();

$s = $p->getShape1();
$f = $s->addFill($b, SWFFILL_TILED_BITMAP);
```

```
$f->moveTo(-$width/2, -$height/4);
$f->scaleTo(1.0, 0.5);
$s->setLeftFill($f);
$s->movePenTo(-$width/2, -$height/4);
$s->drawLine($width, 0);
$s->drawLine(0, $height/2);
$s->drawLine(-$width, 0);
$s->drawLine(0, -$height/2);

$s = $p->getShape2();
$f = $s->addFill($b, SWFFILL_TILED_BITMAP);

// これらの 2 つは何の効力も発揮しません!
$f->moveTo(-$width/4, -$height/2);
$f->scaleTo(0.5, 1.0);

$s->setLeftFill($f);
$s->movePenTo(-$width/4, -$height/2);
$s->drawLine($width/2, 0);
$s->drawLine(0, $height);
$s->drawLine(-$width/2, 0);
$s->drawLine(0, -$height);

$m = new SWFMovie();
$m->setDimension($width, $height);
$i = $m->add($p);
$i->moveTo($width/2, $height/2);

for ($n=0; $n<1.001; $n+=0.03) {
    $i->setRatio($n);
    $m->nextFrame();
}

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

参考

- SWFShape->setLeftFill()
- SWFShape->setRightFill()

SWFShape->drawArc()

(PHP 4 >= 4.0.5)

SWFShape->drawArc() — 現在の位置を中心とした半径 r の円弧を、12 時の方向から時計回りに 数えた角度 startAngle から endAngle まで描く

説明

SWFShape
void **drawArc** (float $r , float $startAngle , float $endAngle)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

参考

- SWFShape->drawCircle()

SWFShape->drawCircle()

(PHP 4 >= 4.0.5)

SWFShape->drawCircle() — 現在の位置を中心とした半径 r の円を、反時計回りに描く

説明

SWFShape
void **drawCircle** (float $r)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFShape->drawCubic()

(PHP 4 >= 4.0.5)

SWFShape->drawCubic() — 現在の位置および指定した三つの制御点を使用して 三次ベジエ曲線を描く

説明

SWFShape
int **drawCubic** (float $bx , float $by , float $cx , float $cy , float $dx , float $dy)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFShape->drawCubicTo()

SWFShape->drawCubicTo()

(No version information available, might be only in CVS)

SWFShape->drawCubicTo() — 現在の位置および指定した三つの制御点を使用して 三次ベジエ曲線を描く

説明

SWFShape
int **drawCubicTo** (float $bx , float $by , float $cx , float $cy , float $dx , float $dy)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFShape->drawCubic()

SWFShape->drawCurve()

(PHP 4 >= 4.0.5)

SWFShape->drawCurve() — 曲線を描く (相対座標)

説明

SWFShape
int **drawCurve** (int $controldx , int $controldy , int $anchordx , int $anchordy [, int $targetdx], int $targetdy)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->drawcurve() は、 (**swfshape->setline()** で設定した現在の線スタイルを 使用して) 現在のペンの位置から相対座標 (anchorx ,anchory) まで 制御点 (controlx ,controly) を使用して 2 次曲線を描きます。つまり、制御点にまず向かった後で、 anchor 点までなめらかに戻ってくるということです。

6 つのパラメータを使用すると、点 (x+targetdx , x+targetdy) に向かって 制御点 (x+controldx , y+controldy) および (x+anchordx , y+anchordy) を使用して 3 次ベジエ曲線を描きます。

参考

- SWFShape->drawCurve()

SWFShape->drawCurveTo()

(PHP 4 >= 4.0.5)

SWFShape->drawCurveTo() — 曲線を描く

説明

SWFShape
int **drawCurveTo** (int $controlx , int $controly , int $anchorx , int $anchory [, int $targetx], int $targety)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->drawcurveto() は、 (**swfshape->setline()** で設定した現在の線スタイルを 使用して) 現在のペンの位置から (anchorx ,anchory) まで 制御点 (controlx ,controly) を使用して 2 次曲線を描きます。つまり、制御点にまず向かった後で、 anchor 点までなめらかに戻ってくるということです。

6 つのパラメータを使用すると、点 (targetx , targety) に向かって 制御点 (controlx , controly) および (anchorx , anchory). を使用して 3 次ベジエ曲線を描きます。

参考

- SWFShape->drawCurveTo()

SWFShape->drawGlyph()

(PHP 4 >= 4.0.5)

SWFShape->drawGlyph() — 指定したフォントのグリフ定義を使用して、 指定した文字列の最初の文字を図形の中に描く

説明

SWFShape
void **drawGlyph** (SWFFont $font , string $character [, int $size])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFShape->drawLine()

(PHP 4 >= 4.0.5)

SWFShape->drawLine() — 線を描く （相対座標）

説明

SWFShape
void **drawLine** (int $dx , int $dy)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->drawline() は、 （**swfshape->setline()** で設定した現在の線スタイルを 使用して） 現在のペンの位置から （dx ,dy ） まで直線を描きます。

返り値

値を返しません。

参考

- SWFShape->drawLineTo()

SWFShape->drawLineTo()

(PHP 4 >= 4.0.5)

SWFShape->drawLineTo() — 線を描く

説明

SWFShape
void **drawLineTo** (int $x , int $y)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->drawlineto() は、 （**swfshape->setline()** で設定した現在の線スタイルを 使用して） 現在のペンの位置から （x ,y ） まで 図形の座標空間で直線を描きます。

返り値

値を返しません。

参考

- SWFShape->drawLine()

SWFShape->movePen()

(PHP 4 >= 4.0.5)

SWFShape->movePen() — 図形のペンを移動する(相対座標)

説明

SWFShape
void **movePen** (int $dx , int $dy)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->setrightfill() は、図形のペンを 図形の座標空間で (現在の x,現在の y) から (現在の x + dx , 現在の y + dy) まで移動します。

返り値

値を返しません。

参考

- SWFShape->movePenTo()

SWFShape->movePenTo()

(PHP 4 >= 4.0.5)

SWFShape->movePenTo() — 図形のペンを移動する

説明

SWFShape
void **movePenTo** (int $x , int $y)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->setrightfill() は、図形のペンを 図形の座標空間で (x ,y) に移動します。

返り値

値を返しません。

参考

- SWFShape->movePen()

SWFShape->setLeftFill()

(PHP 4 >= 4.0.5)

SWFShape->setLeftFill() — 左ラスタ色を設定する

説明

SWFShape
void **setLeftFill** (SWFGradient $fill)
void **setLeftFill** (int $red , int $green , int $blue [, int $a])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

何とナンセンスなことに、すべての輪郭には最大 2 つまでの塗りつぶししか 接することができません。オブジェクトをラスタライズする際に、 どの塗りつぶしがあるかを早めに知っておくと便利です。 そのため swf フォーマットではこれらを指定する必要があるのです。

swfshape->setleftfill() は、輪郭の左側の塗りつぶしを 設定します - これは、もし図形を反時計回りに描いているのなら、その内側を 表します。塗りつぶしオブジェクトは、上で説明した addFill 関数のひとつから 返される SWFFill オブジェクトです。

図形を morph の中にいれている場合、これは反転しているように見えます。 もしブラウザがクラッシュしたら、反対側の塗りつぶしを試してみてください。

swfshape->setleftfill($s->addfill($r, $g, $b [, $a])); の短縮表記です。

返り値

値を返しません。

参考

- SWFShape->setRightFill()

SWFShape->setLine()

(PHP 4 >= 4.0.5)

SWFShape->setLine() — 図形の線種を設定する

説明

SWFShape
void **setLine** (SWFShape $shape)
void **setLine** (int $width , int $red , int $green , int $blue [, int $a])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->setline() は、図形の線種を設定します。 width は線の幅です。width が 0 の場合は線種は削除されます (そして、他のすべての引数が無視されます)。 width > 0 の場合、線の色が red 、green 、blue で設定されます。最後のパラメータ a は オプションです。

使用したい線種は、使用する前にすべて定義しておく必要があります (例を参照ください)。

返り値

値を返しません。

例

この単純な例では、大きな "!#%*@" をおもしろい色と 上品なスタイルで表示します。

Example#1 swfshape->setline() の例

```
<?php
$s = new SWFShape();
$f1 = $s->addFill(0xff, 0, 0);
$f2 = $s->addFill(0xff, 0x7f, 0);
$f3 = $s->addFill(0xff, 0xff, 0);
$f4 = $s->addFill(0, 0xff, 0);
$f5 = $s->addFill(0, 0, 0xff);

// バグ: 使用する前にすべての線種を宣言する必要があります
$s->setLine(40, 0x7f, 0, 0);
$s->setLine(40, 0x7f, 0x3f, 0);
$s->setLine(40, 0x7f, 0x7f, 0);
$s->setLine(40, 0, 0x7f, 0);
$s->setLine(40, 0, 0, 0x7f);

$f = new SWFFont('Techno.fdb');

$s->setRightFill($f1);
$s->setLine(40, 0x7f, 0, 0);
$s->drawGlyph($f, '!');
$s->movePen($f->getWidth('!'), 0);

$s->setRightFill($f2);
$s->setLine(40, 0x7f, 0x3f, 0);
$s->drawGlyph($f, '#');
$s->movePen($f->getWidth('#'), 0);

$s->setRightFill($f3);
$s->setLine(40, 0x7f, 0x7f, 0);
$s->drawGlyph($f, '%');
$s->movePen($f->getWidth('%'), 0);

$s->setRightFill($f4);
$s->setLine(40, 0, 0x7f, 0);
$s->drawGlyph($f, '*');
$s->movePen($f->getWidth('*'), 0);

$s->setRightFill($f5);
$s->setLine(40, 0, 0, 0x7f);
$s->drawGlyph($f, '@');

$m = new SWFMovie();
$m->setDimension(3000,2000);
$m->setRate(12.0);
$i = $m->add($s);
$i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

返り値

値を返しません。

SWFShape->setRightFill()

(PHP 4 >= 4.0.5)

SWFShape->setRightFill() — 右ラスタ色を設定する

説明

SWFShape
void **setRightFill** (SWFGradient $fill)
void **setRightFill** (int $red , int $green , int $blue [, int $a])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfshape->setrightfill($s->addfill($r, $g, $b [, $a])); の短縮表記です。

返り値

値を返しません。

参考

- SWFShape->setLeftFill()

SWFSound

(No version information available, might be only in CVS)

SWFSound — SWFSound クラス

説明

クラスのメンバ

メソッド

- SWFSound

SWFSound

(No version information available, might be only in CVS)

SWFSound — 指定したファイルから、新しい SWFSound オブジェクトを返す

説明

SWFSound
SWFSound **__construct** (string $filename , int $flags)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFSoundInstance

(No version information available, might be only in CVS)

SWFSoundInstance — SWFSoundInstance クラス

説明

SWFSoundInstance オブジェクトは SWFSprite->startSound() および SWFMovie->startSound() メソッドが返します。

クラスのメンバ

メソッド

- SWFSoundInstance->loopCount()
- SWFSoundInstance->loopInPoint()
- SWFSoundInstance->loopOutPoint()
- SWFSoundInstance->noMultiple()

SWFSoundInstance->loopCount()

(No version information available, might be only in CVS)

SWFSoundInstance->loopCount() —

説明

SWFSoundInstance
void ***loopCount*** *(int $point)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

point

返り値

値を返しません。

参考

- *SWFSoundInstance->loopOutPoint()*

SWFSoundInstance->loopInPoint()

(No version information available, might be only in CVS)

SWFSoundInstance->loopInPoint() —

説明

SWFSoundInstance
void ***loopInPoint*** *(int $point)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

point

返り値

値を返しません。

参考

- *SWFSoundInstance->loopOutPoint()*

SWFSoundInstance->loopOutPoint()

(No version information available, might be only in CVS)

SWFSoundInstance->loopOutPoint() —

説明

SWFSoundInstance
void ***loopOutPoint*** *(int $point)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

point

返り値

値を返しません。

参考

- *SWFSoundInstance->loopInPoint()*

SWFSoundInstance->noMultiple()

(No version information available, might be only in CVS)

SWFSoundInstance->noMultiple() —

説明

SWFSoundInstance

void ***noMultiple*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFSprite

(No version information available, might be only in CVS)

SWFSprite — SWFSprite クラス

説明

SWFSprite は "ムービークリップ" ともいいます。これにより、作成したオブジェクトを 時間軸にそって動かすことができるようになります。 したがって、スプライトはムービーのメソッドの大半を保持しています。

クラスのメンバ

メソッド

- *SWFSprite->__construct()*
- *SWFSprite->add()*
- *SWFSprite->labelFrame()*
- *SWFSprite->nextFrame()*
- *SWFSprite->remove()*
- *SWFSprite->setFrames()*
- *SWFSprite->startSound()*
- *SWFSprite->stopSound()*

例

大きな赤い四角形をゆっくり回転させる例です。

Example#1 swfsprite() の例

```
<?php
$s = new SWFShape();
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->movePenTo(-500, -500);
$s->drawLineTo(500, -500);
$s->drawLineTo(500, 500);
$s->drawLineTo(-500, 500);
$s->drawLineTo(-500, -500);

$p = new SWFSprite();
$i = $p->add($s);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();

$m = new SWFMovie();
$i = $m->add($p);
$i->moveTo(1500, 1000);
$i->setName("blah");

$m->setBackground(0xff, 0xff, 0xff);
$m->setDimension(3000, 2000);

header('Content-type:  application/x-shockwave-flash');
$m->output();
?>
```

SWFSprite->__construct()

(PHP 5)

SWFSprite->__construct() — ムービークリップ（スプライト） を作成する

説明

SWFSprite
SWFSprite ***__construct*** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい SWFSprite オブジェクトを作成します。

SWFSprite->add()

(PHP 4 >= 4.0.5)

SWFSprite->add() — オブジェクトをスプライトに追加する

説明

```
SWFSprite
void add ( object $object )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

***swfsprite->add()** は、**swfshape()**・ **swfbutton()**・**swftext()**・ **swfaction()** あるいは **swfsprite()** オブジェクトを追加します。*

*表示可能な型 (**swfshape()**、 **swfbutton()**、**swftext()**、 **swfaction()** あるいは **swfsprite()**) の場合、この関数は表示リスト内でのオブジェクトのハンドルを返します。*

返り値

値を返しません。

SWFSprite->labelFrame()

(PHP 4 >= 4.3.3)

SWFSprite->labelFrame() — フレームにラベルをつける

説明

```
SWFSprite
void labelFrame ( string $label )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFSprite->nextFrame()

(PHP 4 >= 4.0.5)

SWFSprite->nextFrame() — 動画の次のフレームに移動する

説明

```
SWFSprite
void nextFrame ( void )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

***swfsprite->setframes()** は、動画の次のフレームに移動します。*

返り値

値を返しません。

SWFSprite->remove()

(PHP 4 >= 4.0.5)

SWFSprite->remove() — オブジェクトをスプライトから削除する

説明

```
SWFSprite
void remove ( object $object )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfsprite->remove() *は、スプライトから* ***swfshape()*** *・* ***swfbutton()*** *・* ***swftext()*** *・* ***swfaction()*** *あるいは* ***swfsprite()*** *オブジェクトを削除します。*

返り値

値を返しません。

SWFSprite->setFrames()

(PHP 4 >= 4.0.5)

SWFSprite->setFrames() — 動画の総フレーム数を設定する

説明

SWFSprite
void ***setFrames*** *(int $number)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swfsprite->setframes() *は、 動画の総フレーム数を numberofframes に設定します。*

返り値

値を返しません。

SWFSprite->startSound()

(No version information available, might be only in CVS)

SWFSprite->startSound() —

説明

SWFSprite
SWFSoundInstance ***startSound*** *(SWFSound $sount)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *SWFSprite->stopSound()*

SWFSprite->stopSound()

(No version information available, might be only in CVS)

SWFSprite->stopSound() —

説明

SWFSprite
void ***stopSound*** *(SWFSound $sount)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

参考

- *SWFSprite->startSound()*

SWFText

(No version information available, might be only in CVS)

SWFText — SWFText クラス

説明

クラスのメンバ

メソッド

SWFText->__construct()

(PHP 5)

SWFText->__construct() — 新しい SWFText オブジェクトを作成する

説明

```
SWFText
void __construct ( void )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい SWFText オブジェクトを作成します。

例

この単純な例は、白色の背景上に大きな黄色の文字で "PHP generates Flash with Ming" を描きます。

Example#1 swftext() の例

```
<?php
$f = new SWFFont("Techno.fdb");
$t = new SWFText();
$t->setFont($f);
$t->moveTo(200, 2400);
$t->setColor(0xff, 0xff, 0);
$t->setHeight(1200);
$t->addString("PHP generates Flash with Ming!!");

$m = new SWFMovie();
$m->setDimension(5400, 3600);

$m->add($t);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFText->addString()

(PHP 4 >= 4.0.5)

SWFText->addString() — 文字列を描画する

説明

```
SWFText
void addString ( string $string )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftext->addstring() は、文字列 string を現在のペン（カーソル）の位置に描画します。 ペンの位置は、テキストのベースラインとなります。 つまり、ベースラインより上のあるテキストは -y 方向に配置されます。

返り値

値を返しません。

参考

- SWFText->addUTF8String()

SWFText->addUTF8String()

(PHP 5)

SWFText->addUTF8String() — 現在のペンの位置に、現在のフォント・高さ・行間および色設定を使用して 指定したテキストで SWFText オブジェクトを作成する

説明

SWFText
void **addUTF8String** (string $text)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

参考

- SWFText->addString()

SWFText->getAscent()

(PHP 4 >= 4.0.5)

SWFText->getAscent() — 現在のサイズにおけるフォントの ascent（ベースライン上部の高さ）あるいは取得できない場合は 0 を返す

説明

SWFText
float **getAscent** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFText->getDescent()

SWFText->getDescent()

(PHP 4 >= 4.0.5)

SWFText->getDescent() — 現在のサイズにおけるフォントの descent（ベースライン下部の深さ）あるいは取得できない場合は 0 を返す

説明

SWFText
float **getDescent** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFText->getAscent()

SWFText->getLeading()

(PHP 4 >= 4.0.5)

SWFText->getLeading() — 現在のサイズにおけるフォントの leading（行間）あるいは取得できない場合は 0 を返す

説明

SWFText
float **getLeading** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFText->getUTF8Width()

(PHP 5)

SWFText->getUTF8Width() — 現在のフォントおよびサイズにおける指定した文字列の幅を計算する

説明

SWFText
float **getUTF8Width** (string $string)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SWFText->getWidth()

SWFText->getWidth()

(PHP 4 >= 4.0.5)

SWFText->getWidth() — 文字列の幅を計算する

説明

SWFText
float **getWidth** (string $string)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

テキストオブジェクトの 現在のフォント・大きさ・間隔における文字列 string の幅を計算して返します。

参考

- SWFText->getUTF8Width()

SWFText->moveTo()

(PHP 4 >= 4.0.5)

SWFText->moveTo() — ペンを移動する

説明

SWFText
void **moveTo** (int $x , int $y)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftext->moveto() は、ペン（あるいはカーソルと いったほうがわかりやすいでしょうか）を、テキストオブジェクトの座標空間で (x ,y) に移動します。 しかし、両方ともゼロだった場合はどこにも移動しません。 うっとうしいので修正してほしいものです。

返り値

値を返しません。

SWFText->setColor()

(PHP 4 >= 4.0.5)

SWFText->setColor() — 現在のテキスト色を設定する

説明

SWFText
void **setColor** (int $red , int $green , int $blue [, int $a])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在のテキスト色を変更します。

パラメータ

これらのパラメータは、0 から 255 までの整数値か、あるいは 0x00 から 0xFF までの十六進値です。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

a

アルファコンポーネントの値。

返り値

値を返しません。

SWFText->setFont()

(PHP 4 >= 4.0.5)

SWFText->setFont() — 現在のフォントを設定する

説明

```
SWFText
void setFont ( string $font )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftext->setfont() は、現在のフォントを font に設定します。

返り値

値を返しません。

SWFText->setHeight()

(PHP 4 >= 4.0.5)

SWFText->setHeight() — 現在のフォントの高さを設定する

説明

```
SWFText
void setHeight ( int $height )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftext->setheight() は、現在のフォントの高さを height に設定します。デフォルトは 240 です。

返り値

値を返しません。

SWFText->setSpacing()

(PHP 4 >= 4.0.5)

SWFText->setSpacing() — 現在のフォントの間隔を設定する

説明

```
SWFText
void setSpacing ( float $spacing )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftext->setspacing() は、現在のフォントの間隔を spacing に設定します。デフォルトは 1.0 です。 0 にすると、すべての文字が同じ場所に書かれます。これは実際のところ 正常に動作していません。なぜなら文字の間隔が膨らんでしまい、文字の 間に同じだけの間隔をあけることができないからです。 もうちょっとうまい説明があるとは思うのですが。 あるいは単にこれを固定の間隔にしてしまいます。 これは、ただ文字の advance

がどのように働くのかを知るためだけの ものでした……。

返り値

値を返しません。

SWFTextField

(No version information available, might be only in CVS)

SWFTextField — SWFTextField クラス

説明

クラスのメンバ

メソッド

SWFTextField->__construct()

(PHP 5)

SWFTextField->__construct() — テキストフィールドのオブジェクトを作成する

説明

SWFTextField
SWFTextField **__construct** ([int $flags])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield() は、新しいテキストフィールドオブジェクトを 作成します。テキストフィールドは、**swftext()** オブジェクトに 比べてあまり自由度がありません - 回転もできないし、縦横比を変更する拡大縮小も できません。また傾けることもできません。しかし、テキストフィールドはフォームのエントリとして使用することが可能で、ブラウザで定義されているフォントを 使用することが可能です。

オプションのフラグでテキストフィールドの振る舞いを定義します。 以下の値が設定可能です。

- SWFTEXTFIELD_DRAWBOX はテキストフィールドの輪郭線を描きます。
- SWFTEXTFIELD_HASLENGTH
- SWFTEXTFIELD_HTML は HTML タグを使用したテキストのマークアップを行います。
- SWFTEXTFIELD_MULTILINE は複数行を許可します。
- SWFTEXTFIELD_NOEDIT はユーザによるテキストの編集を禁止します。
- SWFTEXTFIELD_NOSELECT はフィールドを選択できなくします。
- SWFTEXTFIELD_PASSWORD は入力内容をぼかします。
- SWFTEXTFIELD_WORDWRAP は長いテキストを折り返します。

複数のフラグを、論理 OR で組み合わせることが 可能です。例えば以下のようになります。

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

これは、編集できないパスワードフィールドという、 まったく使えないものを作成します。

SWFTextField は以下のメソッドを保持します。**swftextfield->setfont()**、 **swftextfield->setbounds()**、**swftextfield->align()**、**swftextfield->setheight()**、**swftextfield->setleftmargin()**、 **swftextfield->setrightmargin()**、**swftextfield->setmargins()**、**swftextfield->setindentation()**、**swftextfield->setlinespacing()**、 **swftextfield->setcolor()**、**swftextfield->setname()** および **swftextfield->addstring()**。

SWFTextField->addChars()

(PHP 5)

SWFTextField->addChars() — テキストフィールド内で使用可能なフォントに文字を追加する

説明

```
SWFTextField
void addChars ( string $chars )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

参考

- SWFTextField->addString()

SWFTextField->addString()

(PHP 4 >= 4.0.5)

SWFTextField->addString() — 指定した文字列をテキストフィールドに結合する

説明

```
SWFTextField
void addString ( string $string )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setname() は、文字列 string をテキストフィールドに結合します。

返り値

値を返しません。

SWFTextField->align()

(PHP 4 >= 4.0.5)

SWFTextField->align() — テキストフィールドの配置を設定する

説明

```
SWFTextField
void align ( int $alignement )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->align() は、テキストフィールドの 配置を alignement に設定します。 alignement で使用可能な値は SWFTEXTFIELD_ALIGN_LEFT、 SWFTEXTFIELD_ALIGN_RIGHT、SWFTEXTFIELD_ALIGN_CENTER および SWFTEXTFIELD_ALIGN_JUSTIFY です。

返り値

値を返しません。

SWFTextField->setBounds()

(PHP 4 >= 4.0.5)

SWFTextField->setBounds() — テキストフィールドの幅と高さを設定する

説明

```
SWFTextField
void setBounds ( int $width , int $height )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setbounds() は、テキストフィールドの 幅を width に、そして高さを height に設定します。これを設定しない場合、Ming は

何とか適当な大きさを 判定しようとします。

返り値

値を返しません。

SWFTextField->setColor()

(PHP 4 >= 4.0.5)

SWFTextField->setColor() — テキストフィールドの色を設定する

説明

```
SWFTextField
void setColor ( int $red , int $green , int $blue [, int $a ] )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setcolor() は、テキストフィールドの 色を設定します。デフォルトは、透明でない黒です。色は、RGB で表されます。

パラメータ

これらのパラメータは、0 から 255 までの整数値、あるいは 0x00 から 0xFF までの十六進値です。

red

赤コンポーネントの値。

green

緑コンポーネントの値。

blue

青コンポーネントの値。

a

アルファコンポーネントの値。

返り値

値を返しません。

SWFTextField->setFont()

(PHP 4 >= 4.2.0)

SWFTextField->setFont() — テキストフィールドのフォントを設定する

説明

```
SWFTextField
void setFont ( string $font )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setfont() は、 テキストフィールドのフォントを [ブラウザ定義の?] font に設定します。

返り値

値を返しません。

SWFTextField->setHeight()

(PHP 4 >= 4.0.5)

SWFTextField->setHeight() — このテキストフィールドのフォントの高さを設定する

説明

```
SWFTextField
void setHeight ( int $height )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setheight() は、 このテキストフィールドのフォントの高さを height に設定します。デフォルトは 240 です。

返り値

値を返しません。

SWFTextField->setIndentation()

(PHP 4 >= 4.0.5)

SWFTextField->setIndentation() — 最初の行の字下げを設定する

説明

```
SWFTextField
void setIndentation ( int $width )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setindentation() は、テキストフィールドの最初の行の字下げを width に設定します。

返り値

値を返しません。

SWFTextField->setLeftMargin()

(PHP 4 >= 4.0.5)

SWFTextField->setLeftMargin() — テキストフィールドの左マージンの幅を設定する

説明

```
SWFTextField
void setLeftMargin ( int $width )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setleftmargin() は、テキストフィールドの左マージンの幅を width に設定します。デフォルトは 0 です。

返り値

値を返しません。

SWFTextField->setLineSpacing()

(PHP 4 >= 4.0.5)

SWFTextField->setLineSpacing() — テキストフィールドの行間を設定する

説明

```
SWFTextField
void setLineSpacing ( int $height )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setlinespacing() は、テキストフィールドの行間の高さを height に設定します。デフォルトは 40 です。

返り値

値を返しません。

SWFTextField->setMargins()

(PHP 4 >= 4.0.5)

SWFTextField->setMargins() — テキストフィールドのマージン幅を設定する

説明

```
SWFTextField
void setMargins ( int $left , int $right )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setmargins() *は、両方のマージンを一度に指定します。せっかちな人向けです。*

返り値

値を返しません。

SWFTextField->setName()

(PHP 4 >= 4.0.5)

SWFTextField->setName() — 変数名を設定する

説明

```
SWFTextField
void setName ( string $name )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setname() *は、このテキストフィールドの変数名を name に設定します。 フォームの送信やアクションスクリプトで使用します。*

返り値

値を返しません。

SWFTextField->setPadding()

(PHP 5)

SWFTextField->setPadding() — テキストフィールドのパディングを設定する

説明

```
SWFTextField
void setPadding ( float $padding )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

SWFTextField->setRightMargin()

(PHP 4 >= 4.0.5)

SWFTextField->setRightMargin() — テキストフィールドの右マージンの幅を設定する

説明

```
SWFTextField
void setRightMargin ( int $width )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

swftextfield->setrightmargin() *は、テキストフィールドの右マージンの幅を width に設定します。デフォルトは 0 です。*

返り値

値を返しません。

SWFVideoStream

(No version information available, might be only in CVS)

SWFVideoStream — SWFVideoStream クラス

説明

クラスのメンバ

メソッド

SWFVideoStream->__construct()

(No version information available, might be only in CVS)

SWFVideoStream->__construct() — SWFVideoStream オブジェクトを返す

説明

SWFVideoStream
SWFVideoStream ***__construct*** *([string $file])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFVideoStream->getNumFrames()

(No version information available, might be only in CVS)

SWFVideoStream->getNumFrames() —

説明

SWFVideoStream
int ***getNumFrames*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SWFVideoStream->setDimension()

(No version information available, might be only in CVS)

SWFVideoStream->setDimension() —

説明

SWFVideoStream
void ***setDimension*** *(int $x , int $y)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

---

ming_keypress

(PHP 5)

ming_keypress — keyPress(char) のアクションフラグを返す

説明

int ***ming_keypress*** *(string $char)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ming_setcubicthreshold

(PHP 4 >= 4.0.5, PHP 5)

ming_setcubicthreshold — 三次元の閾値を設定する

説明

void ***ming_setcubicthreshold*** *(int $threshold)*

三次ベジエ曲線を描画する際の閾値を設定します。

パラメータ

threshold

閾値。小さいほど正確になりますが、そのぶんサイズが大きくなります。

返り値

値を返しません。

---

ming_setscale

(PHP 4 >= 4.0.5, PHP 5)

ming_setscale — スケールを設定する

説明

void **ming_setscale** (int $scale)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

---

ming_setswfcompression

(PHP 5 >= 5.2.1)

ming_setswfcompression — SWF 出力の圧縮を設定する

説明

void **ming_setswfcompression** (int $level)

SWF 出力の圧縮レベルを設定する。

パラメータ

level

新しい圧縮レベル。1 から 9 までの値でなければなりません。

返り値

値を返しません。

---

ming_useconstants

(PHP 5)

ming_useconstants — 定数プールを使用する

説明

void **ming_useconstants** (int $use)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

---

ming_useswfversion

(PHP 4 >= 4.2.0, PHP 5)

ming_useswfversion — SWF のバージョンを設定する

説明

void **ming_useswfversion** (int $version)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

目次

- SWFSprite->labelFrame() — フレームにラベルをつける
- SWFSprite->nextFrame() — 動画の次のフレームに移動する
- SWFSprite->remove() — オブジェクトをスプライトから削除する
- SWFSprite->setFrames() — 動画の総フレーム数を設定する
- SWFSprite->startSound() — 説明
- SWFSprite->stopSound() — 説明
- SWFText — SWFText クラス
- SWFText->__construct() — 新しい SWFText オブジェクトを作成する
- SWFText->addString() — 文字列を描画する
- SWFText->addUTF8String() — 現在のペンの位置に、現在のフォント・高さ・行間および色設定を使用して 指定したテキストで SWFText オブジェクトを作成する
- SWFText->getAscent() — 現在のサイズにおけるフォントの ascent（ベースライン上部の高さ）あるいは取得できない場合は 0 を返す
- SWFText->getDescent() — 現在のサイズにおけるフォントの descent（ベースライン下部の深さ）あるいは取得できない場合は 0 を返す
- SWFText->getLeading() — 現在のサイズにおけるフォントの leading（行間）あるいは取得できない場合は 0 を返す
- SWFText->getUTF8Width() — 現在のフォントおよびサイズにおける指定した文字列の幅を計算する
- SWFText->getWidth() — 文字列の幅を計算する
- SWFText->moveTo() — ペンを移動する
- SWFText->setColor() — 現在のテキスト色を設定する
- SWFText->setFont() — 現在のフォントを設定する
- SWFText->setHeight() — 現在のフォントの高さを設定する
- SWFText->setSpacing() — 現在のフォントの間隔を設定する
- SWFTextField — SWFTextField クラス
- SWFTextField->__construct() — テキストフィールドのオブジェクトを作成する
- SWFTextField->addChars() — テキストフィールド内で使用可能なフォントに文字を追加する
- SWFTextField->addString() — 指定した文字列をテキストフィールドに結合する
- SWFTextField->align() — テキストフィールドの配置を設定する
- SWFTextField->setBounds() — テキストフィールドの幅と高さを設定する
- SWFTextField->setColor() — テキストフィールドの色を設定する
- SWFTextField->setFont() — テキストフィールドのフォントを設定する
- SWFTextField->setHeight() — このテキストフィールドのフォントの高さを設定する
- SWFTextField->setIndentation() — 最初の行の字下げを設定する
- SWFTextField->setLeftMargin() — テキストフィールドの左マージンの幅を設定する
- SWFTextField->setLineSpacing() — テキストフィールドの行間を設定する
- SWFTextField->setMargins() — テキストフィールドのマージン幅を設定する
- SWFTextField->setName() — 変数名を設定する
- SWFTextField->setPadding() — テキストフィールドのパディングを設定する
- SWFTextField->setRightMargin() — テキストフィールドの右マージンの幅を設定する
- SWFVideoStream — SWFVideoStream クラス
- SWFVideoStream->__construct() — SWFVideoStream オブジェクトを返す
- SWFVideoStream->getNumFrames() — 説明
- SWFVideoStream->setDimension() — 説明
- ming_keypress — keyPress(char) のアクションフラグを返す
- ming_setcubicthreshold — 三次元の閾値を設定する
- ming_setscale — スケールを設定する
- ming_setswfcompression — SWF 出力の圧縮を設定する
- ming_useconstants — 定数プールを使用する
- ming_useswfversion — SWF のバージョンを設定する

その他の関数（Misc）

導入

他のカテゴリには当てはまらない関数群を、こちらに収録しています。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

Misc の設定オプション

名前	デフォルト	変更の可否	変更履歴
ignore_user_abort	"0"	PHP_INI_ALL	
highlight.string	"#DD0000"	PHP_INI_ALL	
highlight.comment	"#FF8000"	PHP_INI_ALL	
highlight.keyword	"#007700"	PHP_INI_ALL	
highlight.bg	"#FFFFFF"	PHP_INI_ALL	PHP 6.0.0 で削除。
highlight.default	"#0000BB"	PHP_INI_ALL	
highlight.html	"#000000"	PHP_INI_ALL	
browscap	NULL	PHP_INI_SYSTEM	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

ignore_user_abort boolean

*デフォルトは **FALSE**。**TRUE** に変更すると、 クライアントが接続を破棄した後もスクリプトは終了しません。*

ignore_user_abort() も参照ください。

highlight.bg string highlight.comment string highlight.default string highlight.html string highlight.keyword string highlight.string string

構文ハイライト表示モードの色。 <font color="??????"> 形式のものは全て動作します。

browscap string

browser capabilities file (ブラウザ機能ファイル) の 名前 (例: browscap.ini) と場所。 get_browser() も参照ください。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

***CONNECTION_ABORTED** (integer)*
***CONNECTION_NORMAL** (integer)*
***CONNECTION_TIMEOUT** (integer)*
***__COMPILER_HALT_OFFSET__** (integer)*
PHP 5.1 で追加されました。

---

connection_aborted

(PHP 4, PHP 5)

connection_aborted — クライアントとの接続が切断されているかどうかを調べる

説明

*int **connection_aborted** (void)*

クライアントとの接続が切断されているかどうかを調べます。

返り値

クライアントとの接続が切断されている場合に 1、それ以外の場合に 0 を返します。

参考

- *connection_status()*
- *ignore_user_abort()*
- *PHP における接続処理についての説明は 接続処理*

---

connection_status

(PHP 4, PHP 5)

connection_status — 接続ステータスのビットフィールドを返す

説明

*int **connection_status** (void)*

接続ステータスのビットフィールドを取得します。

返り値

接続ステータスのビットフィールドを返します。これを定数 CONNECTION_XXX と比較することで、接続の状態を判断できます。

参考

- connection_aborted()
- ignore_user_abort()
- PHP における接続処理についての説明は 接続処理

---

connection_timeout

(PHP 4 <= 4.0.4)

connection_timeout — スクリプトがタイムアウトしたかどうかを調べる

説明

int **connection_timeout** (void)

スクリプトがタイムアウトしたかどうかを調べます。

返り値

スクリプトがタイムアウトした場合に 1、それ以外の場合に 0 を返します。

注意

警告

過去の関数

この関数は過去のものであり、4.0.5 以降には存在しません。

参考

- connection_status()
- PHP における接続処理についての説明は 接続処理

---

constant

(PHP 4 >= 4.0.4, PHP 5)

constant — 定数の値を返す

説明

mixed **constant** (string $name)

name で指定した定数の値を返します。

constant() はある定数の値を取得する必要があるが、 その名前が不明な場合に有用です。これは、定数が変数に保存されているか、 関数により返されるかの場合です。

この関数は クラス定数に対しても動作します。

パラメータ

name

定数名。

返り値

定数の値、あるいはその定数が定義されていない場合に **NULL** を返します。

例

Example#1 constant() の例

```php
<?php

define("MAXSIZE", 100);

echo MAXSIZE;
echo constant("MAXSIZE"); //  ひとつ前の行と同じことです

?>
```

参考

- define()
- defined()
- 定数の節

---

define

(PHP 4, PHP 5)

define — 名前を指定して定数を定義する

説明

bool **define** (string $name , mixed $value [, bool $case_insensitive])

実行時に、名前を指定して定数を定義します。

パラメータ

name

定数の名前。

value

定数の値。スカラー値あるいは null 値のみを指定できます。 スカラー値とは integer か float、string、boolean のいずれかのことです。

case_insensitive

TRUE を指定すると、定数は大文字小文字を区別しないようになります。 デフォルトでは大文字小文字を区別します。つまり CONSTANT と Constant は別の値を表すわけです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 定数の定義

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // "Hello world." を出力します
echo Constant; // "Constant" を出力し、警告が発生します

define("GREETING", "Hello you.", true);
echo GREETING; // "Hello you." を出力します
echo Greeting; // "Hello you." を出力します

?>
```

参考

- defined()
- constant()
- 定数の節

---

defined

(PHP 4, PHP 5)

defined — 指定した名前の定数が存在するかどうかを調べる

説明

bool **defined** (string $name)

指定した定数が存在し、定義されているかどうかを調べます。

> **注意**: 変数が存在するかどうかを知りたければ、isset() を利用してください。**defined()** は 定数にしか適用できません。 関数が存在するかどうかを知りたければ、 function_exists() を利用してください。

パラメータ

name

定数名。

返り値

name で指定した名前の定数が定義されている 場合に **TRUE**、その他の場合に**FALSE**を返します。

例

Example#1 定数のチェック

```
<?php
/* 引用符の使い方に注意してください。これは重要です。この例では
 * 文字列 'CONSTANT' が、定数 CONSTANT の名前かどうかを調べています。*/
if (defined('CONSTANT')) {
    echo CONSTANT;
}
?>
```

参考

- define()
- constant()
- get_defined_constants()
- function_exists()
- 定数の節

---

die

(PHP 4, PHP 5)

die — exit() と同等

説明

この言語構造は、exit() と同等です。

---

eval

(PHP 4, PHP 5)

eval — 文字列を PHP コードとして評価する

説明

mixed **eval** (string $code_str)

code_str で与えられた文字列を PHP コードとして評価します。 中でも、データベースのテキストフィールドにコードを保存し、 後で実行するためには便利です。

eval() を使用する際、注意するべき点がいくつかあります。 パーサが **eval()** の処理中に落ちないように、 渡す文字列はセミコロンで文が終了するといった有効な PHP コード である必要があります。また、code_str の 中の文字を適切にエスケープする必要があります。 HTML 出力と PHP コードを一緒に使用するために、 PHP 終了タグを使用して PHP モードを抜けることが可能です。

eval() の中で値を与えた変数は、 この後、メインスクリプトの中でもこれらの値を維持することも 覚えておいてください。

パラメータ

code_str

評価するコード文字列。 code_str には PHP 開始タグを含める必要はありません。

return 文は、文字列の評価をただちに終了します。

返り値

PHP 4 では、評価されるコードの中で return が コールされない限り、**eval()** は **NULL** を返します。 return がコールされた場合は、その値を返します。 評価されるコードの中でパースエラーが発生した場合は、 **eval()** は **FALSE** を返します。 それ以降のコードは通常通り実行されます。 **eval()** の中でのパースエラーを set_error_handler() で捕捉することはできません。

例

Example#1 eval() の例 - 簡単なテキストのマージ

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.';
echo $str. "¥n";
eval("¥$str = ¥"$str¥";");
echo $str. "¥n";
?>
```

上の例の出力は以下となります。

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

注意

注意: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

ヒント

ブラウザに直接結果を出力するすべてのものと同様に、出力制御関数 を使用してこの関数の出力をキャプチャーし、(例えば)文字列 (string)に保存することが可能です。

注意: 評価されるコードの中で致命的なエラーが発生した場合は、 スクリプト全体が終了します。

参考

- call_user_func()

---

exit

(PHP 4, PHP 5)

exit — メッセージを出力し、現在のスクリプトを終了する

説明

```
void exit ([ string $status ] )
void exit ( int $status )
```

スクリプトの実行を終了します。

パラメータ

status

status が文字列の場合は、この関数は終了直前に status を表示します。

status が integer の場合は、 その値は終了ステータスとしても用いられます。終了ステータスは 0 から 254 までの値でなければなりません。終了ステータス 255 は PHP に予約されており、使用してはいけません。ステータス 0 は、 プログラムを正常終了させる際に使用します。

注意: PHP >= 4.2.0 ではstatus が integer の場合それを表示しません。

返り値

値を返しません。

例

Example#1 exit() の例

```
<?php

$filename = '/path/to/data-file';
$file = fopen($filename, 'r')
    or exit("ファイル ($filename) をオープンできません");

?>
```

Example#2 exit() でステータスを指定する例

```
<?php

// 正常終了
exit;
exit();
exit(0);

// エラーコードつきの終了
exit(1);
exit(0376); // 八進数

?>
```

注意

注意: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

注意: この言語構造は、die() と等価です。

参考

- register_shutdown_function()

---

get_browser

(PHP 4, PHP 5)

get_browser — ユーザのブラウザの機能を取得する

説明

```
mixed get_browser ([ string $user_agent [, bool $return_array ]] )
```

ユーザのブラウザの機能を調べます。これは、browscap.ini ファイルのブラウザ情報を調べることにより行います。

パラメータ

user_agent

処理するユーザエージェント。デフォルトでは、HTTP の User-Agent ヘッダの内容を使用します。しかし、このパラメータを渡すことでこれを変更する（別のブラウザの情報を取得する）ことが可能です。

このパラメータを処理しないようにするには NULL 値を渡します。

return_array

TRUE を指定すると、この関数はオブジェクトでなく配列を返します。

返り値

情報は、オブジェクトあるいは配列形式で返されます。 たとえばブラウザのメジャーバージョン番号、マイナーバージョン番号や ID 文字列といったさまざまなデータが含まれています。また、 フレームや JavaScript、クッキーといった機能についての TRUE/FALSE 値も含んでいます。

cookies の値は、単にそのブラウザがクッキーを扱う機能を 有していることを示すだけであり、ユーザがクッキーを受け入れる設定に しているかどうかを表すものではありません。それをチェックする唯一の方法は、 いったん setcookie() でクッキーを設定してからリロードし、 その値を調べることです。

変更履歴

バージョン	説明
4.3.2	オプションのパラメータ return_array が追加されました。

例

Example#1 ユーザのブラウザについての全情報の一覧

```
<?php
echo $_SERVER['HTTP_USER_AGENT'] . "¥n¥n";

$browser = get_browser(null, true);
print_r($browser);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7) Gecko/20040803 Firefox/0.9.3

Array
(
    [browser_name_regex] => ^mozilla/5¥.0 (windows; .; windows nt 5¥.1; .*rv:.*) gecko/.* firefox/0¥.9.*$
    [browser_name_pattern] => Mozilla/5.0 (Windows; ?; Windows NT 5.1; *rv:*) Gecko/* Firefox/0.9*
    [parent] => Firefox 0.9
    [platform] => WinXP
    [browser] => Firefox
    [version] => 0.9
    [majorver] => 0
    [minorver] => 9
    [css] => 2
    [frames] => 1
    [iframes] => 1
    [tables] => 1
    [cookies] => 1
    [backgroundsounds] =>
    [vbscript] =>
    [javascript] => 1
    [javaapplets] => 1
    [activexcontrols] =>
    [cdf] =>
    [aol] =>
    [beta] => 1
    [win16] =>
    [crawler] =>
    [stripper] =>
    [wap] =>
    [netclr] =>
)
```

注意

注意: この関数が正常に機能するためには、php.ini の browscap 設定が、システム上の browscap.ini の正確な位置を 指している必要があります。
browscap.ini は PHP にはバンドルされていません。 しかし、ここで最新の » php_browscap.ini を入手することができます。
browscap.ini は多くのブラウザに関する情報をもっていますが、 データベースを最新に保つのはユーザーによる更新に依存しています。 ファイルのフォーマット自体を見ればおおよそのことがわかります。

__halt_compiler

(No version information available, might be only in CVS)

__halt_compiler — コンパイラの実行を中止する

説明

void **__halt_compiler** (void)

コンパイラの実行を中止します。 これは、インストール用ファイルのようなデータを PHP スクリプトに埋め込んでいる場合に便利です。

データの開始位置（バイト値）は、定数 **__COMPILER_HALT_OFFSET__** で定義されています。 これはファイル内で **__halt_compiler()** が使用されている場合にのみ存在します。

返り値

値を返しません。

例

Example#1 __halt_compiler() の例

```
<?php

// このファイルをオープンします
$fp = fopen(__FILE__, 'r');

// データのある位置までファイルポインタを移動します
fseek($fp, __COMPILER_HALT_OFFSET__);

// それを出力します
var_dump(stream_get_contents($fp));

// ここでスクリプトの実行を終了します
__halt_compiler();インストールデータ（例: tar, gz, PHP など..）
```

注意

注意: **__halt_compiler()** は、いちばん外側のスコープでのみ使用可能です。

---

highlight_file

(PHP 4, PHP 5)

highlight_file — ファイルの構文ハイライト表示

説明

mixed **highlight_file** (string $filename [, bool $return])

filename の中のコードを構文ハイライト表示して 出力します。色は、PHP 組込の構文ハイライタで定義されているものを使用します。

多くのサーバでは、拡張子が phps のファイルは 自動的に構文ハイライト表示されるように設定されています。例えば example.phps のようなファイルは、構文ハイライト したソースファイルとして表示されます。これを有効にするには、 httpd.conf に以下のような行を追加します。

```
AddType  application/x-httpd-php-source  .phps
```

パラメータ

filename

ハイライト表示する PHP ファイルへのパス。

return

このパラメータを **TRUE** にすると、この関数はハイライトされたコードを返します。

返り値

return が **TRUE** の場合は、 ハイライトされたコードを文字列として返し、表示しません。 それ以外の場合は、成功した場合に **TRUE**、 失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.2.1	この関数は safe_mode および open_basedir の影響を受けます。
4.2.0	パラメータ return が追加されました。

注意

警告

highlight_file() 関数を使用する場合には、 パスワードやその他潜在的なセキュリティ上の危険を生む類の重要な 情報を不注意で公開してしまわないように注意する必要があります。

注意: この関数は、このパラメータに対して内部的に出力バッファリングを使用しています。そのため、ob_start() コールバック関数の中で使用することはできません。

参考

- highlight_string()

---

highlight_string

(PHP 4, PHP 5)

highlight_string — 文字列の構文ハイライト表示

説明

mixed **highlight_string** (string $str [, bool $return])

PHP 組込みの 構文ハイライタで定義されたカラーを使用して str を構文ハイライト表示したものを出力あるいは返します。

パラメータ

str

ハイライト表示する PHP コード。開始タグを含む必要があります。

return

このパラメータを **TRUE** にすると、この関数はハイライトされたコードを返します。

返り値

return が **TRUE** の場合は、 ハイライトされたコードを文字列として返し、表示しません。 それ以外の場合は、成功した場合に **TRUE**、 失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.2.0	パラメータ return が追加されました。

例

Example#1 highlight_string() の例

```
<?php
highlight_string('<?php phpinfo(); ?>');
?>
```

上の例の出力は (PHP 4 では) 以下のようになります。

```
<code><font color="#000000">
<font color="#0000BB">&lt;?php phpinfo</font><font color="#007700">(); </font><font color="#0000BB">?&gt;</font>
</font>
</code>
```

上の例の出力は (PHP 5 では) 以下のようになります。

```
<code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php phpinfo</span><span style="color: #007700">(); </span><span style="color: #0000BB"
</span>
</code>
```

注意

注意: この関数は、このパラメータに対して内部的に出力バッファリングを使用しています。そのため、ob_start() コールバック関数の中で使用することはできません。

参考

- highlight_file()

---

ignore_user_abort

(PHP 4, PHP 5)

ignore_user_abort — クライアントの接続が切断された際にスクリプトの実行を終了するかどうかを設定する

説明

int ***ignore_user_abort*** *([bool $setting])*

クライアントの接続が切断された際にスクリプトの実行を終了するかどうかを設定します。

パラメータ

setting

省略した場合は、この関数は単に現在の設定を返します。

返り値

以前の設定を表す *boolean* 値を返します。

注意

PHP は、クライアントに情報が返ってくるまでは ユーザが切断しようとしたかどうかを検出できません。 単に *echo* 文を使っただけでは、情報が送信されたかどうかを保証できません。 *flush()* を参照ください。

参考

- *connection_aborted()*
- *connection_status()*
- *PHP* における接続処理についての説明は *接続処理*

---

pack

(PHP 4, PHP 5)

pack — データをバイナリ文字列にパックする

説明

string ***pack*** *(string $format [, mixed $args [, mixed $...]])*

指定された引数を *format* に基づいて バイナリ文字列にパックします。

この関数のアイデアは *Perl* からのものであり、フォーマット指定用の コードは *Perl* と同様に動作します。しかし、中には存在しない書式コードもあります。 たとえば *Perl* の *"u"* は存在しません。

符号付及び符号無しの区別は関数 *unpack()* にのみ 影響を与えます。関数 ***pack()*** は符号付及び符号無しの フォーマットコードのどちらでも同じ結果となることに注意しましょう。

PHP は内部的に値をマシン依存の大きさの符号付の *integer* 値として保持することにも注意してください。 このように保持するには大きすぎる符号無しの値を与えた場合、 *float* に変換する際にしばしば期待外れの結果となります。

パラメータ

format

フォーマット文字列は、 フォーマットコードの後にオプションの反復指定用引数が続く形式と なっています。反復指定用引数として整数値、または入力データの最後まで 反復を意味する * のどちらかを指定することができます。 *a*, *A*, *h*, *H* の場合、 反復数はそのデータ引数が取得する文字の数を指定します。反復数が *@* の場合、 次のデータを置く場所の絶対位置を表します。その他の場合、反復数は データ引数が使われる数を指定し、結果のバイナリ文字列にパックされます。

現在、実装されているものを以下に示します。

pack() の書式文字

コード	説明
a	NUL で埋めた文字列
A	空白で埋めた文字列
h	十六進文字列、下位ニブルが先
H	十六進文字列、上位ニブルが先
c	signed char
C	unsigned char
s	signed short (常に 16 ビット、マシンのバイトオーダー)
S	unsigned short (常に 16 ビット、マシンのバイトオーダー)
n	unsigned short (常に 16 ビット、ビッグエンディアンバイトオーダー)
v	unsigned short (常に 16 ビット、リトルエンディアンバイトオーダー)
i	signed integer (サイズおよびバイトオーダーはマシン依存)
I	unsigned integer (サイズおよびバイトオーダーはマシン依存)
l	signed long (常に 32 ビット、マシンのバイトオーダー)
L	unsigned long (常に 32 ビット、マシンのバイトオーダー)
N	unsigned long (常に 32 ビット、ビッグエンディアンバイトオーダー)
V	unsigned long (常に 32 ビット、リトルエンディアンバイトオーダー)
f	float (サイズおよび表現はマシン依存)
d	double (サイズおよび表現はマシン依存)
x	NUL バイト
X	1 バイト戻る
@	絶対位置まで NUL で埋める

args

返り値

バイナリ文字列を含むデータを返します。

例

Example#1 pack() の例

```
<?php
$binarydata = pack("nvc*", 0x1234, 0x5678, 65, 66);
?>
```

この結果のバイナリ文字列の長さは 6 バイト長で、バイト列 0x12, 0x34, 0x78, 0x56, 0x41, 0x42となります。

参考

- unpack()

---

php_check_syntax

(PHP 5 <= 5.0.4)

php_check_syntax — 指定したファイルの文法チェック（と実行）を行う

説明

bool ***php_check_syntax*** *(string $filename [, string &$error_message])*

指定したファイル filename に対して文法チェック (lint) を行い、スクリプトにエラーがないかどうかを調べます。

これは コマンドライン から ***php -l*** *を利用するのと似ていますが、* ***php_check_syntax()*** *は実際に filename を実行します（結果は出力しません）。*

たとえば、もし filename の中で関数が 定義されていた場合に ***php_check_syntax()*** *はそれを実行しますが、filename の 結果は表示されません。*

> ***注意***: *技術的な理由により、この関数は廃止され、PHP から削除されました。 かわりに、コマンドライン から php -l somefile.php を利用してください。*

パラメータ

filename

調べるファイルの名前。

error_message

error_message パラメータが指定された場合、 文法チェックの際に生成されたエラーメッセージがここに格納されます。 error_message は 参照で渡されます。

返り値

文法チェックが成功した場合に ***TRUE*** *、チェックが失敗したり file_name がオープンできなかった場合に* ***FALSE*** *を返します。*

変更履歴

バージョン	説明
5.0.5	この関数は PHP から削除されました。
5.0.3	php_check_syntax() の後に exit() をコールするとセグメンテーションフォールトが発生します。
5.0.1	*error_message* が参照渡しになりました。

例

```
php -l somefile.php
```

上の例の出力は、たとえば以下のようになります。

```
PHP Parse error: unexpected T_STRING in /tmp/somefile.php on line 81
```

参考

- include()
- is_readable()

---

php_strip_whitespace

(PHP 5)

php_strip_whitespace — コメントと空白文字を取り除いたソースを返す

説明

string **php_strip_whitespace** (string $filename)

PHP のソースコード filename からコメントと 空白文字を取り除いたものを返します。これは、スクリプトの中で 実際のコードの量がどれくらいなのかを知るのに役立つでしょう。 これは コマンドライン から **php -w** を実行するのと同じです。

パラメータ

filename

PHP ファイルへのパス。

返り値

成功した場合に処理済みのソースコード、失敗した場合に空の文字列を返します。

> **注意**: PHP 5.0.1 以降、この関数は記述どおりに動作するようになりました。 それまでは単に空の文字列を返すだけでした。このバグについての詳細な情報は、 バグ番号 » 29606 を参照ください。

例

Example#1 php_strip_whitespace() の例

```
<?php
// これは PHP のコメントです

/*
 * これも PHP のコメントです
 */

echo         php_strip_whitespace(__FILE__);
// 改行は空白文字と同じ扱いで、取り除かれます
do_nothing();
?>
```

上の例の出力は以下となります。

```
<?php
  echo  php_strip_whitespace(__FILE__);  do_nothing();  ?>
```

PHP のコメントが削除されていること、 最初の echo 文の後の改行や空白文字が削除されていることに注目しましょう。

---

show_source

(PHP 4, PHP 5)

show_source — highlight_file() のエイリアス

説明

この関数は次の関数のエイリアスです。 highlight_file().

---

sleep

(PHP 4, PHP 5)

sleep — 実行を遅延させる

説明

int **sleep** (int $seconds)

seconds で与えられた秒数ぶんプログラムの実行を遅延させます。

パラメータ

seconds

秒単位の停止時間。

返り値

成功した場合にゼロ、エラー時に **FALSE** を返します。

エラー / 例外

指定した秒数 seconds が負の場合、 この関数は **E_WARNING** を発生させます。

例

Example#1 sleep() の例

```
<?php

// 現在の時刻
echo date('h:i:s') . "¥n";

// 10 秒間遅延させる
sleep(10);

// 再開!
echo date('h:i:s') . "¥n";

?>
```

この例の（10 秒後の）出力は以下のようになります。

```
05:31:23
05:31:33
```

参考

usleep(), set_time_limit()

---

sys_getloadavg

(PHP 5 >= 5.1.3)

sys_getloadavg — システムの平均負荷を取得する

説明

array **sys_getloadavg** (void)

過去 1、5、15 分間のシステムの平均負荷（システムの実行キューの中のプロセス数）を表す三つの値を返します。

返り値

（過去 1、5、15 分間の）三つの値を array で返します。

例

Example#1 sys_getloadavg() の例

```
<?php
$load = sys_getloadavg();
if ($load[0] > 80) {
    header('HTTP/1.1 503 Too busy, try again later');
    die('Server too busy. Please try again later.');
}
?>
```

注意

> **注意**: この関数は Windows 環境にはまだ実装されていません。

---

time_nanosleep

(PHP 5)

time_nanosleep — 秒およびナノ秒単位で実行を遅延する

説明

mixed **time_nanosleep** (int $seconds , int $nanoseconds)

指定した seconds および nanoseconds の時間だけプログラムの実行を遅延させます。

パラメータ

seconds

正の整数である必要があります。

nanoseconds

十億よりも小さい正の整数である必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

シグナルによって遅延処理が中断された場合、以下の要素からなる連想配列を返します。

- seconds - 残りの秒数
- nanoseconds - 残りのナノ秒数

例

Example#1 time_nanosleep() の例

```
<?php
// 注意! もし配列が返された場合、これはうまく動作しません
if (time_nanosleep(0, 500000000)) {
    echo "0.5 秒遅延しました。\n";
}

// こちらのほうがよいでしょう
if (time_nanosleep(0, 500000000) === true) {
    echo "0.5 秒遅延しました。\n";
}

// そしてこれが最良の方法です
$nano = time_nanosleep(2, 100000);

if ($nano === true) {
    echo "2.1 秒遅延しました。\n";
} elseif ($nano === false) {
    echo "遅延に失敗しました。\n";
} elseif (is_array($nano)) {
    $seconds = $nano['seconds'];
    $nanoseconds = $nano['nanoseconds'];
    echo "シグナルによって中断しました。\n";
    echo "残りの秒数は $seconds 秒と $nanoseconds ナノ秒です。";
}
?>
```

注意

注意: この関数は Windows 環境にはまだ実装されていません。

参考

sleep(), usleep(), set_time_limit()

---

time_sleep_until

(PHP 5 >= 5.1.0)

time_sleep_until — 指定した時刻まで実行を遅延する

説明

bool **time_sleep_until** (float $timestamp)

指定した timestamp までスクリプトの実行を遅延させます。

パラメータ

timestamp

スクリプトが再開する時刻。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

過去の timestamp を指定した場合、この関数は **E_WARNING** を発生させます。

例

Example#1 time_sleep_until() の例

```
<?php

// false を返し、警告を発生します
var_dump(time_sleep_until(time()-1));

// 高速なコンピュータ上でのみ動作します。実行を 0.2 秒遅延します。
var_dump(time_sleep_until(time()+0.2));

?>
```

注意

注意: すべてのシグナルは、スクリプトが再開した後で送信されます。

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- sleep()
- usleep()
- time_nanosleep()

---

uniqid

(PHP 4, PHP 5)

uniqid — 一意な ID を生成する

説明

string **uniqid** ([string $prefix [, bool $more_entropy]])

マイクロ秒単位の現在時刻にもとづいた、接頭辞つきの一意な ID を取得します。

パラメータ

prefix

これが有用なのは、たとえば複数ホストで同時に ID を生成するような場合です。このような場合、同じマイクロ秒で同じ ID が生成されてしまう可能性があります。

空の prefix を指定すると、 返される文字列は 13 文字となります。 more_entropy が **TRUE** の場合は 23 文字となります。

more_entropy

TRUE にすると、**uniqid()** は 返り値の最後にさらに別のエントロピーを（線形合同法を使用して）追加します。これにより、結果がより一意になります。

返り値

一意な識別子を文字列で返します。

例

一意な ID またはトークンが必要な場合、そして、ネットワーク経由で ユーザにそのトークンを渡そうとする場合（例えば、セッションクッキー）、次の例のようにするのが推奨されます。

この例は、極めて予測困難な 32 文字の ID（128 ビット十六進数） を作成します。

Example#1 uniqid() の例

```
<?php
// 接頭辞なし
// PHP 5 以降でのみ動作します
$token = md5(uniqid());

// よりよい、推測しにくい方法
$better_token = md5(uniqid(rand(), true));
?>
```

変更履歴

バージョン	説明
5.0.0	パラメータ prefix が必須ではなくなりました。
4.3.1	prefix の制限が 114 文字までではなくなりました。

---

unpack

(PHP 4, PHP 5)

unpack — バイナリ文字列からデータを切り出す

説明

array **unpack** (string $format , string $data)

format に基づき、バイナリ文字列から配列に分解します。

unpack() の動作は Perl とは微妙に異なります。 Perl のように結果が連想配列に保存されることはありません。 このようにするには、別のフォーマットコードを使用してそれらを スラッシュ / で区切る必要があります。

パラメータ

format

書式コードの説明は pack() を参照ください。

data

パックされたデータ。

返り値

バイナリ文字列を切り出した要素を含む連想配列を返します。

例

Example#1 unpack() の例

```
<?php
$array = unpack("c2chars/nint", $binarydata);
?>
```

結果の配列のエントリは "chars1"、"chars2" および "int" となります。

注意

警告

PHP は内部的に整数を符号付きで保持することに注意しましょう。 大きな値の unsigned long を切り出した場合、PHP の内部で保持された値は、同じ大きさの符号付き整数となり、符号無しを指定して切出された場合でも 結果は負の数となります。

参考

- pack()

---

usleep

(PHP 4, PHP 5)

usleep — マイクロ秒単位で実行を遅延する

説明

void **usleep** *(int $micro_seconds)*

指定したマイクロ秒数だけプログラムの実行を遅延させます。

パラメータ

micro_seconds

実行を停止するマイクロ秒数。マイクロ秒とは、一秒の百万分の一です。

返り値

値を返しません。

変更履歴

バージョン	説明
5.0.0	この関数は Windows 上でも動作するようになりました。

例

Example#1 usleep() の例

```
<?php

// 現在の時刻
echo date('h:i:s') . "¥n";

// 2 秒待つ
usleep(2000000);

// 復帰!
echo date('h:i:s') . "¥n";

?>
```

上の例の出力は以下となります。

```
11:13:28
11:13:30
```

参考

- sleep()
- set_time_limit()

目次

mnoGoSearch 関数

導入

ここで示す関数により、フリーの検索エンジン mnoGoSearch (旧名は UdmSearch) へアクセスすることが可能となります。 mnoGoSearch はイントラネットおよびインターネットサーバ用の多機能な 検索エンジンソフトウエアであり、GNU ライセンスのもとで配布されています。 mnoGoSearch は、サイト内の検索から料理レシピまたは新聞検索、 ftp アーカイブ検索、新聞記事検索といった特定の検索システムといった 広い範囲のアプリケーションを構築する等といったユニークないくつかの 機能を有しています。mnoGoSearch により HTML、PDF、テキストドキュメントに 関する全文テキストインデックス作成と検索が可能になります。 mnoGoSearch は二つの部分から構成されます。最初の部分は、 インデックス機構 (indexer) です。indexer は、HTTP, FTP, NEWS サーバ またはローカルファイルにアクセスし、再帰的に全てのドキュメントを取得して、 そのドキュメントに関するメタデータを優れた効率的な手法で SQL データベースに保存します。各ドキュメントがその対応する URL で 参照された後、indexer により収集されたメタデータが後で検索処理において 使用されます。検索は、Web インターフェースにより行われます。 C CGI、PHP、Perl 用の検索フロントエンドが含まれています。

mnoGoSearchに関するより詳細な情報は、» http://www.mnogosearch.org/にあります。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 5.1.0.

> **注意**: この拡張モジュールは Windows 環境では利用できません。

要件

» http://www.mnogosearch.org/ から mnoGosearch を ダウンロードし、使用するシステムにインストールしてください。 以下の関数を使用するには、3.1.10 以降のバージョンの mnoGoSearch が インストールされている必要があります。

インストール手順

これらの関数を利用可能とするためには、オプション --with-mnogosearch により mnoGosearch サポートを指定して PHP をコンパイルする必要があります。 mnoGosearch のパスを指定せずにこのオプションを使用した場合、PHP は デフォルトで mnoGosearch が /usr/local/mnogosearch にあるものとして検索を行います。mnoGosearch を他の場所にインストール している場合には、--with-mnogosearch=DIR のようにそのパスを指定する必要があります。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/mnogosearch.

> **注意**: PHP には MySQL との接続ライブラリが組み込まれており、MySQL に アクセスすることが可能です。mnoGoSearch はこの組み込みライブラリと 互換性がなく、通常の MySQL ライブラリとの組合せでのみ動作します。 このため、mnoGoSearch を MySQL と組み合わせて使用する際には、PHP の configure に MySQL をインストールしたディレクトリを指定する必要が あります。これは mnoGoSearch に関する設定で使用され、例えば 次のようになります。 --with-mnogosearch --with-mysql=/usr

実行時設定

設定ディレクティブは定義されていません。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

UDM_FIELD_URLID (integer)
UDM_FIELD_URL (integer)
UDM_FIELD_CONTENT (integer)
UDM_FIELD_TITLE (integer)
UDM_FIELD_KEYWORDS (integer)
UDM_FIELD_DESC (integer)
UDM_FIELD_DESCRIPTION (integer)
UDM_FIELD_TEXT (integer)
UDM_FIELD_SIZE (integer)
UDM_FIELD_RATING (integer)
UDM_FIELD_SCORE (integer)
UDM_FIELD_MODIFIED (integer)
UDM_FIELD_ORDER (integer)
UDM_FIELD_CRC (integer)
UDM_FIELD_CATEGORY (integer)
UDM_FIELD_LANG (integer)
UDM_FIELD_CHARSET (integer)
UDM_PARAM_PAGE_SIZE (integer)
UDM_PARAM_PAGE_NUM (integer)
UDM_PARAM_SEARCH_MODE (integer)
UDM_PARAM_CACHE_MODE (integer)
UDM_PARAM_TRACK_MODE (integer)
UDM_PARAM_PHRASE_MODE (integer)
UDM_PARAM_CHARSET (integer)
UDM_PARAM_LOCAL_CHARSET (integer)
UDM_PARAM_BROWSER_CHARSET (integer)
UDM_PARAM_STOPTABLE (integer)
UDM_PARAM_STOP_TABLE (integer)
UDM_PARAM_STOPFILE (integer)
UDM_PARAM_STOP_FILE (integer)
UDM_PARAM_WEIGHT_FACTOR (integer)
UDM_PARAM_WORD_MATCH (integer)
UDM_PARAM_MAX_WORD_LEN (integer)
UDM_PARAM_MAX_WORDLEN (integer)
UDM_PARAM_MIN_WORD_LEN (integer)
UDM_PARAM_MIN_WORDLEN (integer)
UDM_PARAM_ISPELL_PREFIXES (integer)
UDM_PARAM_ISPELL_PREFIX (integer)
UDM_PARAM_PREFIXES (integer)
UDM_PARAM_PREFIX (integer)
UDM_PARAM_CROSS_WORDS (integer)
UDM_PARAM_CROSSWORDS (integer)
UDM_PARAM_VARDIR (integer)
UDM_PARAM_DATADIR (integer)
UDM_PARAM_HLBEG (integer)
UDM_PARAM_HLEND (integer)
UDM_PARAM_SYNONYM (integer)
UDM_PARAM_SEARCHD (integer)
UDM_PARAM_QSTRING (integer)
UDM_PARAM_REMOTE_ADDR (integer)
UDM_LIMIT_CAT (integer)
UDM_LIMIT_URL (integer)
UDM_LIMIT_TAG (integer)
UDM_LIMIT_LANG (integer)
UDM_LIMIT_DATE (integer)
UDM_PARAM_FOUND (integer)
UDM_PARAM_NUM_ROWS (integer)
UDM_PARAM_WORDINFO (integer)
UDM_PARAM_WORD_INFO (integer)
UDM_PARAM_SEARCHTIME (integer)
UDM_PARAM_SEARCH_TIME (integer)
UDM_PARAM_FIRST_DOC (integer)
UDM_PARAM_LAST_DOC (integer)
UDM_MODE_ALL (integer)
UDM_MODE_ANY (integer)
UDM_MODE_BOOL (integer)
UDM_MODE_PHRASE (integer)
UDM_CACHE_ENABLED (integer)
UDM_CACHE_DISABLED (integer)
UDM_TRACK_ENABLED (integer)
UDM_TRACK_DISABLED (integer)
UDM_PHRASE_ENABLED (integer)
UDM_PHRASE_DISABLED (integer)
UDM_CROSS_WORDS_ENABLED (integer)
UDM_CROSSWORDS_ENABLED (integer)
UDM_CROSS_WORDS_DISABLED (integer)
UDM_CROSSWORDS_DISABLED (integer)
UDM_PREFIXES_ENABLED (integer)
UDM_PREFIX_ENABLED (integer)
UDM_ISPELL_PREFIXES_ENABLED (integer)
UDM_ISPELL_PREFIX_ENABLED (integer)
UDM_PREFIXES_DISABLED (integer)
UDM_PREFIX_DISABLED (integer)
UDM_ISPELL_PREFIXES_DISABLED (integer)
UDM_ISPELL_PREFIX_DISABLED (integer)
UDM_ISPELL_TYPE_AFFIX (integer)
UDM_ISPELL_TYPE_SPELL (integer)
UDM_ISPELL_TYPE_DB (integer)
UDM_ISPELL_TYPE_SERVER (integer)
UDM_MATCH_WORD (integer)
UDM_MATCH_BEGIN (integer)
UDM_MATCH_SUBSTR (integer)
UDM_MATCH_END (integer)

---

udm_add_search_limit

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_add_search_limit — 種々の検索の制約を設定する

説明

bool **udm_add_search_limit** (resource $agent , int $var , string $val)

udm_add_search_limit() は、検索の制約を追加します。

パラメータ

agent

udm_alloc_agent() から返された、エージェントへのリンク。

var

パラメータを定義し、制限を示します。 var には以下の値が設定できます。

- **UDM_LIMIT_URL** - データベースのサブセクションにおける検索を制限するためにドキュメント URL に制限を課します。この機能は、SQL の % および _ LIKE ワイルドカードをサポートします。ただし、% は（ゼロを含む）任意の数の文字の並びで、 _ は 1 文字だけにマッチします。 例えば、http://www.example.__/catalog は http://www.example.com/catalog および http://www.example.net/catalog を表すことが可能です。
- **UDM_LIMIT_TAG** - サイト TAG 制約を定義します。indexer-conf において特定の TAG を様々なサイトやあるサイトの一部に割り付けることが可能です。 mnoGoSearch 3.1.x のタグは複数行とすることが可能で、メタ記号 % および _ を含むことが可能です。メタ記号は、 タグ集合により中からの検索が可能となります。 例えば、ABCDおよびABCEを有するリンクがあり、 検索の制約がABC_であるとすると、この検索は両方のタグについて行われます。
- **UDM_LIMIT_LANG** - ドキュメントの言語に関する制限を定義します。
- **UDM_LIMIT_CAT** - ドキュメントのカテゴリに関する制限を定義します。 カテゴリはタグ機能に似ていますが、ネストすることが可能です。 このため、あるカテゴリの中に他のカテゴリを有するといったことが可能です。 各レベルについて 2 つの文字を使用する必要があります。0-F の 16 進数または 0-Z の 36 進数を使用してください。この場合、'Auto' のような最上位のカテゴリは 01 になります。このカテゴリが 'Ford' のようなサブカテゴリを有している場合、トップカテゴリが 01(親カテゴリ)、 'Ford' が 01 となります。この結果をまとめると 0101 となります。' Auto' が 'VW' という名前の他のサブカテゴリを有している場合、 'Ford' カテゴリに属しているためにその ID は 01 になるかもしれませんが、 通常は次のカテゴリであるために 02 になります。このため、その ID は 0102 となるでしょう。VW が 'Engine' という名前のサブカテゴリを 有している場合には、その ID は再び 01 になり、'VW' が ID02 を有しており、'Auto' の ID が 01 であるので、まとめると、010201 となります。このカテゴリに関してサイトの検索を行う場合には、URL に cat=010201 を指定します。
- **UDM_LIMIT_DATE** - ドキュメントの更新時刻についての制限を定義します。

  パラメータの書式は、最初に < あるいは > があり、その後に空白を置かずに unixtime 形式の日付を続けます。例えば以下のようになります。

  ```
  <?php
  udm_add_search_limit($udm, UDM_LIMIT_DATE, "&lt;908012006");
  ?>
  ```

  > 文字が使用された場合、更新時刻がその日付以降のドキュメントのみが検索対象となります。 < の場合はそれ以前となります。

val

現在のパラメータの値を定義します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

udm_alloc_agent_array

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_alloc_agent_array — mnoGoSearch セッションを割り当てる

説明

resource **udm_alloc_agent_array** (array $databases)

udm_alloc_agent_array() は、 複数データベース接続にエージェントを作成します。

パラメータ

databases

databases の各要素にはデータベースの URL を指定します。 これは udm_alloc_agent() の最初のパラメータと同じ形式です。

返り値

成功した場合にリソースリンク ID、失敗した場合に **FALSE** を返します。

参考

- udm_alloc_agent()

---

udm_alloc_agent

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_alloc_agent — mnoGoSearch セッションを確保する

説明

resource **udm_alloc_agent** (string $dbaddr [, string $dbmode])

mnoGoSearch セッションを割り当てます。

パラメータ

dbaddr

dbaddr - URL 形式のデータベース名。SQL データベースへ接続する際のオプション (型、ホスト、データベース名、 ポート、ユーザ、パスワード)。 組み込みのテキストファイルサポートには関係ありません。 フォーマットは以下のようになります。
DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/ 現在、サポートされている DBType の値は mysql, pgsql, msql, solid, mssql, oracle, ibase です。 実際、ネーティブなライブラリのサポートは重要ではありません。しかし、 ODBC ユーザは、サポートされる値の一つを指定する必要があります。 使用するデータベース型がサポートされていない場合、unknown を代わりに使用することも可能です。

dbmode

dbmode - 単語の記憶用 SQL データベース モードを選択可能です。dbmode でとりうる値は single、multi、crc あるいは crc-multi です。 single を指定した場合、全ての単語が同じ テーブルに保存されます。multi を選択した場合、 単語はその長さに応じて別々のテーブルに保存されます。通常は "multi" モードの方が通常高速ですが、データベース上でより多くのテーブルを 必要とします。"crc" モードが選択された場合、mnoGoSearch は、単語の 代わりに CRC32 アルゴリズムで計算された 32 ビット整数の単語 ID を 保存します。このモードで必要なディスク容量はより小さいですが、 "single" および "multi" モードと比較してより高速です。 crc-multi は、"crc" モードと同じ記憶構造を 使用しますが、"multi" モードのように単語長に応じて別々のデータベースに 単語を保存します。

> **注意**: dbaddr および dbmode は、インデックス作成時に使用される これらの選択肢に一致している必要があります。

返り値

成功した場合に mnogosearch エージェント ID、失敗した場合に **FALSE** を返します。この関数は、データベースパラメータを有するセッションを生成します。

注意

> **注意**: 実際、これらの関数はデータベースへの接続をオープンする必要はなく、 よって、ログイン名やパスワードを確認しません。実際のデータベースへの 接続およびログイン/パスワード認証は、 udm_find() で行われます。

---

udm_api_version

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_api_version — mnoGoSearch API バージョンを取得する

説明

int **udm_api_version** (void)

mnoGoSearch API バージョン番号を取得します。

この関数により、利用可能なAPI関数をユーザが調べることが可能となります。 例えば、udm_get_doc_count() 関数は mnoGoSearch 3.1.11 以降でのみ利用可能です。

返り値

udm_api_version() は、mnoGoSearch API バージョン番号を返します。 例えば、mnoGoSearch 3.1.10 APIを使用している場合、 この関数は、30110 を返します。

例

Example#1 udm_api_version() の例

```
<?php
if (udm_api_version() >= 30111) {
    echo "Total number of URLs in database: " . udm_get_doc_count($udm) . "<br />¥n";
}
?>
```

---

udm_cat_list

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_cat_list — 現在のカテゴリと同じレベルのカテゴリを全て取得する

説明

array **udm_cat_list** (resource $agent , string $category)

現在のカテゴリと同じレベルの全てのカテゴリの一覧を配列として返します。

この関数は、カテゴリツリーのブラウザを開発する際に有用です。

パラメータ

agent

事前の **>udm_alloc_agent()** *のコールにより返されたエージェント ID。*

category

返り値

カテゴリツリーにおける現在の category と同レベルのカテゴリの一覧を配列で返します。

返される配列は、組で構成されています。偶数添字番号の要素にはカテゴリパス、 奇数要素には、対応するカテゴリ名が含まれます。

```
   $array[0] will contain '020300'
   $array[1] will contain 'Audi'
   $array[2] will contain '020301'
   $array[3] will contain 'BMW'
   $array[4] will contain '020302'
   $array[5] will contain 'Opel'
  ...
 etc.
```

例

以下は、現在のレベルのリンクをこの形式で表示する例です。

```
   Audi
   BMW
   Opel
   ...
```

Example#1 udm_cat_list() の例

```
<?php
  $cat_list_arr = udm_cat_list($udm_agent, $cat);
 $cat_list = '';
 for ($i=0; $i<count($cat_list_arr); $i+=2) {
    $path = $cat_list_arr[$i];
    $name = $cat_list_arr[$i+1];
    $cat_list .= "<a href=¥"$_SERVER[PHP_SELF]?cat=$path¥">$name</a><br />";
 }
?>
```

参考

- udm_cat_path()

---

udm_cat_path

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_cat_path — 現在のカテゴリへのパスを取得する

説明

array **udm_cat_path** *(resource $agent , string $category)*

ツリーのルートから category が指す現在のカテゴリまでのパスを記述した配列を返します。

パラメータ

agent

事前の **>udm_alloc_agent()** *のコールにより返されたエージェントへのリンク。*

category

返り値

返される配列は、組で構成されています。偶数添字番号の要素にはカテゴリパス、 奇数要素には、対応するカテゴリ名が含まれます。

たとえば、$array=udm_cat_path($agent, '02031D'); をコールすると以下のような配列を返します。

```
 $array[0] will contain ''
 $array[1] will contain 'Root'
 $array[2] will contain '02'
 $array[3] will contain 'Sport'
 $array[4] will contain '0203'
 $array[5] will contain 'Auto'
 $array[4] will contain '02031D'
 $array[5] will contain 'Ferrari'
```

例

Example#1 現在のカテゴリへのパスは以下の形式で指定します。 '> Root > Sport > Auto > Ferrari'

```
<?php
   $cat_path_arr = udm_cat_path($udm_agent, $cat);
  $cat_path = '';
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path = $cat_path_arr[$i];
    $name = $cat_path_arr[$i+1];
    $cat_path .= " > <a href=¥"$_SERVER[PHP_SELF]?cat=$path¥">$name</a> ";
  }
?>
```

参考

- udm_cat_list()

---

udm_check_charset

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_check_charset — 指定した charset が mnogosearch で認識されるかどうか調べる

説明

bool **udm_check_charset** (resource $agent , string $charset)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

udm_check_stored

(PHP 4 >= 4.2.0)

udm_check_stored — 保存された接続を調べる

説明

int **udm_check_stored** (resource $agent , int $link , string $doc_id)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

udm_clear_search_limits

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_clear_search_limits — mnoGoSearch 検索に関する全ての制約をクリアする

説明

bool **udm_clear_search_limits** (resource $agent)

udm_clear_search_limits() は、 検索に関する制約をリセットします。

パラメータ

agent

事前の **>udm_alloc_agent()** のコールにより返されたエージェントへのリンク。

返り値

Returns **TRUE**.

参考

- udm_add_search_limit()

---

udm_close_stored

(PHP 4 >= 4.2.0)

udm_close_stored — 保存した接続を閉じる

説明

int **udm_close_stored** (resource $agent , int $link)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

udm_crc32

(PHP 4 >= 4.2.0, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_crc32 — 指定した文字列の CRC32 チェックサムを計算する

説明

int **udm_crc32** (resource $agent , string $str)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

udm_errno

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_errno — mnoGoSearch エラー番号を取得する

説明

int **udm_errno** (resource $agent)

数字のエージェントエラーコードを受信します。

パラメータ

agent

udm_alloc_agent() へのコールから返されるエージェント ID へのリンク。

返り値

mnoGoSearch エラー番号を返します。 エラーがない場合にゼロを返します。

---

udm_error

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_error — mnoGoSearch エラーメッセージを得る

説明

string **udm_error** (resource $agent)

エージェントエラーメッセージを取得します。

パラメータ

agent

udm_alloc_agent() をコールした際に得られたエージェント ID へのリンク。

返り値

udm_error() は mnoGoSearch エラーメッセージを返します。 エラーがない場合には、空の文字列を返します。

---

udm_find

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_find — 検索を実行する

説明

resource **udm_find** (resource $agent , string $query)

検索を行います。

検索を行います。最初の引数はセッション、次の引数はクエリ本体です。 検索の実行は、探す単語を入力し、投稿ボタンを押すだけで可能です。 例えば、"mysql odbc"。この例で引用符は他のテキストから区別するために 使用されており、クエリに引用符 " を使用する必要はありません。 mnoGoSearch は、単語 "mysql" および/または単語 "odbc" を含む全ての 文書を見付けます。最も大きな重みを有する文書が最初に表示されま

す。 検索モードに ALL を使用している場合、検索は入力した単語(とその他の 単語)を共に含む文書を返します。モードANYを使用している場合、検索は、 入力した単語のどれかを含む文書のリストを返します。より高度な結果を 得たい場合には、クエリ言語を使用することも可能です。この場合は、 検索フォームで検索モード "bool" を選択する必要があります。

パラメータ

agent

udm_alloc_agent() をコールした際に得られたエージェント ID へのリンク。

query

mnoGoSearch では次の論理演算子が使用可能です。

& - 論理積。例えば、"mysql & odbc"。 mnoGoSearch は、単語 "mysql" および単語 "odbc" を含む全ての URL を見付けます。

| - 論理和。例えば、"mysql|odbc"。 mnoGoSearch は、単語 "mysql" または単語 "odbc"を含む全ての URL を見付けます。

~ - 論理否定。例えば、"mysql & ~odbc"。 mnoGoSearch は、単語 "mysql" を含み、同時に単語 "odbc" を含まない全てのURLを探します。~ は、指定した単語を 結果から除外するだけであることに注意してください。クエリ "~odbc" は何も見付けません!

() - より複雑なクエリを作成するためのグループ化コマンド。例えば、 "(mysql | msql) & ~postgres"。クエリ言語は、簡単であり、 同時に強力です。クエリは通常の論理式と同等と考えてください。

返り値

成功した場合に結果リンク ID、失敗した場合に **FALSE** を返します。

---

udm_free_agent

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_free_agent — mnoGoSearch セッションを開放する

説明

int **udm_free_agent** (resource $agent)

エージェントセッション用に確保されたメモリを開放します。

パラメータ

agent

udm_alloc_agent() から返されたエージェント ID へのリンク。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

udm_free_ispell_data

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_free_ispell_data — ispell データ用に確保されたメモリを解放する

説明

bool **udm_free_ispell_data** (int $agent)

ispell データ用に確保されたメモリを解放します。

パラメータ

agent

udm_alloc_agent() により得られるエージェントリンク ID。

返り値

udm_free_ispell_data() は、常に **TRUE** を返します。

注意

> **注意**: この関数は、mnoGoSearch バージョン 3.1.12 以降でサポートされた関数で、 これより前のバージョンではサポートされていません。

---

udm_free_res

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_free_res — mnoGoSearch 結果を開放する

説明

bool **udm_free_res** (resource $res)

結果用に確保されたメモリを開放します。

パラメータ

res

udm_find() から返された結果 ID へのリンク。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

udm_get_doc_count

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_get_doc_count — データベース内のドキュメントの総数を得る

説明

int **udm_get_doc_count** (resource $agent)

udm_get_doc_count() は、データベース内の ドキュメントの数を返します。

パラメータ

agent

udm_alloc_agent() のコールにより取得した、エージェント ID へのリンク。

返り値

ドキュメントの数を返します。

注意

> **注意**: この関数は、mnoGoSearch 3.1.11 かそれ以降のバージョンでのみ サポートされています。

---

udm_get_res_field

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_get_res_field — mnoGoSearch 結果フィールドを取得する

説明

string **udm_get_res_field** (resource $res , int $row , int $field)

Fetch a mnoGoSearch result field.

パラメータ

res

res - udm_find() から返された結果 ID へのリンク。

row

row - カレントページのリンク番号。0 から UDM_PARAM_NUM_ROWS-1 までの番号。

field

field - フィールド ID。次の値のどれか。

- UDM_FIELD_URL - ドキュメント URL フィールド。
- UDM_FIELD_CONTENT - ドキュメント Content-type フィールド (例えば、
- UDM_FIELD_CATEGORY - ドキュメントの category フィールド。 カテゴリツリーのルートから現在のカテゴリまでの完全なパスを 取得するには udm_cat_path() を使用します (このパラメータは、PHP 4.0.6 以降でのみ使用可能です)。
- UDM_FIELD_TITLE - ドキュメントの title フィールド。
- UDM_FIELD_KEYWORDS - ドキュメント keywords フィールド(META KEYWORDS タグから)。
- UDM_FIELD_DESC - ドキュメント description フィールド (META DESCRIPTION タグから)。
- UDM_FIELD_TEXT - ドキュメント body テキスト (最初の数行で ドキュメントの内容に関するアイデアを示す)。
- UDM_FIELD_SIZE - ドキュメントのサイズ。
- UDM_FIELD_URLID - リンクへのユニークな URL ID。
- UDM_FIELD_RATING - (mnoGoSearch で計算された)ページのレーティング。
- UDM_FIELD_MODIFIED - unixtime 形式の last-modified フィールド。
- UDM_FIELD_ORDER - 見つかったドキュメントの中の現在のドキュメントの数。
- UDM_FIELD_CRC - ドキュメント CRC。

返り値

udm_get_res_field() は、成功時に結果フィールド、 エラー時に **FALSE** を返します。

udm_get_res_param

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_get_res_param — mnoGoSearch 結果パラメータを取得する

説明

string **udm_get_res_param** (resource $res , int $param)

mnoGoSearch 結果パラメータを取得します。

パラメータ

res

res - udm_find() から返された結果 ID へのリンク。

param

param - パラメータ ID であり、次の値のどれかとなります。

- UDM_PARAM_NUM_ROWS - カレントのページで見付かったリンクの数。 全ての検索ページでの最後のページでの UDM_PARAM_PAGE_SIZE、残りの リンクに等しい。
- UDM_PARAM_FOUND - クエリにマッチする結果の合計の数。
- UDM_PARAM_WORDINFO - 見付かった単語に関する情報。例えば、"a good book" に関するクエリは "a: stopword, good:5637, book: 120" を返します。
- UDM_PARAM_SEARCHTIME - 秒単位の検索時間。
- UDM_PARAM_FIRST_DOC - カレントのページに表示される最初の ドキュメントの数。
- UDM_PARAM_LAST_DOC - カレントのページに表示される最後の ドキュメントの数。

返り値

udm_get_res_param() は成功時に結果パラメータを返します。 エラー時に **FALSE** を返します。

---

udm_hash32

(PHP 4 >= 4.3.3, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_hash32 — 指定した文字列の Hash32 チェックサムを返す

説明

int **udm_hash32** (resource $agent , string $str)

udm_hash32() は、文字列 str を受け取り、その 32 ビットハッシュ値を返します。

パラメータ

agent

事前の **>udm_alloc_agent()** のコールにより返されたエージェントへのリンク。

str

入力文字列。

返り値

32 ビットのハッシュ値を返します。

参考

- udm_alloc_agent()

---

udm_load_ispell_data

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_load_ispell_data — ispell データを読み込む

説明

bool **udm_load_ispell_data** (resource $agent , int $var , string $val1 , string $val2 , int $flag)

udm_load_ispell_data() は、 ispell データを読み込みます。

この関数を使用した後に ispell データに割り当てられていたメモリを 開放するには、udm_free_ispell_data() を使用します。 たとえ UDM_ISPELL_TYPE_SERVER モードを使用していたとしても同様です。

パラメータ

agent

udm_alloc_agent() のコールにより取得した、 エージェントのリンク ID。

var

ispell データの取得元を示すパラメータ。とりうる値は以下のとおりです。

- *UDM_ISPELL_TYPE_DB - ispell* データを *SQL* によって読み込むことを 示します。この場合、パラメータ *val1* および *val2* は無視されるので空白にしておきます。 *flag* は *1* に設定します。

  注意: *flag* は、指定した場所から *ispell* データを 読み込んだ後でそれを並べ替えることを示します (*ispell* を正常に 機能させるために必要です)。*ispell* データをファイルから読み込む 場合は **udm_load_ispell_data()** を何度かコールする ことになりますが、並べ替えを毎回行う必要はなく、最後にコールした 後にのみ行います。*DB* モードではすべてのデータが *1* 度のコールで 読み込まれるので、このパラメータは *1* で なければなりません。このモードでエラーが発生した場合 (たとえば *ispell* テーブルが存在しないなど)、この関数は **FALSE** を返し、 エラーコードとエラーメッセージは *udm_error()* および *udm_errno()* で取得できます。

- *UDM_ISPELL_TYPE_AFFIX - ispell* データをファイルから読み込み、また 接辞ファイルを読み込むことを指定します。この場合、*val1* はどの接辞を読み込むかを表す *2* 文字の言語コードとなります。 また *val2* はファイルのパスです。 相対パスが指定された場合は、このモジュールは *UDM_CONF_DIR* でなく 現在のパス、つまりスクリプトの実行パスからの相対パスとして検索する ことに注意しましょう。ファイルが存在しないなどの理由でエラーが 発生した場合、この関数は **FALSE** を返し、エラーメッセージが 表示されます。エラーメッセージの内容を *udm_error()* および *udm_errno()* で取得することはできません。なぜなら、これらの関数は *SQL* に 関連付けられたメッセージを返すものだからです。*UDM_ISPELL_TYPE_DB* における *flag* パラメータの説明を参照ください。

  Example#1 udm_load_ispell_data() の例

  ```
  <?php
  if ((! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'en', '/opt/ispell/en.aff', 0)) ||
      (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'ru', '/opt/ispell/ru.aff', 0)) ||
      (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'en', '/opt/ispell/en.dict', 0)) ||
      (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'ru', '/opt/ispell/ru.dict', 1))) {
      exit;
  }
  ?>
  ```

  注意: 最後のコール時にのみ *flag* を *1* とします。

- *UDM_ISPELL_TYPE_SPELL - ispell* データをファイルから読み込み、 *ispell* ディレクトリファイルを読み込むことを指定します。この場合、 *val1* はどの接辞を読み込むかを表す *2* 文字の言語コードとなります。また *val2* はファイルのパスです。 相対パスが指定された場合は、このモジュールは *UDM_CONF_DIR* でなく 現在のパス、つまりスクリプトの実行パスからの相対パスとして検索する ことに注意しましょう。ファイルが存在しないなどの理由でエラーが 発生した場合、この関数は **FALSE** を返し、エラーメッセージが 表示されます。エラーメッセージの内容を *udm_error()* および *udm_errno()* で取得することはできません。なぜなら、これらの関数は *SQL* に 関連付けられたメッセージを返すものだからです。*UDM_ISPELL_TYPE_DB* における *flag* パラメータの説明を参照ください。

  ```
  <?php
  if ((! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'en', '/opt/ispell/en.aff', 0)) ||
     (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'ru', '/opt/ispell/ru.aff', 0)) ||
     (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'en', '/opt/ispell/en.dict', 0)) ||
     (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'ru', '/opt/ispell/ru.dict', 1))) {
   exit;
    }
  ?>
  ```

  注意: 最後のコール時にのみ *flag* を *1* とします。

- *UDM_ISPELL_TYPE_SERVER - spell* サーバのサポートを有効にします。 *val1* パラメータで、*spell* サーバが 稼動しているホストのアドレスを指定します。*val2* は現在は使われていません。しかし将来のリリースでは *spell* サーバの ポート番号を指定するようになります。*flag* はこの場合は必要ありません。なぜなら *spell* サーバに保存されている データは既に並べ替えられているからです。

  spelld サーバは、設定ファイル (デフォルトは */usr/local/mnogosearch/etc/spelld.conf*) から スペルデータを読み込み、それを並び替えた上でメモリに保存します。 クライアントとサーバの間の通信方法は *2* 通りあります。 インデクサは *search.cgi* サーバから (インデクサを高速に開始するため) 全データを受け取り、それを正規化した形式でクライアント (*search.cgi*) に渡します。 *DB* モードや *TEXT* モードに比べ、これは (全スペルデータの読み込みや 並べ替えをしないことで) 一番高速に検索を実行します。

  UDM_ISPELL_TYPE_SERVER モードでは、 **udm_load_ispell_data()** 関数は実際には *ispell* データを読み込みません。単にサーバのアドレスを定義する だけです。実際には、*udm_find()* 関数で検索を 行う際にこのサーバが自動的に使用されます。*spell* サーバが 稼動していなかったりホストの指定が間違っていたりなどの理由で エラーが発生した場合、メッセージは何も返されず、*ispell* による 変換は動作しません。

  注意: この関数は、*mnoGoSearch 3.1.12*以降でのみ利用可能です。

  例
  ```
  <?php
  if (!udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SERVER, '', '', 1)) {
      echo "Error loading ispell data from server<br />¥n";
      exit;
  }
  ?>
  ```

いちばん高速なモードは *UDM_ISPELL_TYPE_SERVER* です。*UDM_ISPELL_TYPE_TEXT* はそれより遅く、*UDM_ISPELL_TYPE_DB* が一番遅くなります。この関係は *mnoGoSearch 3.1.10 - 3.1.11* において成り立ちます。 将来のバージョンでは *DB* モードの高速化を行い、*TEXT* モードより 高速になる予定です。

val1

val2

flag

返り値

成功した場合に **TRUE** *を、失敗した場合に* **FALSE** *を返します。*

例

Example#2 udm_load_ispell_data() の例

```
<?php
if (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_DB, '', '', 1)) {
  printf("Error  #%d:  '%s'¥n", udm_errno($udm), udm_error($udm));
   exit;
}
?>
```

---

udm_open_stored

(PHP 4 >= 4.2.0)

udm_open_stored — 保存した接続をオープンする

説明

int **udm_open_stored** (resource $agent , string $storedaddr)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

udm_set_agent_param

(PHP 4 >= 4.0.5, PHP 5 <= 5.0.5, PECL mnogosearch:1.0.0)

udm_set_agent_param — mnoGoSearch エージェントのセッションパラメータを設定する

説明

bool **udm_set_agent_param** (resource $agent , int $var , string $val)

mnoGoSearch セッションパラメータを定義します。

パラメータ

agent

udm_alloc_agent() のコールにより取得した、 エージェントのリンク ID。

var

以下のパラメータおよびそれらの値が利用可能です。

- UDM_PARAM_PAGE_NUM - 検索結果のページ番号を選択するために 使用されます (結果は、ページ毎に UDM_PARAM_PAGE_SIZE 個の結果を有する 0 から始まるページで返されます)。
- UDM_PARAM_PAGE_SIZE - 1 ページに表示される検索結果の数です。
- UDM_PARAM_SEARCH_MODE - 検索モード。次の値が利用可能です。 UDM_MODE_ALL - 全ての単語を検索します。; UDM_MODE_ANY - いずれかの単語で検索します。; UDM_MODE_PHRASE - 熟語で検索します。; UDM_MODE_BOOL - 論理値で検索します。論理値検索に関する詳細については udm_find() を参照ください。
- UDM_PARAM_CACHE_MODE - 検索結果のキャッシュモードをオンまたは オフにします。有効の場合、検索エンジンは検索結果をディスクに 保存します。似たような検索が後で実行された場合、エンジンは より高速にキャッシュから結果を得ることが可能です。 利用可能な値: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - クエリ追跡モードをオンまたはオフにします。 バージョン 3.1.2 以降、mnoGoSearch はクエリ追跡モードをサポート しています。追跡は SQL バージョンにのみ実装されており、組み込み データベースでは利用できません。追跡を使用するには、テーブルを 追跡サポート用に作成する必要があります。MySQL の場合、 create/mysql/track.txt を使用してください。検索実行時に、 フロントエンドはクエリ単語、見つけたドキュメントの数、カレントの 秒単位の Unix タイムスタンプを保存するためにこれらのテーブルを 使用します。 利用可能な値: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - 熟語を用いたインデックスデータベースか どうかを定義します (indexer.confにおける"phrase" パラメータ)。 使用可能な値: UDM_PHRASE_ENABLED および UDM_PHRASE_DISABLED。 熟語検索が有効な場合 (UDM_PHRASE_ENABLED) でも、全てのモード (ANY、ALL、BOOL、PHRASE) で検索が可能であることに注意してください。 mnoGoSearch のバージョン 3.1.10 で、SQL と組み込みデータベース モードでのみ熟語検索がサポートされました。一方、3.1.11 で キャッシュモードでも熟語検索がサポートされ始めました。熟語検索の例: "Arizona desert" - このクエリは、"Arizona desert" を熟語として含む 全てのドキュメントを返します。熟語の前後に二重引用符が 必要であることに注意してください。
- UDM_PARAM_CHARSET - ローカルな charset を定義します。利用可能な値: mnoGoSearch によりサポートされるcharset、 例えば、koi8-r, cp1251, ...
- UDM_PARAM_STOPFILE - stopword ファイルの名前とパスを定義します。 (mnoGoSearch と若干違いがあります。つまり、mnoGoSearch においては、 相対パスまたはパスが入力されなかった場合、UDM_CONF_DIR からの 相対パスとしてこのファイルが探されます。一方このモジュールは、 カレントのパス、すなわち PHP スクリプトが実行されるパスからの 相対パスで探します)。
- UDM_PARAM_STOPTABLE - 指定した SQL テーブルから停止単語をロード します。複数の StopwordTable コマンドを使用可能です。このコマンドは、 SQL データベースサポートを有効にせずにコンパイルした場合は 使用できません。
- UDM_PARAM_WEIGHT_FACTOR - 指定したドキュメント部分の重み係数を 表します。現在、body, title, keywords, description, url がサポートされています。この機能を有効にするには、indexer.conf の *Weight コマンドに 2 の累乗を指定してください。ここで次のような重みを指定した場合を考えてみましょう。

```
URLWeight      1
BodyWeight     2
TitleWeight    4
 KeywordWeight 8
DescWeight     16
```

同じドキュメントにいくつかの単語が複数回現れる場合、 indexerが単語の重みに OR 演算子を使用するので、異なった ドキュメントの部分に単語が現れる回数を検出可能です。本文にのみ現れる 単語は、(2進標記で) 重み集合 00000010 を有します。すべての ドキュメント部分で使用される単語は、重み集合 00011111 を有します。 このパラメータの値は、16進数文字列 ABCDE です。各桁は、単語重みの 対応するビットの因子です。上で指定した重み設定は次のようになります。

```
E is a factor for weight 1  (URL Weight bit)
D is a factor for weight 2  (BodyWeight bit)
C is a factor for weight 4  (TitleWeight bit)
B is a factor for weight 8  (KeywordWeight bit)
A is a factor for weight 16 (DescWeight bit)
```

例: UDM_PARAM_WEIGHT_FACTOR=00001 は、URL のみを検索します。 UDM_PARAM_WEIGHT_FACTOR=00100 は、Title のみを検索します。 UDM_PARAM_WEIGHT_FACTOR=11100 は、Title, Keywords, Desctription を 検索しますが、URL と Body は検索しません。 UDM_PARAM_WEIGHT_FACTOR=F9421 は、次の検索を行います。

```
Description with factor 15  (F hex)
Keywords with factor 9
Title with factor  4
Body with factor 2
URL with factor 1
```

UDM_PARAM_WEIGHT_FACTOR 変数が省略された場合、元の重みの値は ソート結果から引き出されます。上記で指定した重み設定の場合、 ドキュメントの Description は、最大重み 16 を有するドキュメントの description を意味します。

- UDM_PARAM_WORD_MATCH - 単語にマッチ。このパラメータを単語検索型を 選択する際に使用可能です。この機能は、"single" および "multi" モードで SQL データベースおよび組み込みのデータベースを使用する 場合にのみ動作します。この機能は、cachemode および他のモードでは 動作しません。これは、これらのモードが単語 CRC を使用しており、 部分文字列検索をサポートしていないからです。 利用可能な値は次のようになります。 UDM_MATCH_BEGIN - 単語の始めにマッチ UDM_MATCH_END - 単語の終りにマッチ UDM_MATCH_WORD - 単語全体にマッチ UDM_MATCH_SUBSTR - 単語の部分文字列にマッチ
- UDM_PARAM_MIN_WORD_LEN - 単語の最短長を定義します。 この制限より短い単語は、stopword とみなされます。このパラメータは その値自身も範囲に含むことに注意してください。つまり、 UDM_PARAM_MIN_WORD_LEN=3 の場合、3 文字の長さの単語は stopword と 見なされませんが、2 文字の単語はstopwordとみなされます。 デフォルトは 1 です。
- UDM_PARAM_ISPELL_PREFIXES - 利用可能な値は UDM_PREFIXES_ENABLED および UDM_PREFIXES_DISABLED で、それぞれプレフィックスの使用を有効あるいは 無効とします。例えば検索クエリに "tested" が含まれていた場合、 "test" や "testing" などといった単語も対象とします。デフォルトでは このようなサフィックスのみがサポートされています。プレフィックスは しばしば単語の意味を変えてしまいます。たとえば、"tested" を検索した際の結果として "untested" が出てきてほしいと思う人は ほとんどいないでしょう。プレフィックスのサポートは、サイトの スペルチェックのために使用すると便利です。ispell を有効にするには、udm_load_ispell_data() で ispell データを 読み込む必要があります。
- UDM_PARAM_CROSS_WORDS - クロスワードのサポートを有効あるいは無効にします。 利用可能な値: UDM_CROSS_WORDS_ENABLED および UDM_CROSS_WORDS_DISABLED クロスワード機能により、 と の間の 単語もリンク先ドキュメントに設定します。これは SQL データベース モードで動作し、組み込みのデータベースやキャッシュモードでは 使用できません。
- UDM_PARAM_VARDIR - 組み込みのデータベースやキャッシュモードを 使用する際に、インデクサがデータを保存するディレクトリへの パスを指定します。デフォルトでは、mnoGoSearch をインストールした ディレクトリの下の /var ディレクトリが 使用されます。文字列値のみを指定可能です。

val

変更履歴

バージョン	説明
4.1.0	**UDM_PARAM_VARDIR** が追加されました。

返り値

成功した場合に ***TRUE*** を、失敗した場合に ***FALSE*** を返します。

注意

注意: クロスワードは mnoGoSearch 3.1.11 以降でのみサポートされています。

目次

Microsoft SQL Server 関数

導入

以下の関数によりMS SQL Server データベースにアクセス可能となります。

要件

Win32 プラットフォームでの要件

この拡張モジュールは、PHP がインストールされているシステムに MS SQL クライアントツールのインストールを要します。クライアントツールは、MS SQL Server CD か、またはサーバの ¥winnt¥system32 から ntwdblib.dll を PHP システムの ¥winnt¥system32 にコピーすることによりインストール可能です。 ntwdblib.dll のコピーでは、名前付きパイプでのアクセスのみが可能となります。 クライアントの設定には全てのツールのインストールが必要です。

Unix/Linux プラットフォームでの要件

MSSQL 拡張モジュールを Unix/Linux で使用するには、まず最初に FreeTDS ライブラリをビルドしてインストールする必要があります。ソースコードや インストール手順は FreeTDS のホームページ » http://www.freetds.org/ にあります。

> **注意**: Windows では、Microsoft の DBLIB が使用されます。カラム名を返す関数は、 DBLIB の dbcolname() 関数を使用しています。 DBLIB は SQL Server 6.x 用に開発されており、識別子の最大長は 30 です。 そのため、カラム名は最大 30 文字までとなります。FreeTDS を使用する プラットフォーム (Linux) ではこの問題は発生しません。

インストール手順

MSSQL 拡張モジュールは、extension=php_mssql.dll を php.ini に追加する ことにより、利用可能となります。

これらの関数を使用するには、 --with-mssql[=DIR] を指定して PHP を コンパイルする必要があります。DIR は FreeTDS のインストール先です。 また、FreeTDS は --enable-msdblib を指定してコンパイルする必要があります。

実行時設定

php.ini の設定により動作が変化します。

MS SQL Server 設定オプション

名前	デフォルト	変更の可否	変更履歴
mssql.allow_persistent	"1"	PHP_INI_SYSTEM	
mssql.max_persistent	"-1"	PHP_INI_SYSTEM	
mssql.max_links	"-1"	PHP_INI_SYSTEM	
mssql.min_error_severity	"10"	PHP_INI_ALL	
mssql.min_message_severity	"10"	PHP_INI_ALL	
mssql.compatability_mode	"0"	PHP_INI_ALL	
mssql.connect_timeout	"5"	PHP_INI_ALL	
mssql.timeout	"60"	PHP_INI_ALL	PHP 4.1.0 以降で使用可能です。
mssql.textsize	"-1"	PHP_INI_ALL	
mssql.textlimit	"-1"	PHP_INI_ALL	
mssql.batchsize	"0"	PHP_INI_ALL	PHP 4.0.4 以降で使用可能です。
mssql.datetimeconvert	"1"	PHP_INI_ALL	PHP 4.2.0 以降で使用可能です。
mssql.secure_connection	"0"	PHP_INI_SYSTEM	PHP 4.3.0 以降で使用可能です。
mssql.max_procs	"-1"	PHP_INI_ALL	PHP 4.3.0 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

MSSQL_ASSOC (integer)
MSSQL_NUM (integer)
MSSQL_BOTH (integer)
SQLTEXT (integer)
SQLVARCHAR (integer)
SQLCHAR (integer)
SQLINT1 (integer)
SQLINT2 (integer)
SQLINT4 (integer)
SQLBIT (integer)
SQLFLT8 (integer)

---

mssql_bind

(PHP 4 >= 4.0.7, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_bind — ストアドプロシージャまたはリモートストアドプロシージャへパラメータを追加する

説明

bool **mssql_bind** (resource $stmt , string $param_name , mixed &$var , int $type [, int $is_output [, int $is_null [, int $maxlen]]])

ストアドプロシージャまたはリモートストアドプロシージャへパラメータをバインドします。

パラメータ

stmt

mssql_init() で取得したステートメントリソース。

param_name

パラメータ名を表す文字列。

注意: T-SQL 構文のように @ 文字を含めなければなりません。 mssql_execute() にある例を参照ください。

var

MSSQL パラメータとしてバインドする PHP 変数。 値渡し、参照渡しのどちらも可能です。実行後に OUTPUT や RETVAL を取得するには参照渡しとします。

type

SQLTEXT, **SQLVARCHAR**, **SQLCHAR**, **SQLINT1**, **SQLINT2**, **SQLINT4**, **SQLBIT**, **SQLFLT4**, **SQLFLT8**, **SQLFLTN** のいずれか。

is_output

値が OUTPUT パラメータであるかどうか。OUTPUT パラメータであることに気づかなかった場合、それは通常の INPUT パラメータとして扱われ、エラーは発生しません。

is_null

パラメータが **NULL** かどうか。**NULL** 値を var に渡しても正しく動作しません。

maxlen

char/varchar 値で使用します。データの長さを指定する必要があります。 たとえばパラメータが varchar(50) の場合、型は **SQLVARCHAR** で、この値が 50 となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 mssql_bind() の例

```php
<?php

$cn = mssql_connect($DBSERVER, $DBUSER, $DBPASS);
mssql_select_db($DB, $cn);

$sp = mssql_init("WDumpAdd"); // ストアドプロシージャ名

mssql_bind($sp, "@productname", stripslashes($newproduct), SQLVARCHAR, false, false, 150);
mssql_bind($sp, "@quantity", stripslashes($newquantity), SQLVARCHAR, false, false, 50);

mssql_execute($sp);
mssql_close($cn);

?>
```

参考

- mssql_execute()
- mssql_free_statement()
- mssql_init()

---

mssql_close

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_close — MS SQL Server への接続を閉じる

説明

bool **mssql_close** ([resource $link_identifier])

指定したリンク ID が指す MS SQL Server データベースへのリンクを閉じます。 リンク ID が指定されない場合、最後にオープンされたリンクが指定されたと仮定します。

持続的でないリンクはスクリプト実行終了時に自動的に閉じられるので この関数は通常は必要ではありません。

パラメータ

link_identifier

mssql_connect() が返す MS SQL リンク ID。

この関数は、 *mssql_pconnect()* が作成した持続的リンクを閉じません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *mssql_connect()*
- *mssql_pconnect()*

---

mssql_connect

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_connect — MS SQL サーバ接続をオープンする

説明

resource **mssql_connect** ([string $servername [, string $username [, string $password [, bool $new_link]]]])

mssql_connect() はMS SQL サーバへの接続を確立します。 引数 servername は 'interfaces' ファイルに定義された有効なサーバ名です。

サーバへのリンクは、事前に *mssql_close()* により 明示的に閉じられていない限り、スクリプト終了後すぐに閉じられます。

パラメータ

servername

MS SQL サーバ。 hostname,port のようにポート番号を含めることもできます。

username

ユーザ名。

password

パスワード。

new_link

同じ引数で **mssql_connect()** が再度コールされた場合、 新規のリンクは作成されず、代わりに既にオープンされたリンク ID が返されます。 このパラメータは、この振る舞いを変更し、**mssql_connect()** が常に新しいリンクを返すようにします。同じパラメータで事前に **mssql_connect()** がコールされていたとしても、 新しいリンクを返します。

返り値

成功時に MS SQL リンク ID、エラー発生時に **FALSE** を返します。

変更履歴

バージョン	説明
4.4.1 および 5.1.0	パラメータ *new_link* が追加されました。

参考

- *mssql_close()*
- *mssql_pconnect()*

---

mssql_data_seek

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_data_seek — 内部行ポインタを移動する

説明

bool **mssql_data_seek** (resource $result_identifier , int $row_number)

mssql_data_seek() は、指定した結果 ID が指す MS SQL 結果に関する内部行ポインタを指定した行番号に移動します。 最初の行は行番号 0 となります。 この後、*mssql_fetch_row()* をコールした場合、 その行を返します。

パラメータ

result_identifier

処理対象となる結果リソース。

row_number

新しい結果ポインタの行番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

mssql_execute

(PHP 4 >= 4.0.7, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_execute — MS SQL サーバデータベースでストアドプロシージャを実行する

説明

mixed **mssql_execute** (resource $stmt [, bool $skip_results])

MS SQL サーバデータベースでストアドプロシージャを実行します。

パラメータ

stmt

mssql_init() で取得したステートメントハンドル.

skip_results

注意

注意: ストアドプロシージャが返すパラメータや返り値は、ストアドプロシージャが 複数の結果セットを返す場合を除いて **mssql_execute()** のコール後に使用可能となります。複数の結果セットを返す場合は mssql_next_result() を使用して結果に移動します。 最後の結果セットが処理されてはじめてパラメータや返り値が使用可能と なります。

参考

- mssql_bind()
- mssql_free_statement()
- mssql_init()

---

mssql_fetch_array

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_fetch_array — 連想配列・数値添字配列・あるいはその両方で結果の行を取得する

説明

array **mssql_fetch_array** (resource $result [, int $result_type])

mssql_fetch_array() は、 mssql_fetch_row() の拡張版です。 この関数は、結果の配列に数値インデックスにデータを保持するのに加えて、フィールド名をキーとしてデータを連想配列にも保存します。

機能がかなり増えているにもかかわらず、**mssql_fetch_array()** は mssql_fetch_row() よりもそれほど遅くはないということを強調しておきます。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

result_type

取得する配列の形式。定数で、以下のいずれかの値となります。 **MSSQL_ASSOC**、 **MSSQL_NUM** あるいは **MSSQL_BOTH**（デフォルト）。

返り値

取得された行に対応する配列、行がもうない場合に **FALSE** を返します。

注意

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- mssql_fetch_row()

---

mssql_fetch_assoc

(PHP 4 >= 4.2.0, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_fetch_assoc — 結果の現在行を連想配列として返す

説明

array **mssql_fetch_assoc** (resource $result_id)

取得した行に対応する連想配列を返し、内部データポインタをひとつ進めます。 **mssql_fetch_assoc()** は、 mssql_fetch_array() の二番目のオプションパラメータに **MSSQL_ASSOC** を指定してコールするのと同等です。

パラメータ

result_id

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

取得した行に対応する連想配列を返します。もう行がない場合には **FALSE** を返します。

---

mssql_fetch_batch

(PHP 4 >= 4.0.4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_fetch_batch — レコードの次のバッチを返す

説明

int **mssql_fetch_batch** (resource $result)

レコードの次のバッチを返します。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

バッチ番号を整数値で返します。

例

Example#1 mssql_fetch_batch() の例

```
<?php
$resDb = mssql_connect('localhost', 'user', 'name');
$result = mssql_query('SELECT * FROM MYTABLE', $resDb, 10000);

$intNumRows = mssql_num_rows($result);

while ($intNumRows > 0) {
     while ($arrRow = mssql_fetch_assoc($result)) {
        // 何らかの処理 ...
     }
     $intNumRows = mssql_fetch_batch($result);
}
?>
```

---

mssql_fetch_field

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_fetch_field — フィールド情報を取得する

説明

object **mssql_fetch_field** (resource $result [, int $field_offset])

mssql_fetch_field() は、あるクエリー結果のフィールドに関して情報を得るために使用します。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

field_offset

フィールドオフセット。フィールドオフセットが指定されない場合、 **mssql_fetch_field()** によりまだ取得されていない次のフィールドが取得されます。 field_offset は 0 から始まります。

返り値

フィールド情報を含むオブジェクトを返します。

オブジェクトのプロパティは次のようになります。

- name - カラム名。カラムが関数の結果である場合、このプロパティは computed#N に設定されます。ただし、#N はシリアル番号です。
- column_source - そのカラムが取得されたテーブル
- max_length - カラムの最大長
- numeric - そのカラムが数字である場合に 1
- type - カラムの型

参考

- mssql_field_seek()

---

mssql_fetch_object

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_fetch_object — オブジェクトとして行を取得する

説明

object **mssql_fetch_object** (resource $result)

mssql_fetch_object() は mssql_fetch_array() に似ていますが、配列の代わりに オブジェクトが返されるという違いがあります。 間接的にこのことは、データをフィールド名でのみアクセスすることが 可能であり、そのオフセットではアクセスできないことを意味します（番号はプロパティ名としては使用できません）。

速度面でこの関数は mssql_fetch_array() と同等であり、 mssql_fetch_row() とほとんど同じです（違いは僅かです）。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

取得された行に対応するプロパティを有するオブジェクト、 またはもう行がない場合に **FALSE** を返します。

注意

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- mssql_fetch_array()
- mssql_fetch_row()

---

mssql_fetch_row

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_fetch_row — 配列として行を取得する

説明

array **mssql_fetch_row** (resource $result)

mssql_fetch_row() は指定した結果 ID に関連する結果から 1 行分のデータを取得します。行は配列として返されます。 配列オフセットに保存された各結果カラムは、オフセット 0 から始まります。

mssql_fetch_rows() を続けてコールした場合、 結果セットの次の行が返され、行がもうない場合は **FALSE** が返されます。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

取得された行に対応する配列、または行がもうない場合に **FALSE** を返します。

注意

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- mssql_fetch_array()
- mssql_fetch_object()
- mssql_data_seek()
- mssql_result()

---

mssql_field_length

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_field_length — フィールド長を得る

説明

int **mssql_field_length** (resource $result [, int $offset])

結果 result のフィールド番号 offset のフィールド長を返します。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

offset

フィールドオフセット。0 から始まります。 省略した場合は現在のフィールドを使用します。

返り値

成功した場合は指定したフィールドの長さ、 失敗した場合は **FALSE** を返します。

注意

注意: **Win32 ユーザへの注意** PHP が使用している API (MS DbLib C API) の制限により、VARCHAR フィールドの長さは 255 までに限定されます。 それ以上のデータを保存したい場合は、かわりに TEXT フィールドを 使用します。

参考

- mssql_field_name()
- mssql_field_type()

---

mssql_field_name

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_field_name — フィールド名を得る

説明

string **mssql_field_name** (resource $result [, int $offset])

result のフィールド番号 offset の名前を返します。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

offset

フィールドオフセット。0 から始まります。省略した場合は現在のフィールドを使用します。

返り値

成功した場合は指定したフィールドインデックスの名前、 失敗した場合は **FALSE** を返します。

参考

- mssql_field_length()
- mssql_field_type()

---

mssql_field_seek

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_field_seek — 指定したフィールドオフセットに移動する

説明

bool **mssql_field_seek** (resource $result , int $field_offset)

指定したフィールドオフセットを探します。この後、フィールドオフセット を指定せずに mssql_fetch_field() をコールした 場合、このフィールドが返されます。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

field_offset

フィールドオフセット。0 から始まります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mssql_fetch_field()

---

mssql_field_type

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_field_type — フィールド型を得る

説明

string **mssql_field_type** (resource $result [, int $offset])

result のフィールド番号 offset の型を返します。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

offset

フィールドオフセット。0 から始まります。省略した場合は現在のフィールドを使用します。

返り値

成功した場合は指定したフィールドインデックスの型、 失敗した場合は **FALSE** を返します。

参考

- mssql_field_length()
- mssql_field_name()

---

mssql_free_result

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_free_result — 結果保持用メモリを解放する

説明

bool **mssql_free_result** (resource $result)

スクリプト実行時のメモリ使用量が過大であると懸念される場合のみ **mssql_free_result()** はコールするべきです。 全ての結果保持用メモリはスクリプト実行終了時に自動的に解放されます。 引数に結果 ID を指定して **mssql_free_result()** を コールすることが可能です。 この場合、関連する結果保持用メモリが解放されます。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

mssql_free_statement

(PHP 4 >= 4.3.2, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_free_statement — ステートメントのメモリを開放する

説明

bool **mssql_free_statement** (resource $stmt)

mssql_free_statement() をコールする必要があるのは、 スクリプトの実行中に大量のメモリを使用することが気になる場合のみです。 すべてのステートメントメモリはスクリプトが終了する際に自動的に開放 されます。ステートメント ID を引数に指定して **mssql_free_statement()** をコールすることで、 関連付けられたステートメントのメモリが開放されます。

パラメータ

stmt

mssql_init() で取得したステートメントリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mssql_bind()
- mssql_execute()
- mssql_init()

---

mssql_get_last_message

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_get_last_message — サーバの直近のメッセージを返す

説明

string **mssql_get_last_message** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mssql_guid_string

(PHP 4 >= 4.0.7, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_guid_string — 16 バイトバイナリ GUID を文字列に変換する

説明

string **mssql_guid_string** (string $binary [, int $short_format])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mssql_init

(PHP 4 >= 4.0.7, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_init — ストアドプロシージャまたはリモートのストアドプロシージャを初期化する

説明

resource **mssql_init** (string $sp_name [, resource $link_identifier])

ストアドプロシージャまたはリモートのストアドプロシージャを初期化します。

パラメータ

sp_name

ストアドプロシージャ名。たとえば ownew.sp_name や otherdb.owner.sp_name のようになります。

link_identifier

mssql_connect() が返す MS SQL リンク ID。

返り値

"statement" リソースの ID を返します。これを使用して、 mssql_bind() や mssql_execute() をコールします。エラー時には FALSE を返します。

参考

- mssql_bind()
- mssql_execute()
- mssql_free_statement()

---

mssql_min_error_severity

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_min_error_severity — エラー判定基準を小さく設定する

説明

void **mssql_min_error_severity** (int $severity)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

severity

返り値

値を返しません。

---

mssql_min_message_severity

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_min_message_severity — メッセージ判定基準を小さく設定する

説明

void **mssql_min_message_severity** (int $severity)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

severity

返り値

値を返しません。

---

mssql_next_result

(PHP 4 >= 4.0.5, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_next_result — 次の結果に内部結果ポインタを移動する

説明

bool **mssql_next_result** (resource $result_id)

サーバへ複数の SQL 命令を送信するか複数の結果を有するストアドプロシージャを 実行する場合、この関数はサーバが複数の結果集合を返すようにします。 この関数は、サーバから追加の結果が存在するかどうかを調べます。 追加の結果集合が存在する場合、既存の結果集合を解放し、新しい 結果集合から行を取得するための準備を行います。

パラメータ

result_id

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

追加の結果集合が取得可能な場合に TRUE、 その他の場合に FALSE を返します。

例

Example#1 mssql_next_result() の例

```
<?php
$link = mssql_connect("localhost", "userid", "secret");
mssql_select_db("MyDB", $link);
$sql = "Select * from table1 select * from table2";
$rs = mssql_query($sql, $link);
do {
    while ($row = mssql_fetch_row($rs)) {
    }
} while (mssql_next_result($rs));
mssql_free_result($rs);
mssql_close($link);
?>
```

---

mssql_num_fields

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_num_fields — 結果のフィールド数を得る

説明

int **mssql_num_fields** (resource $result)

mssql_num_fields() は結果のフィールド数を返します。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

フィールドの数を整数値で返します。

参考

- mssql_query()
- mssql_fetch_field()
- mssql_num_rows()

---

mssql_num_rows

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_num_rows — 結果の行数を得る

説明

int **mssql_num_rows** (resource $result)

mssql_num_rows() は結果の行数を返します。

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

返り値

行の数を整数値で返します。

参考

- mssql_query()
- mssql_fetch_row()

---

mssql_pconnect

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_pconnect — 持続的 MS SQL 接続をオープンする

説明

resource **mssql_pconnect** ([string $servername [, string $username [, string $password [, bool $new_link]]]])

mssql_pconnect() は mssql_connect() とほとんど同じく動作しますが、違う点が 2 つあります。

第 1 に、この関数は接続時に同じホスト、ユーザ名、パスワードで既に オープンされている(持続的)リンクを探そうとします。 もし見つかった場合、新規の接続をオープンせずにその ID を返します。

第 2 に、SQL サーバはスクリプトの実行終了時に接続を閉じません。 代わりに、リンクは後に使用されるためにオープンされたままとします (mssql_close() は **mssql_pconnect()** により確立されたリンクを閉じません)。

この型のリンクはこのため 'persistent' と呼ばれます。

パラメータ

servername

MS SQL サーバ。 hostname:port のようにポート番号を含めることもできます。

username

ユーザ名。

password

パスワード。

new_link

同じ引数で **mssql_pconnect()** が再度コールされた場合、 新規のリンクは作成されず、代わりに既にオープンされたリンク ID が返されます。 このパラメータは、この振る舞いを変更し、**mssql_pconnect()** が常に新しいリンクを返すようにします。同じパラメータで事前に **mssql_pconnect()** がコールされていたとしても、 新しいリンクを返します。

返り値

成功時に正の MS SQL 持続的リンク ID、エラー時に **FALSE** を返します。

---

mssql_query

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_query — MS SQL クエリを送る

説明

mixed **mssql_query** (string $query [, resource $link_identifier [, int $batch_size]])

mssql_query() は指定したリンク ID が指すサーバ上の現在アクティブなデータベースにクエリを送信します。

パラメータ

query

SQL クエリ。

link_identifier

mssql_connect() あるいは mssql_pconnect() が返す MS SQL リンク ID。

リンク ID が指定されない場合、最後にオープンされたリンクが 仮定されます。リンクがオープンされない場合、この関数は mssql_connect() がコールされた時と同様に リンクの確立を試み、これを使用します。

batch_size

バッファにまとめるレコードの数。

返り値

成功時に MS SQL 結果 ID、結果が返されなかった場合に **TRUE**、エラー時に **FALSE** を返します。

注意

> **注意**: クエリが複数の結果を返す場合は、 mssql_next_result() ですべての結果を取得するか、 次のクエリを実行する前に mssql_free_result() で結果を解放する必要があります。

参考

- mssql_select_db()
- mssql_connect()

---

mssql_result

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_result — 結果データを得る

説明

string ***mssql_result*** *(resource $result , int $row , mixed $field)*

mssql_result()*は、MS SQL 結果行からセルの内容を返します。*

パラメータ

result

処理対象となる結果リソース。これは mssql_query() のコールによって取得します。

row

行番号。

field

フィールドオフセット、フィールド名または テーブル名.フィールド名の形式で指定することができます。 カラム名がエイリアス定義されている場合 ('select foo as bar from...')、そのカラム名の代わりにエイリアスが使用されます。

> ***注意****: field 引数でオフセット番号を指定する方が、フィールド名または テーブル名.フィールド名で引数で指定するよりもかなり高速です。*

返り値

指定したセルの内容を返します。

注意

> ***注意****: 大量の結果を処理する場合、(以下に示す)行全体を取得する関数のどれかを 使用することを考える必要があります。これらの関数は一回の関数 コールで複数のセルの内容を返すので、* ***mssql_result()*** *よりもかなり高速です。*

参考

推奨される高性能な代替関数:

- *mssql_fetch_row()*
- *mssql_fetch_array()*
- *mssql_fetch_object()*

---

mssql_rows_affected

(PHP 4 >= 4.0.4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_rows_affected — クエリにより変更されたレコード数を返す

説明

int ***mssql_rows_affected*** *(resource $link_identifier)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

link_identifier

mssql_connect() あるいは mssql_pconnect() が返す MS SQL リンク ID。

---

mssql_select_db

(PHP 4, PHP 5, PECL odbtp:1.1.1-1.1.4)

mssql_select_db — MS SQL データベースを選択する

説明

bool ***mssql_select_db*** *(string $database_name [, resource $link_identifier])*

mssql_select_db() *は、指定したリンク ID が指すサーバの 現在アクティブなデータベースを設定します。*

以降の mssql_query() のコールは アクティブなデータベースに対して行われます。

パラメータ

database_name

The database name.

スペースやハイフン ("-") 、あるいはその他の例外文字を含むデータベース名を エスケープするには、下の例で示すようにデータベース名をブラケットで囲む 必要があります。この手法は、データベースの名前に予約語 (たとえば primary) が含まれている場合にも同様に使用する必要があります。

link_identifier

mssql_connect() あるいは mssql_pconnect() が返す MS SQL リンク ID。

リンク ID が指定されない場合、最後にオープンされたリンクが仮定されます。 リンクがオープンされない場合、関数は mssql_connect() がコールされた場合と同様に リンクを確立し、これを使用しようとします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 mssql_select_db() の例

```
<?php
$conn = mssql_connect('MYSQLSERVER', 'sa', 'password');
mssql_select_db('[my data-base]', $conn);
?>
```

参考

- mssql_connect()
- mssql_pconnect()
- mssql_query()

目次

Microsoft SQL Server および Sybase 関数 (PDO_DBLIB)

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

PDO_DBLIB は PHP Data Objects (PDO) インターフェイスを実装したドライバで、 PHP から Microsoft SQL Server および Sybase データベースに対して FreeTDS ライブラリを使用したアクセスが可能となります。

Windows では、Microsoft SQL Server や Sybase データベースにアクセスする際には PDO_ODBC ドライバを使用すべきです。 Windows 版のネイティブの DB-LIB は時代遅れになっており、 スレッドセーフではない上に Microsoft にもサポートされていません。

PDO_DBLIB DSN

(No version information available, might be only in CVS)

PDO_DBLIB DSN — Microsoft SQL Server および Sybase データベースに接続する

説明

PDO_DBLIB データソース名 (DSN) は以下の要素で構成されます。

DSN 接頭辞

PDO_DBLIB が FreeTDS ライブラリに対してリンクされている場合は DSN 接頭辞は **sybase:** です。 Microsoft SQL Server ライブラリに対してリンクされている場合は **mssql:** 、それ以外の DB-lib に対してリンクされている場合は **dblib:** となります。

host

データベースサーバが存在するホスト名を指定します。

dbname

データベース名を指定します。

例

Example#1 PDO_DBLIB DSN の例

以下の例は、Microsoft SQL Server および Sybase データベースに接続するための PDO_DBLIB DSN を表します。

```
mssql:host=localhost;dbname=testdb
sybase:host=localhost;dbname=testdb
dblib:host=localhost;dbname=testdb
```

Mohawk Software セッションハンドラ関数

導入

msession は、ローカルまたはリモートで実行される高速セッションデーモンへの インターフェイスです。複数の PHP Web サーバにおいて集中的なセッション管理を行うために設計されています。 msession とセッションサーバソフトウエアに関するより詳細な情報は、 » http://www.mohawksoft.org/?q=node/8 にあります。

> **注意**: この拡張モジュールは Windows 環境では利用できません。
>
> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.3.

要件

インストール手順

PHP で Msession サポートを有効にするには、 --with-msession[=DIR] を configure に 指定してください。ただし、DIR は Msession のインストールディレクトリです。

実行時設定

リソース型

定義済み定数

---

msession_connect

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_connect — msession サーバに接続する

説明

bool **msession_connect** (string $host , string $port)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_count

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_count — セッション数を得る

説明

int ***msession_count*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_create

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_create — *セッションを作成する*

説明

bool ***msession_create*** *(string $session , string $classname , string $data)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_destroy

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_destroy — *セッションを破棄する*

説明

bool ***msession_destroy*** *(string $name)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_disconnect

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_disconnect — *msession サーバへの接続を閉じる*

説明

void ***msession_disconnect*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_find

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_find — *名前と値で、すべてのセッションを検索する*

説明

array ***msession_find*** *(string $name , string $value)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_get_array

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_get_array — *msession 変数の配列を得る*

説明

array ***msession_get_array*** *(string $session)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_get_data

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_get_data — データセッションの、構造化されていないデータを取得する

説明

string **msession_get_data** (string $session)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_get

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_get — セッションから値を取得する

説明

string **msession_get** (string $session , string $name , string $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_inc

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_inc — セッションの値を増加させる

説明

string **msession_inc** (string $session , string $name)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_list

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_list — すべてのセッションの一覧を取得する

説明

array **msession_list** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_listvar

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_listvar — セッションの一覧を変数を使用して取得する

説明

array **msession_listvar** (string $name)

共通の属性を有するセッションを検索する際に使用します。

返り値

name という名前の変数を有する全ての セッションを値とする連想配列を返します。

---

msession_lock

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_lock — セッションをロックする

説明

int **msession_lock** (string $name)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_plugin

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_plugin — msession personality プラグイン内のエスケープ関数をコールする

説明

string **msession_plugin** (string $session , string $val [, string $param])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_randstr

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_randstr — ランダムな文字列を取得する

説明

string **msession_randstr** (int $param)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_set_array

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_set_array — セッションに配列の値を設定する

説明

void **msession_set_array** (string $session , array $tuples)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_set_data

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_set_data — データセッションの、構造化されていないデータを設定する

説明

bool **msession_set_data** (string $session , string $value)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_set

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_set — セッションに値を設定する

説明

bool **msession_set** (string $session , string $name , string $value)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_timeout

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_timeout — セッションの有効期間を設定/取得する

説明

int ***msession_timeout*** *(string $session [, int $param])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_uniq

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_uniq — ユニークな ID を取得する

説明

string ***msession_uniq*** *(int $param , string $classname , string $data)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

msession_unlock

(PHP 4 >= 4.2.0, PHP 5 <= 5.1.2)

msession_unlock — セッションのロックを解除する

説明

int ***msession_unlock*** *(string $session , int $key)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

---

mSQL 関数

導入

以下の関数により mSQL データベースサーバにアクセスすることが 可能になります。mSQL に関する詳細な情報は、 » http://www.hughes.com.au/ にあります。

インストール手順

以下の関数を利用可能とするには、 --with-msql[=dir] オプションにより mSQL サポートを追加して PHP をコンパイルする必要があります。DIR は mSQL のインストールディレクトリで、デフォルトの位置は /usr/local/msql3 です。

> ***注意**: **Win32 ユーザへの注意** この拡張モジュールを動作させるには、 Windows システムの PATH が通った場所に DLL ファイルが存在する必要があります。 FAQ の "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" で、その方法*

を説明しています。 DLL ファイルを PHP のフォルダから Windows のシステムディレクトリにコピーしても動作します（システムディレクトリは、デフォルトで PATH に含まれるからです）が、これは推奨しません。 この拡張モジュールを使用するには、以下のファイルが PATH の通った場所にある必要があります。 msql.dll

実行時設定

php.ini の設定により動作が変化します。

mSQL 設定オプション

名前	デフォルト	変更の可否	変更履歴
msql.allow_persistent	"1"	PHP_INI_ALL	
msql.max_persistent	"-1"	PHP_INI_ALL	
msql.max_links	"-1"	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

msql.allow_persistent boolean

持続的な mSQL 接続を許可するかどうか。

msql.max_persistent integer

プロセスごとの、持続的 mSQL 接続の最大数。

msql.max_links integer

プロセスごとの mSQL 接続の最大数。持続的接続の数も含めます。

リソース型

mSQL モジュールでは 2 種類のリソース型が使用されます。ひとつは データベース接続のリンク ID で、もうひとつはクエリ結果を保持する リソースです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

MSQL_ASSOC (integer)
MSQL_NUM (integer)
MSQL_BOTH (integer)

例

ここでは、mSQL データベースへの接続、クエリの実行、結果の表示 そして接続の切断を行う簡単な例を示します。

Example#1 mSQL の使用例

```
<?php
/* 接続し、データベースを選択する */
$link = msql_connect('localhost', 'username', 'password')
    or die('Could not connect : ' . msql_error($link));

msql_select_db('database', $link)
    or die('Could not select database');

/* SQL クエリを発行する */
$query = 'SELECT * FROM my_table';
$result = msql_query($query, $link) or die('Query failed : ' . msql_error());

/* 結果を HTML で表示する */
echo "<table>¥n";
while ($row = msql_fetch_array($result, MSQL_ASSOC)) {
    echo "¥t<tr>¥n";
    foreach ($row as $col_value) {
        echo "¥t¥t<td>$col_value</td>¥n";
    }
    echo "¥t</tr>¥n";
}
echo "</table>¥n";

/* 結果セットを開放する */
msql_free_result($result);

/* 接続をクローズする */
msql_close($link);
?>
```

---

msql_affected_rows

(PHP 4, PHP 5)

msql_affected_rows — 変更された行の数を返す

説明

int **msql_affected_rows** (resource $result)

result に関連する直近の SELECT、UPDATE あるいは DELETE クエリにより変更された行の数を返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

返り値

成功した場合には変更された行の数、エラー時に **FALSE** を返します。

参考

- msql_query()

---

msql_close

(PHP 4, PHP 5)

msql_close — mSQL 接続を閉じる

説明

bool **msql_close** ([resource $link_identifier])

msql_close() は、指定したリンク ID に関連付けられた mSQL サーバとの持続的でない接続を閉じます。

通常は **msql_close()** を使用する必要はありません。 なぜなら、持続的でないリンクはスクリプトの実行終了時に自動的に 閉じられるからです。 リソースの開放 も参照ください。

パラメータ

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- msql_connect()
- msql_pconnect()

---

msql_connect

(PHP 4, PHP 5)

msql_connect — mSQL 接続を開く

説明

resource **msql_connect** ([string $hostname])

msql_connect() は mSQL サーバとの接続を確立します。

同じ引数で **msql_connect()** が再度コールされた場合、 新しいリンクは作成されません。替わりに、既に開かれているリンクの ID が返されます。

msql_close() を明示的にコールしなければ、サーバとの リンクはスクリプトの実行終了直後に閉じられます。

パラメータ

hostname

ホスト名にはポート番号を含めることも可能です。例: hostname,port

指定されなかった場合、接続は Unix ドメインソケットで確立されます。 これは、ローカルホストへの TCP ソケット接続より効率的です。

> **注意**: この関数はホスト名とポートの区切りとしてコロン（:） も受け付けますが、カンマ（,）で区切るほうが 推奨されます。

返り値

成功した場合には正の mSQL リンク IDを、エラー時には **FALSE** を返します。

参考

- msql_pconnect()

- msql_close()

---

msql_create_db

(PHP 4, PHP 5)

msql_create_db — mSQL データベースを作成する

説明

bool **msql_create_db** (string $database_name [, resource $link_identifier])

msql_create_db() は、mSQL サーバ上に データベースの作成を試みます。

パラメータ

database_name

mSQL データベース名。

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- msql_drop_db()

---

msql_createdb

(PHP 4, PHP 5)

msql_createdb — msql_create_db() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_create_db()

---

msql_data_seek

(PHP 4, PHP 5)

msql_data_seek — 行に関する内部ポインタを移動する

説明

bool **msql_data_seek** (resource $result , int $row_number)

msql_data_seek() は、指定したクエリー ID (query_identfier)が指す mSQL の結果行への内部ポインタが指定した行番号 (row_number)を指すようにします。 次に msql_fetch_row() をコールした際には、 その行を返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

row_number

指定する行番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- msql_fetch_array()
- msql_fetch_object()
- msql_fetch_row()
- msql_result()

---

msql_db_query

(PHP 4, PHP 5)

msql_db_query — mSQL クエリを送信する

説明

resource **msql_db_query** (string $database , string $query [, resource $link_identifier])

msql_db_query() はデータベースを選択し、 そこに対してクエリを実行します。

パラメータ

database

mSQL データベースの名前。

query

SQL クエリ。

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

成功した場合には正の mSQL リンク IDを、エラー時には **FALSE** を返します。

参考

- msql_query()

---

msql_dbname

(PHP 4, PHP 5)

msql_dbname — msql_result() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_result()

---

msql_drop_db

(PHP 4, PHP 5)

msql_drop_db — mSQL データベースを破棄(削除)する

説明

bool **msql_drop_db** (string $database_name [, resource $link_identifier])

msql_drop_db() は、mSQL サーバから データベースを削除することを試みます。

パラメータ

database_name

データベースの名前。

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- msql_create_db()

---

msql_error

(PHP 4, PHP 5)

msql_error — 最後の msql コールに関するエラーメッセージを返す

説明

string **msql_error** (void)

msql_error() は、mSQL サーバで最後に発生したエラーを 返します。**msql_error()** でアクセス可能なのは 直近のエラーメッセージだけであることに注意しましょう。

返り値

直近のエラーメッセージを返します。エラーが発生しなかった場合は 空文字列を返します。

---

msql_fetch_array

(PHP 4, PHP 5)

msql_fetch_array — 結果の行を配列として取得する

説明

array **msql_fetch_array** (resource $result [, int $result_type])

msql_fetch_array() は、 msql_fetch_row() の拡張版です。 結果配列のデータを数値インデックスに格納することに加え、 フィールド名をキーとして連想配列にも格納します。

msql_fetch_array() の使用に際して注意すべきことは、 かなりの付加機能を提供するにもかかわらず、 msql_fetch_row() を使用する場合よりそんなに遅くないということです。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

result_type

以下の定数 **MSQL_ASSOC**、**MSQL_NUM**、および **MSQL_BOTH** のうちのいずれか。 **MSQL_BOTH** がデフォルト。

返り値

取得した行に対応する配列を返します。行が残っていない場合には **FALSE** を返します。

例

Example#1 msql_fetch_array() の例

```
<?php
$con = msql_connect();
if (!$con) {
    die('Server connection problem: ' . msql_error());
}

if (!msql_select_db('test', $con)) {
    die('Database connection problem: ' . msql_error());
}

$result = msql_query('SELECT id, name FROM people', $con);
if (!$result) {
    die('Query execution problem: ' . msql_error());
}

while ($row = msql_fetch_array($result, MSQL_ASSOC)) {
    echo $row['id'] . ': ' . $row['name'] . "¥n";
}

msql_free_result($result);
?>
```

変更履歴

バージョン	説明
4.3.11 および 5.0.4	NULL 値を含むカラムからデータを取得する際のバグが修正されました。 そのようなカラムは結果の配列に含まれていませんでした。

参考

- msql_fetch_row()
- msql_fetch_object()
- msql_data_seek()
- msql_result()

---

msql_fetch_field

(PHP 4, PHP 5)

msql_fetch_field — フィールド情報を得る

説明

object **msql_fetch_field** (resource $result [, int $field_offset])

msql_fetch_field() はあるクエリ結果に含まれる フィールドの情報を取得する際に使用することができます。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

field_offset

フィールドオフセット。指定されていない場合は、まだ **msql_fetch_field()** で取得されていない 次のフィールドが返されます。

返り値

フィールド情報を含むオブジェクトを返します。プロパティは以下のとおりです。

- name - カラム名
- table - カラムが属するテーブル名
- not_null - カラムが **NULL** になることができない場合に 1
- unique - カラムがユニークキーである場合に 1
- type - カラムの型

参考

- msql_field_seek()

---

msql_fetch_object

(PHP 4, PHP 5)

msql_fetch_object — 結果の行をオブジェクトとして取得する

説明

object **msql_fetch_object** (resource $result)

msql_fetch_object() は、 msql_fetch_array() に似ていますが、配列の代わりに オブジェクトを返すところだけが異なっています。遠まわしに言うと、 オブセット値によってではなくフィールド名によってのみデータを アクセスすることができることを意味しています （数字は、プロパティ名として使用できません）。

速度面でこの関数は msql_fetch_array() と同等です。そして、ほぼ msql_fetch_row() と同等の速度を有しています（その差は無視できるほどです）。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

返り値

得た行に対応するプロパティを有するオブジェクトを返します。 もう行がない場合には **FALSE** を返します。

例

Example#1 msql_fetch_object() の例

```
<?php
$con = msql_connect();
if (!$con) {
    die('Server connection problem: ' . msql_error());
}

if (!msql_select_db('test', $con)) {
    die('Database connection problem: ' . msql_error());
}

$result = msql_query('SELECT id, name FROM people', $con);
if (!$result) {
    die('Query execution problem: ' . msql_error());
}

while ($row = msql_fetch_object($result, MSQL_ASSOC)) {
    echo $row->id . ': ' . $row->name . "¥n";
}

msql_free_result($result);
```

```
?>
```

変更履歴

バージョン	説明
4.3.11 および 5.0.4	**NULL** 値を含むカラムからデータを取得する際のバグが修正されました。 そのようなカラムは結果の配列に含まれていませんでした。

参考

- msql_fetch_array()
- msql_fetch_row()
- msql_data_seek()
- msql_result()

---

msql_fetch_row

(PHP 4, PHP 5)

msql_fetch_row — 結果の行を数値配列として取得する

説明

array **msql_fetch_row** (resource $result)

msql_fetch_row() は、指定したリンク ID が指す 結果から 1 行分のデータを得ます。行は、配列として返されます。 各結果のカラムは、0 から始まる配列オフセットに保存されます。

連続して **msql_fetch_row()** をコールした場合、 次の行を結果として返します。もう行がない場合には **FALSE** を返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

返り値

結果の行に対応する配列を返します。もう行がない場合には **FALSE** を返します。

例

Example#1 msql_fetch_row() の例

```
<?php
$con = msql_connect();
if (!$con) {
    die('Server connection problem: ' . msql_error());
}

if (!msql_select_db('test', $con)) {
    die('Database connection problem: ' . msql_error());
}

$result = msql_query('SELECT id, name FROM people', $con);
if (!$result) {
    die('Query execution problem: ' . msql_error());
}

while ($row = msql_fetch_row($result)) {
    echo $row[0] . ': ' . $row[1] . "¥n";
}

msql_free_result($result);
?>
```

変更履歴

バージョン	説明
4.3.11 および 5.0.4	**NULL** 値を含むカラムからデータを取得する際のバグが修正されました。 そのようなカラムは結果の配列に含まれていませんでした。

参考

- msql_fetch_array()
- msql_fetch_object()
- msql_data_seek()
- msql_result()

---

msql_field_flags

(PHP 4, PHP 5)

msql_field_flags — フィールドのフラグを取得する

説明

string **msql_field_flags** (resource $result , int $field_offset)

msql_field_flags() は、指定したフィールドの フィールドフラグを返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は、1から始まります。

返り値

指定したキーのフィールドフラグを含む文字列を返します。 とりうる値は: primary key not null、 not null、primary key、 unique not null あるいは unique です。

---

msql_field_len

(PHP 4, PHP 5)

msql_field_len — フィールドの長さを取得する

説明

int **msql_field_len** (resource $result , int $field_offset)

msql_field_len() は指定したフィールドの長さを返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は、1から始まります。

返り値

指定したフィールドの長さを返します。エラー時には **FALSE** を返します。

---

msql_field_name

(PHP 4, PHP 5)

msql_field_name — 結果における指定したフィールドの名前を取得する

説明

string **msql_field_name** (resource $result , int $field_offset)

msql_field_name() は、指定したフィールドインデックスの 名前を取得します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は、1から始まります。

返り値

フィールドの名前を返します。失敗した場合は **FALSE** を返します。

参考

- msql_field_len()

---

msql_field_seek

(PHP 4, PHP 5)

msql_field_seek — フィールドオフセットを設定する

説明

bool **msql_field_seek** (resource $result , int $field_offset)

指定されたフィールドオフセットに移動します。 この後で msql_fetch_field() を フィールドオフセットを指定せずにコールした場合は、 ここで指定したフィールドが返されます。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は、1から始まります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- msql_fetch_field()

---

msql_field_table

(PHP 4, PHP 5)

msql_field_table — フィールドのテーブル名を取得する

説明

int **msql_field_table** (resource $result , int $field_offset)

指定したフィールドが含まれるテーブルの名前を返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は、1から始まります。

返り値

成功した場合はテーブルの名前、失敗した場合は **FALSE** を返します。

---

msql_field_type

(PHP 4, PHP 5)

msql_field_type — フィールドの型を取得する

説明

string **msql_field_type** (resource $result , int $field_offset)

msql_field_type() は、指定したフィールドインデックスの 型を取得します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は、1から始まります。

返り値

フィールドの型。int、 char、real、ident、 null あるいは unknown のいずれか。 失敗した場合には **FALSE** を返します。

msql_fieldflags

(PHP 4, PHP 5)

msql_fieldflags — msql_field_flags() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_field_flags()

msql_fieldlen

(PHP 4, PHP 5)

msql_fieldlen — msql_field_len() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_field_len()

msql_fieldname

(PHP 4, PHP 5)

msql_fieldname — msql_field_name() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_field_name()

msql_fieldtable

(PHP 4, PHP 5)

msql_fieldtable — msql_field_table() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_field_table()

msql_fieldtype

(PHP 4, PHP 5)

msql_fieldtype — msql_field_type() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_field_type()

msql_free_result

(PHP 4, PHP 5)

msql_free_result — 結果保持用メモリを開放する

説明

bool **msql_free_result** (resource $result)

msql_free_result() は、 query_identifier が指すメモリを開放します。 リクエストの処理を完了した時、このメモリは自動的に開放されます。 よって、この関数を使用する必要があるのは、 スクリプトの実行時に大量のメモリを使用しないことを 確実にしたい場合のみです。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

msql_list_dbs

(PHP 4, PHP 5)

msql_list_dbs — サーバー上の mSQL データベースのリストを返す

説明

resource **msql_list_dbs** ([resource $link_identifier])

msql_list_dbs() は、指定した link_identifier 上で使用可能なデータベースの 一覧を返します。

パラメータ

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

結果セットを返します。これは msql_fetch_array() のような結果セットを読み込む関数によって処理が可能です。失敗した場合は、 この関数は **FALSE** を返します。

参考

- msql_list_tables()
- msql_list_fields()

---

msql_list_fields

(PHP 4, PHP 5)

msql_list_fields — 結果フィールドのリストを得る

説明

resource **msql_list_fields** (string $database , string $tablename [, resource $link_identifier])

msql_list_fields() は、指定したテーブルに関する情報を返します。

パラメータ

database

データベースの名前。

tablename

テーブルの名前。

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

結果セットを返します。これは msql_fetch_array() のような結果セットを読み込む関数によって処理が可能です。失敗した場合は、 この関数は **FALSE** を返します。

参考

- msql_list_tables()
- msql_list_dbs()

---

msql_list_tables

(PHP 4, PHP 5)

msql_list_tables — mSQL データベースにおけるテーブルのリストを得ます

説明

resource **msql_list_tables** (string $database [, resource $link_identifier])

msql_list_tables() は、指定した database 上のテーブルのリストを返します。

パラメータ

database

データベースの名前。

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

結果セットを返します。これは msql_fetch_array() のような結果セットを読み込む関数によって処理が可能です。失敗した場合は、 この関数は **FALSE** を返します。

参考

- msql_list_fields()
- msql_list_dbs()

---

msql_num_fields

(PHP 4, PHP 5)

msql_num_fields — 結果におけるフィールドの数を得る

説明

int **msql_num_fields** (resource $result)

msql_num_fields()は、結果セットにおける フィールドの数を返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

返り値

結果セットにおけるフィールドの数を返します。

参考

- msql_query()
- msql_fetch_field()
- msql_num_rows()

---

msql_num_rows

(PHP 4, PHP 5)

msql_num_rows — 結果における行の数を得る

説明

int **msql_num_rows** (resource $query_identifier)

msql_num_rows() は、結果セットにおける 行の数を返します。

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

返り値

結果セットにおける行の数を返します。

参考

- msql_num_fields()

---

msql_numfields

(PHP 4, PHP 5)

msql_numfields — msql_num_fields() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_num_fields()

---

msql_numrows

(PHP 4, PHP 5)

msql_numrows — msql_num_rows() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_num_rows()

---

msql_pconnect

(PHP 4, PHP 5)

msql_pconnect — 持続的な mSQL 接続をオープンする

説明

resource **msql_pconnect** ([string $hostname])

msql_pconnect() は msql_connect() とほぼ同じ動作をしますが、 大きな違いがふたつあります。

ひとつめは、この関数は接続する際に同じホストにおいてすでに確立された （持続的な）リンクを探そうとすることです。そのようなリンクが見つかった 場合、新たな接続をオープンするかわりにそのリンクの ID が返されます。

ふたつめは、スクリプトの実行が終了しても SQL サーバへの接続が閉じられない ということです。そのかわりに、次に利用するときのために リンクが開かれたままとなります （msql_close() はこの関数によって確立されたリンクを閉じません）。

パラメータ

hostname

ホスト名にはポート番号を含めることも可能です。例: hostname,port

指定されなかった場合、接続は Unix ドメインソケットで確立されます。 これは、ローカルホストへの TCP ソケット接続より効率的です。

> **注意**: この関数はホスト名とポートの区切りとしてコロン（:） も受け付けますが、カンマ（,）で区切るほうが 推奨されます。

返り値

成功した場合には正の mSQL リンク IDを、エラー時には **FALSE** を返します。

参考

- msql_connect()
- msql_close()

---

msql_query

(PHP 4, PHP 5)

msql_query — mSQL クエリを送信する

説明

resource **msql_query** (string $query [, resource $link_identifier])

msql_query() は、指定されたリンク ID に 関連付けられたサーバ上の現在アクティブなデータベースに クエリを送信します。

パラメータ

query

SQL クエリ。

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

成功した場合には正の mSQL リンク IDを、エラー時には **FALSE** を返します。

例

Example#1 msql_query() の例

```
<?php
$link = msql_connect("dbserver")
    or die("unable to connect to msql server: " . msql_error());
msql_select_db("db", $link)
    or die("unable to select database 'db': " . msql_error());

$result = msql_query("SELECT * FROM table WHERE id=1", $link);
if (!$result) {
    die("query failed: " . msql_error());
}

while ($row = msql_fetch_array($result)) {
    echo $row["id"];
}
?>
```

参考

- msql_db_query()
- msql_select_db()
- msql_connect()

---

msql_regcase

(PHP 4, PHP 5)

msql_regcase — sql_regcase() のエイリアス

説明

この関数は次の関数のエイリアスです。 sql_regcase()

---

msql_result

(PHP 4, PHP 5)

msql_result — 結果のデータを得る

説明

string **msql_result** (resource $result , int $row [, mixed $field])

msql_result() は、mSQL 結果セットの セルの内容を返します。

大きな結果セットを処理する際には、(以下に指定した) 行全体を取り出す関数のどれかの使用を考慮するべきです。 これらの関数は 1 度のコールで複数セルの内容を返し、**msql_result()** よりもかなり高速に動作します。

推奨される方法は以下のとおりです。 msql_fetch_row()、 msql_fetch_array()、および msql_fetch_object()

パラメータ

result

評価された結果 リソース。この結果は、msql_query() のコールにより得られたものです。

row

行オフセット。

field

フィールドのオフセット、あるいはフィールド名、またはテーブル名 .フィールド名 のいずれか。カラム名がエイリアス定義されている ('select foo as bar from ...') 場合、カラム名のかわりに エイリアスを使用します。

> **注意**: フィールド名やテーブル名.フィールド名を指定するよりも フィールドのオフセットを数値で指定するほうがはるかに高速です。

返り値

指定した mSQL 結果セットから、行とオフセットに対応するセルの内容を返します。

---

msql_select_db

(PHP 4, PHP 5)

msql_select_db — mSQL データベースを選択する

説明

bool **msql_select_db** (string $database_name [, resource $link_identifier])

msql_select_db() は、指定した link_identifier に関連付けられたサーバ上の アクティブなデータベースを設定します。

これ以降の msql_query() のコールは、アクティブな データベースに対して行われます。

パラメータ

database_name

データベース名。

link_identifier

mSQL 接続。指定されない場合、msql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、この関数は、 msql_connect()をコールした時と同様にリンクを確立し、使用します。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- msql_connect()
- msql_pconnect()
- msql_query()

---

msql_tablename

(PHP 4, PHP 5)

msql_tablename — msql_result() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_result()

---

msql

(PHP 4, PHP 5)

msql — msql_db_query() のエイリアス

説明

この関数は次の関数のエイリアスです。 msql_db_query()

目次

- msql_free_result — 結果保持用メモリを開放する
- msql_list_dbs — サーバー上の mSQL データベースのリストを返す
- msql_list_fields — 結果フィールドのリストを得る
- msql_list_tables — mSQL データベースにおけるテーブルのリストを得ます
- msql_num_fields — 結果におけるフィールドの数を得る
- msql_num_rows — 結果における行の数を得る
- msql_numfields — msql_num_fields のエイリアス
- msql_numrows — msql_num_rows のエイリアス
- msql_pconnect — 持続的な mSQL 接続をオープンする
- msql_query — mSQL クエリを送信する
- msql_regcase — sql_regcase のエイリアス
- msql_result — 結果のデータを得る
- msql_select_db — mSQL データベースを選択する
- msql_tablename — msql_result のエイリアス
- msql — msql_db_query のエイリアス

マルチバイト文字列関数 (mbstring)

導入

全ての文字をシングルバイトで一対一表現可能な言語は数多くありますが、 文字表現に単一バイトによる表現範囲を越えるほど多くの文字を必要とする言語も多くあります (1 バイトは 8 ビットから構成されます。各ビットには、1 あるいは 0 の 2 種類の値しか保持できません。そのため、単一のバイトで表すことのできる値は 256 (2 の 8 乗) 種類までとなります)。 マルチバイト文字のエンコーディング法は、 256 を越える文字を通常のビット単位の符号化システムで表現するために開発されました。

マルチバイトエンコーディングで符号化された文字列を (trim, split, splice などで) 処理する際、こうしたエンコーディングでは二つ以上の連続するバイトが 一つの文字を表す可能性があるため、特別な関数を使用する必要があります。 マルチバイトに対応しない文字列関数を文字列に適用した場合、 マルチバイト文字の先頭バイトまたは終了バイトを検出できずに文字列を壊し、 多くの場合には元の意味を失わせてしまう可能性があります。

mbstring はマルチバイト対応の文字列関数を提供し、 PHP でマルチバイトエンコーディングを処理することを容易にします。 それに加えて、mbstring は、 可能な範囲での文字エンコーディングの変換を処理します。 mbstring は UTF-8 や UCS-2 のような Unicode に基づくエンコーディングや多くのシングルバイトエンコーディングを処理するのに便利です (対応するエンコーディングを以下に示します)。

PHP の文字エンコーディングに関する要件

以下の型のエンコーディングが、PHP で安全に使用することができます。

- シングルバイトエンコーディングで、
  - 00h から 7fh の範囲の文字に関して、 ASCII 互換 (ISO646 互換) のマッピングを有する。
- マルチバイトエンコーディングで、
  - 00h から 7fh の範囲の文字では、 ASCII 互換のマッピングを有する。
  - ISO2022 エスケープシーケンスを使用しない。
  - 単一の文字を表す複数バイトのいずれにおいても 00h から 7fh の値を使用しない。

PHP で動作しないと思われる文字エンコーディングの例を以下に示します。

JIS, SJIS, ISO-2022-JP, BIG-5

これらのエンコーディングで書かれた PHP スクリプトは、 特に符号化された文字列がスクリプトで記述子やリテラルに使用される場合には、 動作しない可能性がありますが、入力される HTTP クエリに関して mbstring の透過的なエンコーディングフィルタを 設定することでこれらのエンコーディングをほとんど使用しないようにすることができます。

> **注意**: SJIS, BIG5, CP936, CP949, GB18030 は、読者がパーサ/コンパイラ、 文字エンコーディングと文字エンコーディングの問題点について精通していない限り 内部エンコーディングとして使用するべきではありません。

> **注意**: PHP でデータベースに接続する場合、性能を向上させるためにデータベースと PHP の内部エンコーディングについて同じ文字エンコーディングを使用することを推奨します。
> PostgreSQL を使用している場合、 バックエンドの文字エンコーディングと異なる文字エンコーディングを使用することが可能です。
> 詳細については、PostgreSQL のマニュアルを参照ください。

インストール手順

mbstring は拡張モジュールです。つまり、デフォルトでは 有効にならないということです。 configure スクリプトでモジュールを有効にする必要が あります。詳細は、インストールの 節を参照してください。

mbstring モジュールに関係する設定オプションは 以下のとおりです。

- --enable-mbstring: mbstring 関数を有効にします。このオプションは、 mbstring 関数を利用するために必要です。

  mbstring 拡張モジュールを使用するためには、 libmbflライブラリが必要です。 libmbflは、mbstring 拡張モジュールにバンドルされています。システムにインストールされている libmbfl を利用する場合、 --with-libmbfl[=DIR] を指定します。

  PHP 4.3.0 以降、mbstring 拡張モジュールは 日本語のほかに中国語 (簡体字)・中国語 (繁体字)・ 韓国語・ロシア語をサポートするように機能拡張されました。

  PHP 4.3.4以前のバージョンの場合、 この機能を使用するには、--enable-mbstring=LANGの LANG パラメータに 以下のオプションのいずれかを追加する必要があります。 --enable-mbstring=cn を 使用した場合、簡体字中国語のエンコーディングがサポートされます。 --enable-mbstring=tw を 使用した場合、繁体字中国語のエンコーディングがサポートされます。 --enable-mbstring=kr を 使用した場

合、韓国語のエンコーディングがサポートされます。 --enable-mbstring=ru を 使用した場合、ロシア語のエンコーディングがサポートされます。そして、 --enable-mbstring=ja (デフォルト)を 使用した場合、日本語のエンコーディングがサポートされます。 --enable-mbstring=all を指定した場合、サポートされるすべての文字エンコーディングが有効となります。 以前のバージョンとの互換性のため、何もオプションを指定せずに --enable-mbstring を使用した場合は 日本語のエンコーディングがサポートされます。

注意: PHP 4.3.4以降では、--enable-mbstringにより、 libmbflでサポートされている全てのエンコーディングが有効となるよう になっています。

- --enable-mbstr-enc-trans : mbstring 変換エンジンを使用した、HTTP 入力の 文字エンコーディング変換を有効にします。この機能が有効の場合、 HTTP 入力文字エンコーディングは、自動的に mbstring.internal_encoding に変換されます。

  注意: PHP 4.3.0 以降、このオプション --enable-mbstr-enc-trans は廃止され、実行時の設定 mbstring.encoding_translation に変更となります。HTTP 入力文字エンコーディング変換は、 このオプションを On に設定した場合のみ 有効となります (デフォルトは Offです)。

- --disable-mbregex : マルチバイト対応の正規表現関数を無効にします。

実行時設定

php.ini の設定により動作が変化します。

mbstring 設定オプション

名前	デフォルト	変更可能な範囲	変更履歴
mbstring.language	"neutral"	PHP_INI_PERDIR	PHP 4.3.0 から利用可能です。
mbstring.detect_order	NULL	PHP_INI_ALL	PHP 4.0.6 から利用可能です。
mbstring.http_input	"pass"	PHP_INI_ALL	PHP 4.0.6 から利用可能です。
mbstring.http_output	"pass"	PHP_INI_ALL	PHP 4.0.6 から利用可能です。
mbstring.internal_encoding	NULL	PHP_INI_ALL	PHP 4.0.6 から利用可能です。
mbstring.script_encoding	NULL	PHP_INI_ALL	PHP 4.3.0 から利用可能です。
mbstring.substitute_character	NULL	PHP_INI_ALL	PHP 4.0.6 から利用可能です。
mbstring.func_overload	"0"	PHP_INI_PERDIR	PHP 4.2.0 から利用可能で、PHP <= 4.2.3 は PHP_INI_SYSTEM です。
mbstring.encoding_translation	"0"	PHP_INI_PERDIR	PHP 4.3.0 から利用可能です。
mbstring.strict_detection	"0"	PHP_INI_ALL	PHP 5.1.2 から利用可能です。

PHP_INI_* 定数の詳細及び定義については、ini_set() を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

mbstring.language string

mbstring で使用される言語設定(NLS)のデフォルト値。 この設定は mbstring.internal_encoding を定義するため、 php.ini の中で mbstring.internal_encoding は、 mbstring.language の後に置く必要があることに注意してください。

mbstring.encoding_translation boolean

入力される HTTP クエリに関して、 文字エンコーディング検出および内部文字エンコーディングへの変換を行う 透過的な文字エンコーディングフィルタを有効にします。

mbstring.internal_encoding string

内部文字エンコーディングのデフォルト値を定義します。

mbstring.http_input string

HTTP 入力文字エンコーディングのデフォルト値を定義します。

mbstring.http_output string

HTTP 出力文字エンコーディングのデフォルト値を定義します。

mbstring.detect_order string

文字コード検出のデフォルト値を定義します。 mb_detect_order()も参照ください。

mbstring.substitute_character string

無効な文字を代替する文字を定義します。

mbstring.func_overload string

シングルバイト対応の関数を mbstring 関数の対応する関数でオーバーロード (置換)します。詳細は、 関数のオーバーロード を参照してください。

mbstring.strict_detection boolean

厳密なエンコーディング検出を行います。

» HTML 4.01の規約 によると、Web ブラウザは、 フォームのデータを投稿する際にページで使用される文字エンコーディングと 異なるエンコーディングを使用することができます。 ブラウザで使用される文字エンコーディングを検出するには、 mb_http_input() を参照ください。

一般的に使用されるブラウザでは、 指定したHTML文書の文字エンコーディングをかなり正確に推定することができますが、 header() または設定パラメータ default_charset により、 Content-Type HTTP ヘッダで charset を設定する方がより良いでしょう。

Example#1 php.ini 設定の例

```
; デフォルトの言語を設定
mbstring.language          = neutral; デフォルト言語を中立(UTF-8)に設定 (デフォルト)
```

```
mbstring.language          = English; デフォルト言語を英語に設定
mbstring.language          = Japanese; デフォルト言語を日本語に設定

;; デフォルトの内部エンコーディングを設定
;; 注意: PHPで動作する文字エンコーディングを使用すること
mbstring.internal_encoding     = UTF-8   ; 内部エンコーディングを UTF-8 に設定

;; HTTP入力エンコーディング変換を有効にする
mbstring.encoding_translation  =  On

;; デフォルトのHTTP入力文字エンコーディングを設定
;; 注意: スクリプトではhttp_inputの設定は変更できません。
mbstring.http_input             = pass     ; 変換しない。
mbstring.http_input             = auto     ; HTTP 入力を auto に設定
                                           ; 「auto」は、「ASCII,JIS,UTF-8,
                                  ; EUC-JP,SJIS」に展開されます。
mbstring.http_input              = SJIS     ; HTTP入力をSJISに設定
mbstring.http_input              = UTF-8,SJIS,EUC-JP ; 順番を指定

;; デフォルトのHTTP出力文字エンコーディングを設定
mbstring.http_output            = pass     ; 変換しない
mbstring.http_output            = UTF-8    ; HTTP出力エンコーディングを
                                           ;  UTF-8 に指定

;; デフォルトの文字エンコーディング検出順序を設定
mbstring.detect_order           = auto     ; デフォルトの順番を auto に設定
mbstring.detect_order             = ASCII,JIS,UTF-8,SJIS,EUC-JP ; 順番を指定

;; 代替文字のデフォルト値を設定
mbstring.substitute_character  =  12307    ; Unicode 値を指定
mbstring.substitute_character  =  none     ; 文字を出力しない
mbstring.substitute_character  =  long     ; long の例: U+3000,JIS+7E7E
```

Example#2 EUC-JP ユーザ用の php.ini の設定

```
;; 出力バッファリングを無効にする
output_buffering        = Off

;; HTTP charsetヘッダを設定
default_charset         = EUC-JP

;; デフォルトの言語を日本語にする
mbstring.language  =  Japanese

;; HTTP 入力変換を有効にする
mbstring.encoding_translation  =  On

;; HTTP 入力エンコーディング変換を auto に設定
mbstring.http_input    = auto

;; HTTP 出力を EUC-JP に設定
mbstring.http_output   = EUC-JP

;; 内部エンコーディングを EUC-JP に設定
mbstring.internal_encoding  =  EUC-JP

;; 無効な文字を出力しない
mbstring.substitute_character  =  none
```

Example#3 SJIS ユーザ用の php.iniの 設定

```
;; 出力のバッファリングを有効に
output_buffering       = On

;; 出力の変換を有効にするために mb_output_handler を設定
output_handler          = mb_output_handler

;; HTTPヘッダ charset を設定
default_charset        = Shift_JIS

;; デフォルトの言語を日本語に設定
mbstring.language  =  Japanese

;; HTTP 入力変換を有効にする
mbstring.encoding_translation  =  On

;; HTTP 入力エンコーディング変換を auto に設定
mbstring.http_input   =  auto

;; SJIS に変換
mbstring.http_output  =  SJIS

;; 内部エンコーディングを EUC-JP に設定
mbstring.internal_encoding  =  EUC-JP

;; 無効な文字を出力しない
mbstring.substitute_character  =  none
```

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

MB_OVERLOAD_MAIL (integer)
MB_OVERLOAD_STRING (integer)
MB_OVERLOAD_REGEX (integer)
MB_CASE_UPPER (integer)
MB_CASE_LOWER (integer)
MB_CASE_TITLE (integer)

HTTP 入出力

HTTP 入出力の文字エンコーディング変換はバイナリデータも変換して しまいます。HTTP 入出力にバイナリデータが使用される場合、ユーザは、 文字エンコーディング変換を制御する必要があります。

> **注意**: PHP 4.3.2 およびそれ以前のバージョンの場合、 HTML フォームのenctype が multipart/form-data に設定された場合、mbstring は、POST データの文字エンコーディングを変換しません。 この場合、文字列を内部文字エンコーディングに変換してやる必要があります。
>
> **注意**: PHP 4.3.3 以降、HTML フォームの enctype が multipart/form-data に設定され、かつ、 php.ini において mbstring.encoding_translation に On が指定されている場合、 POST データの変数とアップロードされたファイルの名前の文字エンコーディングは、 内部文字エンコーディングに変換されます。 ただし、クエリーに関しては、変換されません。

- HTTP 入力

  PHP スクリプトで HTTP 入力文字変換を制御する手段はありません。 HTTP 入力文字変換を無効にするには、php.ini で行う必要があります。

  Example#4 php.ini で HTTP 入力変換を無効にする

  ```
  ;; HTTP 入力変換を無効にする
  mbstring.http_input = pass
  ;; HTTP 入力変換を無効にする (PHP 4.3.0 以降)
  mbstring.encoding_translation = Off
  ```

  PHP を Apache モジュールで使用する場合、php.ini の設定を httpd.conf により仮想ホスト単位で、または .htaccess によりディレクトリ単位で上書きすることが可能です。 詳細は、設定の節および Apache マニュアルを参照ください。

- HTTP 出力

  出力の文字エンコーディング変換を有効にする方法は複数あります。 まず php.ini、もうひとつは ob_start() で ob_start のコールバック関数として mb_output_handler() を指定するものです。

  > **注意**: PHP3-i18n のユーザにとっては、mbstring の出力変換は、PHP3-i18n とは異なっています。文字エンコーディングは、出力のバッファリング機能を使用して変換されます。

Example#5 php.ini の設定例

```
;; 全ての PHP ページで出力の文字エンコーディング変換を有効にする

;; 出力バッファリングを有効にする
output_buffering = On

;; mb_output_handler による出力変換を有効にする
output_handler = mb_output_handler
```

Example#6 スクリプトの例

```
<?php

// このページでのみ出力の文字エンコーディング変換を有効にする

// HTTP 出力文字エンコーディングをSJISに設定する
mb_http_output('SJIS');

// 出力のバッファリングを開始し、コールバック関数として"mb_output_handler"
// を指定する
ob_start('mb_output_handler');

?>
```

サポートされる文字エンコーディング

現在、以下の文字エンコーディングが mbstring モジュールによりサポートされています。文字エンコーディングは、 mbstring 関数の encoding パラメータで指定することが可能です。

以下の文字エンコーディングがこの PHP 拡張モジュールでサポートされています。

- UCS-4
- UCS-4BE
- UCS-4LE
- UCS-2
- UCS-2BE
- UCS-2LE
- UTF-32
- UTF-32BE
- UTF-32LE
- UTF-16
- UTF-16BE
- UTF-16LE
- UTF-7
- UTF7-IMAP

- UTF-8
- ASCII
- EUC-JP
- SJIS
- eucJP-win
- SJIS-win
- ISO-2022-JP
- JIS
- ISO-8859-1
- ISO-8859-2
- ISO-8859-3
- ISO-8859-4
- ISO-8859-5
- ISO-8859-6
- ISO-8859-7
- ISO-8859-8
- ISO-8859-9
- ISO-8859-10
- ISO-8859-13
- ISO-8859-14
- ISO-8859-15
- byte2be
- byte2le
- byte4be
- byte4le
- BASE64
- HTML-ENTITIES
- 7bit
- 8bit
- EUC-CN
- CP936
- HZ
- EUC-TW
- CP950
- BIG-5
- EUC-KR
- UHC (CP949)
- ISO-2022-KR
- Windows-1251 (CP1251)
- Windows-1252 (CP1252)
- CP866 (IBM866)
- KOI8-R

エンコーディング名を指定する php.ini エントリには、 "auto" および "pass" を指定することもできます。 エンコーディング名を指定する mbstring 関数にも、"auto" を指定することができます。

"pass" が指定された場合、 文字エンコーディングの変換は行われません。

"auto" が指定された場合、この文字列は "ASCII,JIS,UTF-8,EUC-JP,SJIS" に変換されます。

mb_detect_order() も参照ください。

マルチバイト対応版関数による既存関数のオーバーロード

PHP アプリケーションの多くは、英語等のシングルバイトの言語用に設計されており、 日本語を含むマルチバイト文字列を扱う場合には問題を生じる場合があります。 substr() 等の PHP の文字列関数の多くは、 マルチバイト文字列に対応していません。

マルチバイト拡張モジュール (mbstring) では、文字列を処理する PHP 関数のマルチバイト対応版 (例えば substr() の場合は mb_substr()) をサポートしています。

PHP 4.2.0 以降のマルチバイト拡張モジュール (mbstring) では、 対応するマルチバイト文字対応版の関数で既存の PHP 関数を オーバーロードする機能をサポートします。関数のオーバーロードを行うと、例えば substr() を PHP スクリプトでコールした場合に、 mb_substr() が代わりにコールされるようになります。 これにより、マルチバイト文字に対応しないアプリケーションの移植が容易となります。

関数オーバーロードを使用するには、php.ini の mbstring.func_overload ディレクティブに正の値を指定します。 これは、オーバーロードされる関数の種類を指定するビットマスクの組み合わせとなります。 mail() 関数をオーバーロードするには 1 を指定します。 2 は文字列関数、4 は正規表現関数を表します。つまり、例えば 7 を指定すると、メール関数、文字列関数および正規表現関数が オーバーロードされることになります。オーバーロードされる関数の一覧を以下に示します。

オーバーロードされる関数

mbstring.func_overload の値	元の関数	オーバーロードする関数
1	mail()	mb_send_mail()
2	strlen()	mb_strlen()
2	strpos()	mb_strpos()
2	strrpos()	mb_strrpos()
2	substr()	mb_substr()
4	ereg()	mb_ereg()
4	eregi()	mb_eregi()
4	ereg_replace()	mb_ereg_replace()
4	eregi_replace()	mb_eregi_replace()
4	split()	mb_split()

***注意**: ディレクトリ単位の設定でこのオプションを使用することは推奨されません。 これは、実際の運用環境ではまだ安定性が確認されておらず、 予期しない結果をもたらす可能性があるためです。*

日本語のマルチバイト文字に関する基本事項

日本語の文字は、マルチバイトエンコーディングを使用しないと表せません。 また、プラットフォームや使用目的によって複数の標準エンコーディングが 使い分けられています。さらに悪いことに、 これらの複数の標準エンコーディングはそれぞれ微妙に異なります。 日本語環境で使用しやすいウェブアプリケーションを作成するには、 これらの複雑な問題を考慮した上で 適切な文字エンコーディングを使用しなければなりません。

- *1 文字は最大 6 バイトになる*
- *ほとんどの日本語マルチバイト文字は、シングルバイト文字の 2 倍の幅となります。これらの文字のことを、日本語では "全角 (zen-kaku)" と呼びます。これは、 "full width" という意味です。一方、幅の狭い文字のことは "半角 (han-kaku)" と呼びます。これは、 "half width" という意味です。しかしながら、 文字の見た目は、それを表示する際に使用するタイプフェイスに依存します。*
- *いくつかの文字エンコーディングでは、ISO-2022 で定義されたシフト (エスケープ) シーケンスを使用して、特定のコード範囲 (00h から 7fh まで) のコードマップを切り替えます。*
- *SMTP/NNTPでは、ISO-2022-JP を使用する必要があり、ヘッダとエンティティは 各 RFC の規定に基づき再度符号化される必要があります。これらは必須のものではありませんが、多くの一般的なユーザーエージェントは、 他の符号化手法を認識できないため、行っておく方が良いでしょう。*
- *» i-mode, » Vodafone live!, または » EZweb のような携帯電話サービス用に作成された Web ページは、 Shift_JIS を使用することになります。*

リファレンス

マルチバイト文字エンコーディングおよびそれに関連する問題は非常に複雑で、 このドキュメントの範囲を超えています。これらの問題に関連するより詳細な情報は、 以下の URL やその他のリソースを参照ください。

- *Unicode について*

  » http://www.unicode.org/

- *日本語/韓国語/中国語文字に関する情報*

  » http://examples.oreilly.com/cjkvinfo/doc/cjk.inf

サポートされるエンコーディングの概要

サポートされるエンコーディングの概要
***IANA 文字セット登録名:**ISO-10646-UCS-4*
***依存する文字集合:**ISO 10646*
***説明:** 31 ビットコード空間を使用するユニバーサル文字セットで、 ISO/IEC 10646 によって UCS-4 として標準化されています。 最新版の Unicode コードマップと連動しています。*
***注記:** この名前をエンコーディング変換の際に使用すると、 先頭の BOM (バイトオーダーマーク) にもとづいてそれ以降のバイト列のエンディアンを識別します。*
***IANA 文字セット登録名:**ISO-10646-UCS-4*
***依存する文字集合:**UCS-4*
***説明:** 上を参照ください。*
***注記:** UCS-4 とは対象的に、 文字列が常にビッグエンディアン形式とみなされます。*
***IANA 文字セット登録名:**ISO-10646-UCS-4*
***依存する文字集合:**UCS-4*
***説明:** 上を参照ください。*
***注記:** UCS-4 とは対象的に、 文字列が常にリトルエンディアン形式とみなされます。*
***IANA 文字セット登録名:**ISO-10646-UCS-2*
***依存する文字集合:**UCS-2*
***説明:** 16 ビットコード空間を使用するユニバーサル文字セットで、 ISO/IEC 10646 によって UCS-2 として標準化されています。 最新版の Unicode コードマップと連動しています。*
***注記:** この名前をエンコーディング変換の際に使用すると、 先頭の BOM (バイトオーダーマーク) にもとづいてそれ以降のバイト列のエンディアンを識別します。*
***IANA 文字セット登録名:**ISO-10646-UCS-2*
***依存する文字集合:**UCS-2*
***説明:** 上を参照ください。*
***注記:** UCS-2 とは対象的に、 文字列が常にビッグエンディアン形式とみなされます。*
***IANA 文字セット登録名:**ISO-10646-UCS-2*
***依存する文字集合:**UCS-2*
***説明:** 上を参照ください。*
***注記:** UCS-2 とは対象的に、 文字列が常にリトルエンディアン形式とみなされます。*
***IANA 文字セット登録名:**UTF-32*

依存する文字集合: Unicode
説明: 32 ビット幅の Unicode 変換フォーマットで、 そのエンコーディング空間は Unicode のコードセット標準を参照します。 このエンコーディング体系は UCS-4 とは異なります。 なぜなら、Unicode のコード空間は 21 ビットまでに制限されるからです。
注記: この名前をエンコーディング変換の際に使用すると、 先頭の BOM (バイトオーダーマーク) にもとづいてそれ以降のバイト列のエンディアンを識別します。
IANA 文字セット登録名:UTF-32BE
依存する文字集合: Unicode
説明:上を参照ください。
注記: UTF-32 とは対象的に、 文字列が常にビッグエンディアン形式とみなされます。
IANA 文字セット登録名:UTF-32LE
依存する文字集合: Unicode
説明:上を参照ください。
注記: UTF-32 とは対象的に、 文字列が常にリトルエンディアン形式とみなされます。
IANA 文字セット登録名:UTF-16
依存する文字集合: Unicode
説明: 16 ビット幅の Unicode 変換フォーマットです。 注意すべき点は、UTF-16 の仕様が UCS-2 とは異なることです。 なぜなら、Unicode 2.0 より導入されたサロゲート機能により、 UTF-16 は現在 21 ビットコード空間を参照しているからです。
注記: この名前をエンコーディング変換の際に使用すると、 先頭の BOM (バイトオーダーマーク) にもとづいてそれ以降のバイト列のエンディアンを識別します。
IANA 文字セット登録名:UTF-16BE
依存する文字集合: Unicode
説明: 上を参照ください。
注記: UTF-16 とは対象的に、 文字列が常にビッグエンディアン形式とみなされます。
IANA 文字セット登録名:UTF-16LE
依存する文字集合: Unicode
説明: 上を参照ください。
注記: UTF-16 とは対象的に、 文字列が常にリトルエンディアン形式とみなされます。
IANA 文字セット登録名:UTF-8
依存する文字集合: Unicode / UCS
説明: 8 ビット幅の Unicode 変換フォーマットです。
注記:none
IANA 文字セット登録名:UTF-7
依存する文字集合: Unicode
説明: メールで安全に使用できる Unicode 変換フォーマットです。 » RFC2152 で定義されています。
注記:none
IANA 文字セット登録名:(none)
依存する文字集合: Unicode
説明: UTF-7 の変化形です。 » IMAP プロトコル での使用に特化しています。
注記:none
IANA 文字セット登録名: US-ASCII (推奨される MIME 名) / iso-ir-6 / ANSI_X3.4-1986 / ISO_646.irv:1991 / ASCII / ISO646-US / us / IBM367 / CP367 / csASCII
依存する文字集合:ASCII / ISO 646
説明: American Standard Code for Information Interchange は、一般的に使用される 7 ビットエンコーディングです。 国際標準規格 ISO 646 として標準化されています。
注記:(none)
IANA 文字セット登録名: EUC-JP (推奨される MIME 名) / Extended_UNIX_Code_Packed_Format_for_Japanese / csEUCPkdFmtJapanese
依存する文字集合: US-ASCII / JIS X0201:1997 (半角カナの部分) / JIS X0208:1990 / JIS X0212:1990 を合成したもの
説明: この名前が Extended UNIX Code Packed Format for Japanese を短縮したものであることからわかるように、 一般的に UNIX 系のプラットフォームで用いられます。 もととなるエンコーディング方式である Extended UNIX Code は、ISO 2022 にもとづいて設計されています。
注記: EUC-JP が参照している文字セットは IBM932 / CP932 のものとは異なります。 これらはそれぞれ OS/2® および Microsoft® Windows® で用いられています。これらのプラットフォームとの間で情報をやり取りする場合は、 代わりに EUCJP-WIN を使用してください。
IANA 文字セット登録名: Shift_JIS (推奨される MIME 名) / MS_Kanji / csShift_JIS
依存する文字集合:JIS X0201:1997 / JIS X0208:1997 を合成したもの
説明: Shift_JIS が開発されたのは 80 年代初期です。 当時は日本語ワープロが普及していたため、 旧来のエンコーディング方式である JIS X 0201:1976 との互換性を保つために開発されました。 IANA の定義によると、Shift_JIS のコードセットは IBM932 / CP932 とは微妙に異なります。 しかし、"SJIS" / "Shift_JIS" という名前は、 これらのコードセットを表すものとしてしばしば誤用されています。
注記:CP932 コードマップを使用するには、代わりに SJIS-WIN を使用してください。
IANA 文字セット登録名:(none)
依存する文字集合: JIS X0201:1997 / JIS X0208:1997 / IBM 拡張文字 / NEC 拡張文字 を合成したもの
説明: この "エンコーディング" は EUC-JP と同じエンコーディング方式を使用しますが、 もととなる文字セットが異なります。 つまり、EUC-JP とは異なる文字に対応するコードポイントがあるということです。
注記:none
IANA 文字セット登録名: Windows-31J / csWindows31J
依存する文字集合: JIS X0201:1997 / JIS X0208:1997 / IBM 拡張文字 / NEC 拡張文字 を合成したもの
説明: この "エンコーディング" は Shift_JIS と同じエンコーディング方式を使用しますが、 もととなる文字セットが異なります。 つまり、Shift_JIS とは異なる文字に対応するコードポイントがあるということです。
注記:(none)
IANA 文字セット登録名:ISO-2022-JP (推奨される MIME 名) / csISO2022JP
依存する文字集合: US-ASCII / JIS X0201:1976 / JIS X0208:1978 / JIS X0208:1983
説明:» RFC1468
注記:(none)
IANA 文字セット登録名:JIS
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-1
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-2
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-3
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-4
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-5
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-6
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-7
依存する文字集合:

説明:
注記:
IANA 文字セット登録名:ISO-8859-8
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-9
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-10
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-13
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-14
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-8859-15
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:byte2be
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:byte2le
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:byte4be
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:byte4le
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:BASE64
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:HTML-ENTITIES
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:7bit
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:8bit
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:EUC-CN
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:CP936
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:HZ
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:EUC-TW
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:CP950
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:BIG-5
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:EUC-KR
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:UHC (CP949)
依存する文字集合:
説明:
注記:
IANA 文字セット登録名:ISO-2022-KR
依存する文字集合:
説明:
注記:
IANA 文字セット登録名: Windows-1251 (CP1251)
依存する文字集合:
説明:
注記:
IANA 文字セット登録名: Windows-1252 (CP1252)
依存する文字集合:
説明:

注記:
IANA 文字セット登録名: CP866 (IBM866)
依存する文字集合:
説明:
注記:
IANA 文字セット登録名: KOI8-R
依存する文字集合:
説明:
注記:

---

mb_check_encoding

(PHP 4 >= 4.4.3, PHP 5 >= 5.1.3)

mb_check_encoding — 文字列が、指定したエンコーディングで有効なものかどうかを調べる

説明

bool **mb_check_encoding** ([string $var [, string $encoding]])

そのバイトストリームが指定したエンコーディングで有効なものかどうかを調べます。 これは、いわゆる「不正なエンコーディングによる攻撃」を防ぐのに役立ちます。

パラメータ

var

調べるバイトストリーム。省略した場合は、 リクエスト開始時からのすべての入力が対象となります。

encoding

期待するエンコーディング。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

mb_convert_case

(PHP 4 >= 4.3.0, PHP 5)

mb_convert_case — 文字列に対してケースフォルディングを行う

説明

string **mb_convert_case** (string $str , int $mode [, string $encoding])

mode で指定された方法で string に対してケースフォルディングを行います。

パラメータ

str

変換される文字列。

mode

変換モード。 **MB_CASE_UPPER**、 **MB_CASE_LOWER** あるいは **MB_CASE_TITLE** のいずれかです。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

mode で指定された方法で string に対してケースフォルディングを行った結果を返します。

Unicode

標準のケースフォルディング関数である strtolower() や strtoupper() と違い、ケースフォルディングは Unicode 文字属性を基準に行われます。したがって、この関数の挙動は ロケールの設定に影響されず、また、すべてのアルファベット、 例えば A ウムラウト (Ä) を変換することができます。

Unicode 文字属性についての詳細は » http://www.unicode.org/unicode/reports/tr21/ を参照してください。

例

Example#1 mb_convert_case() の例

```php
<?php
$str = "mary had a Little lamb and she loved it so";
$str = mb_convert_case($str, MB_CASE_UPPER, "UTF-8");
echo $str; // Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
$str = mb_convert_case($str, MB_CASE_TITLE, "UTF-8");
echo $str; // Prints Mary Had A Little Lamb And She Loved It So
?>
```

参考

- mb_strtolower()
- mb_strtoupper()
- strtolower()
- strtoupper()
- ucfirst()
- ucwords()

---

mb_convert_encoding

(PHP 4 >= 4.0.6, PHP 5)

mb_convert_encoding — 文字エンコーディングを変換する

説明

string **mb_convert_encoding** (string $str , string $to_encoding [, mixed $from_encoding])

文字列 str の文字エンコーディングを、 オプションで指定した from_encoding から to_encoding に変換します。

パラメータ

str

変換する文字列。

to_encoding

str の変換後の文字エンコーディング。

from_encoding

変換前の文字エンコーディング名を指定します。 これは、配列またはカンマ区切りの文字列とすることが可能です。 from_encoding を指定しなかった場合は、内部文字エンコーディングを使用します。

"auto" を指定すると、 "ASCII,JIS,UTF-8,EUC-JP,SJIS" に展開されます。

返り値

変換後の文字列を返します。

例

Example#1 mb_convert_encoding() の例

```
<?php
/* 内部文字エンコーディングからSJISに変換 */
$str = mb_convert_encoding($str, "SJIS");

/* EUC-JPからUTF-7に変換 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* JIS, eucjp-win, sjis-winの順番で自動検出し、UCS-2LEに変換 */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" は、"ASCII,JIS,UTF-8,EUC-JP,SJIS" に展開される */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
?>
```

参考

- mb_detect_order()

---

mb_convert_kana

(PHP 4 >= 4.0.6, PHP 5)

mb_convert_kana — カナを("全角かな"、"半角かな"等に)変換する

説明

string **mb_convert_kana** (string $str [, string $option [, string $encoding]])

文字列 str に関して「半角」-「全角」変換を行います。 この関数は、日本語のみで使用可能です。

パラメータ

str

変換される文字列。

option

変換オプション。デフォルト値は "KV" です。

以下のオプションを組み合わせて指定します。デフォルト値は KV です。

使用可能な変換オプション

オプション	意味
r	「全角」英字を「半角」に変換します。
R	「半角」英字を「全角」に変換します。
n	「全角」数字を「半角」に変換します。
N	「半角」数字を「全角」に変換します。
a	「全角」英数字を「半角」に変換します。
A	「半角」英数字を「全角」に変換します （"a", "A" オプションに含まれる文字は、U+0022, U+0027, U+005C, U+007Eを除く U+0021 - U+007E の範囲です）。
s	「全角」スペースを「半角」に変換します（U+3000 -> U+0020）。
S	「半角」スペースを「全角」に変換します（U+0020 -> U+3000）。
k	「全角カタカナ」を「半角カタカナ」に変換します。
K	「半角カタカナ」を「全角カタカナ」に変換します。
h	「全角ひらがな」を「半角カタカナ」に変換します。
H	「半角カタカナ」を「全角ひらがな」に変換します。
c	「全角カタカナ」を「全角ひらがな」に変換します。
C	「全角ひらがな」を「全角カタカナ」に変換します。
V	濁点付きの文字を一文字に変換します。"K", "H" と共に使用します。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

変換後の文字列を返します。

例

Example#1 mb_convert_kana() の例

```
<?php
/* 「仮名」を全て「全角カタカナ」に変換します */
$str = mb_convert_kana($str, "KVC");

/* 「半角カタカナ」を「全角カタカナ」に変換し、「全角」英数字を「半角」
   に変換します。 */
$str = mb_convert_kana($str, "KVa");
?>
```

---

mb_convert_variables

(PHP 4 >= 4.0.6, PHP 5)

mb_convert_variables — 変数の文字コードを変換する

説明

string **mb_convert_variables** (string $to_encoding , mixed $from_encoding , mixed &$vars [, mixed &$...])

エンコーディング from_encoding の変数 vars をエンコーディング to_encoding に変換します。

mb_convert_variables() は、エンコーディング検出のために Array または Object の文字列を結合します。これは、 エンコーディング検出は短い文字列では失敗する傾向があるためです。このため、 1 つの配列またはオブジェクトで異なるエンコーディングを混ぜることはできません。

パラメータ

to_encoding

文字列の変換後のエンコーディング。

from_encoding

from_encoding には配列またはカンマ区切りの文字列を指定し、 from-coding からエンコーディングの検出を試みます。 from_encoding が省略された場合、detect_order を使用します。

vars

vars (3番目以降の引数)は、変換する変数への リファレンスです。文字列、配列、オブジェクトを指定することが可能です。
mb_convert_variables() は全てのパラメータが 同じエンコーディングを有することを仮定します。

...

追加の変数。

返り値

成功時に変換前の文字エンコーディングを返し、失敗した場合に **FALSE** を返します。

例

Example#1 mb_convert_variables() の例

```
<?php
/* 変数 $post1, $post2 を内部エンコーディングに変換する */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
?>
```

---

mb_decode_mimeheader

(PHP 4 >= 4.0.6, PHP 5)

mb_decode_mimeheader — MIME ヘッダフィールドの文字列をデコードする

説明

string **mb_decode_mimeheader** (string $str)

エンコードされた MIME ヘッダの文字列 str をデコードします。

パラメータ

str

デコードする文字列。

返り値

内部文字エンコーディングでデコードされた文字列を返します。

参考

- mb_encode_mimeheader()

---

mb_decode_numericentity

(PHP 4 >= 4.0.6, PHP 5)

mb_decode_numericentity — HTML 数値エンティティを文字にデコードする

説明

string **mb_decode_numericentity** (string $str , array $convmap [, string $encoding])

文字列 str において指定した文字領域にある数値エンティティを変換し、 変換後の文字列を返します。

パラメータ

str

デコードする文字列。

convmap

convmap は変換するコード領域を指定する配列です。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

変換された文字列を返します。

例

Example#1 convmap の例

```
$convmap = array (
    int start_code1, int end_code1, int offset1, int mask1,
    int start_code2, int end_code2, int offset2, int mask2,
    ........
    int start_codeN, int end_codeN, int offsetN, int maskN );
// start_codeN および end_codeN に Unicode値を指定
// 値にoffsetNを追加、マスクmaskNを指定してビット毎の'AND'をとり、
// 数値エンティティに値を変換します。
```

参考

- mb_encode_numericentity()

---

mb_detect_encoding

(PHP 4 >= 4.0.6, PHP 5)

mb_detect_encoding — 文字エンコーディングを検出する

説明

string **mb_detect_encoding** (string $str [, mixed $encoding_list [, bool $strict]])

文字列 str の文字エンコーディングを検出します。

パラメータ

str

検出する文字列。

encoding_list

encoding_list は文字エンコーディングのリストで、 エンコーディング検出の順番を配列またはカンマ区切りのリストで指定します。

encoding_list が省略された場合、 detect_order が使用されます。

strict

strict は、 厳格なエンコーディング検出を行うかどうかを指定します。 デフォルトは **FALSE** です。

返り値

検出した文字エンコーディングを返します。

例

Example#1 mb_detect_encoding() の例

```
<?php
/* 現在のdetect_orderで文字エンコーディングを検出 */
echo mb_detect_encoding($str);

/* "auto" は "ASCII,JIS,UTF-8,EUC-JP,SJIS"に展開されます */
echo mb_detect_encoding($str, "auto");

/* カンマ区切りのリストで encoding_list 文字エンコーディングを指定 */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* encoding_list を指定するために配列を使用 */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
?>
```

参考

- mb_detect_order()

---

mb_detect_order

(PHP 4 >= 4.0.6, PHP 5)

mb_detect_order — 文字エンコーディング検出順序を設定あるいは取得する

説明

mixed **mb_detect_order** ([mixed $encoding_list])

自動文字エンコーディング検出の順番を encoding_list に設定します。

パラメータ

encoding_list

encoding_list は、 配列またはカンマ区切りの文字エンコーディングのリストです ("auto" は、 "ASCII, JIS, UTF-8, EUC-JP, SJIS"に展開されます) 。

encoding_list が省略された場合は、 現在の文字エンコーディング検出順を配列で返します。

この設定は、mb_detect_encoding() および mb_send_mail() に影響します。

mbstring が現在実装しているのは、 以下のエンコーディングを検出するフィルタです。 以下のエンコーディングにおいて無効なバイトシーケンスがあった場合、 エンコーディング検出は失敗します。

UTF-8, UTF-7, ASCII, EUC-JP,SJIS, eucJP-win, SJIS-win, JIS, ISO-2022-JP

ISO-8859-*の場合、mbstring は常に ISO-8859-* として検出します。

UTF-16, UTF-32, UCS2, UCS4 の場合、 エンコーディング検出は常に失敗します。

Example#1 無意味な順番の例

; 常に ISO-8859-1 として検出されます

```
detect_order = ISO-8859-1, UTF-8

; ASCII/UTF-7 の値は UTF-8 として有効なため、常に UTF-8 として検出されます
detect_order = UTF-8, ASCII, UTF-7
```

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#2 mb_detect_order() の例

```
<?php
/* リストで検出順を設定 */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* 配列で検出順を設定 */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* 現在の検出順を表示 */
echo implode(", ", mb_detect_order());
?>
```

参考

- mb_internal_encoding()
- mb_http_input()
- mb_http_output()
- mb_send_mail()

---

mb_encode_mimeheader

(PHP 4 >= 4.0.6, PHP 5)

mb_encode_mimeheader — MIMEヘッダの文字列をエンコードする

説明

string **mb_encode_mimeheader** (string $str [, string $charset [, string $transfer_encoding [, string $linefeed [, int $indent]]]])

MIME ヘッダエンコーディング方式によって文字列 str をエンコードします。

パラメータ

str

エンコードする文字列。

charset

charset は、str の文字セット名です。デフォルトは、現在の NLS 設定 (mbstring.language) によって決まります。

transfer_encoding

transfer_encoding は MIME エンコーディングの 方式を指定します。"B" (Base64) または "Q" (Quoted-Printable) のどちらかでなければなりません。 デフォルトは "B" です。

linefeed

linefeed は EOL (行末) のマーカで、 **mb_encode_mimeheader()** が行を折りたたむ (» RFC 用語で、 ある一定より長い行を複数行に分割することを言います。 分割する長さは、現在 74 文字に固定されています) 際に利用します。 デフォルトは "¥r¥n" (CRLF) です。

indent

返り値

文字列を ASCII 表現に変換したものを返します。

変更履歴

バージョン	説明
5.0.0	indent パラメータが追加されました。

例

Example#1 mb_encode_mimeheader() の例

```
<?php
$name = "太郎"; // 漢字
$mbox = "kru";
$doma = "gtinn.mon";
```

```
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $doma . ">";
echo $addr;
?>
```

注意

注意: この関数は、行を分割する際に特別な配慮（単語の区切りなど）を行いません。 このせいで、もとの文字列に予期せぬ空白が入ってしまう可能性があります。

参考

- mb_decode_mimeheader()

---

mb_encode_numericentity

(PHP 4 >= 4.0.6, PHP 5)

mb_encode_numericentity — 文字を HTML 数値エンティティにエンコードする

説明

string **mb_encode_numericentity** (string $str , array $convmap [, string $encoding])

str の中で指定した文字コードを HTML 数値エンティティから文字コードに変換します。

パラメータ

str

エンコードする文字列。

convmap

convmap は、変換するコード領域を指定する配列です。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

変換後の文字列を返します。

例

Example#1 convmap の例

```
$convmap = array (
  int start_code1, int end_code1, int offset1, int mask1,
  int start_code2, int end_code2, int offset2, int mask2,
  ........
  int start_codeN, int end_codeN, int offsetN, int maskN );
// start_codeN および end_codeN に Unicode 値を指定
// 値に offsetN を追加、マスク maskN を指定してビット毎の'AND'をとり、
// 数値エンティティに値を変換します。
```

例

Example#2 mb_encode_numericentity() の例

```
<?php
/* ISO-8859-1 の左面をHTML数値エンティティに変換 */
$convmap = array(0x80, 0xff, 0, 0xff);
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* ブロック 95-104 にある SJIS-win コードのユーザ定義領域を
   数値エンティティに変換 */
$convmap = array(
        0xe000, 0xe03e, 0x1040, 0xffff,
        0xe03f, 0xe0bb, 0x1041, 0xffff,
        0xe0bc, 0xe0fa, 0x1084, 0xffff,
        0xe0fb, 0xe177, 0x1085, 0xffff,
        0xe178, 0xe1b6, 0x10c8, 0xffff,
        0xe1b7, 0xe233, 0x10c9, 0xffff,
        0xe234, 0xe272, 0x110c, 0xffff,
        0xe273, 0xe2ef, 0x110d, 0xffff,
        0xe2f0, 0xe32e, 0x1150, 0xffff,
        0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
?>
```

参考

- mb_decode_numericentity()

---

mb_ereg_match

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_match — マルチバイト文字列が正規表現に一致するか調べる

説明

bool **mb_ereg_match** (string $pattern , string $string [, string $option])

マルチバイト文字列に対する正規表現マッチングを行います。

パラメータ

pattern

正規表現パターン。

string

評価する文字列。

option

返り値

string が正規表現 pattern に一致する場合に**TRUE** 、 そうでない場合に **FALSE** を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg()

---

mb_ereg_replace

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_replace — マルチバイト文字列に正規表現による置換を行う

説明

string **mb_ereg_replace** (string $pattern , string $replacement , string $string [, string $option])

string から pattern にマッチする文字列を探し、 見つかった文字列を replacement で置換します。

パラメータ

pattern

正規表現パターン。

マルチバイト文字を pattern で使用することができます。

replacement

置換文字列。

string

調べたい文字列。

option
option パラメータで、マッチングの動作を変更可能です。 i を指定した場合、大文字・小文字が 区別されなくなります。 x を指定した場合、空白が無視されます。 m を指定した場合、マルチラインモードとなり、 改行文字も "." に含まれるようになります。 p を指定した場合、POSIX モードとなり、 改行も通常文字とみなされるようになります。 e を指定した場合、文字列 replacement がPHPの式として評価されます。

返り値

成功した場合に結果の文字列、エラー時に **FALSE** を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_eregi_replace()

---

mb_ereg_search_getpos

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_search_getpos — 次の正規表現検索を開始する位置を取得する

説明

int **mb_ereg_search_getpos** (void)

次の正規表現マッチングを開始する位置を返します。

パラメータ

この関数にはパラメータはありません。

返り値

mb_ereg_search_getpos() は、 mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs() で検索を開始する位置を返します。位置は、文字列の先頭からのバイト数で表した値です。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_search_setpos()

---

mb_ereg_search_getregs

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_search_getregs — マルチバイト文字列が正規表現に一致する部分があるか調べる

説明

array **mb_ereg_search_getregs** (void)

直近のマルチバイト正規表現マッチングの結果を取得します。

パラメータ

この関数にはパラメータはありません。

返り値

直前の mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs() で一致した部分文字列を含む 配列を返します。一致する部分があった場合には、一致した部分文字列全体が 最初の要素に、最初の括弧でグループ化された部分が 2 番目の要素に、そして 2 番目の括弧でグループ化された部分が 3 番目の要素にといったようになります。 エラー時には **FALSE** を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_search_init()

---

mb_ereg_search_init

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_search_init — マルチバイト正規表現検索用の文字列と正規表現を設定する

説明

bool **mb_ereg_search_init** (string $string [, string $pattern [, string $option]])

mb_ereg_search_init() は、マルチバイト対応の正規 表現検索において検索対象の文字列 string および正規表現 pattern を設定します。 この設定は、mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs() で使用されます。

パラメータ

この関数にはパラメータはありません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_search_regs()

---

mb_ereg_search_pos

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_search_pos — 指定したマルチバイト文字列が正規表現に一致する部分の位置と長さを返す

説明

array **mb_ereg_search_pos** ([string $pattern [, string $option]])

マルチバイト文字列の中で正規表現に一致した部分の位置と長さを配列で返します。

検索対象の文字列は、mb_ereg_search_init() により設定します。省略した場合は、前回のものが利用されます。

パラメータ

pattern

検索パターン。

option

検索オプション。

返り値

マルチバイト文字列の中で正規表現に一致する部分を含む配列を返します。 配列の最初の要素には一致した先頭の位置、2 番目の要素には一致した長さ（バイト数） が格納されます。エラー時は、**FALSE**を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_search_init()

---

mb_ereg_search_regs

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_search_regs — 指定したマルチバイト文字列が正規表現に一致する部分を取得する

説明

array **mb_ereg_search_regs** ([string $pattern [, string $option]])

マルチバイト正規表現にマッチした部分を返します。

パラメータ

pattern

検索パターン。

option

検索オプション。

返り値

mb_ereg_search_regs() は、マルチバイト文字列の 中に正規表現に一致する部分があるかどうか調べ、一致した場合にはそれを 配列で返します。配列の最初の要素には一致した部分文字列全体、2 番目の 要素は最初の括弧でグループ化された部分、3 番目の要素は2番目の括弧で グループ化された部分といったようになります。 エラー時には **FALSE** を返します。

注意

注意: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_search_init()

---

mb_ereg_search_setpos

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_search_setpos — 次の正規表現検索を開始する位置を設定する

説明

bool **mb_ereg_search_setpos** (int $position)

mb_ereg_search_setpos() は、 mb_ereg_search() で検索を開始する位置を設定します。

パラメータ

position

設定する位置。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_search_init()

---

mb_ereg_search

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg_search — 指定したマルチバイト文字列が正規表現に一致するか調べる

説明

bool **mb_ereg_search** ([string $pattern [, string $option]])

指定したマルチバイト文字列に対するマルチバイト正規表現マッチングを行います。

パラメータ

pattern

検索パターン。

option

検索オプション。

返り値

mb_ereg_search()は、マルチバイト文字列が正規表現に一致するかどうか調べ、一致する場合に**TRUE**、それ以外の場合に **FALSE** を返します。検索対象の文字列は、 mb_ereg_search_init() により設定します。 正規表現 pattern を省略した場合は、 前回のものを再利用します。

注意

注意: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_search_init()

---

mb_ereg

(PHP 4 >= 4.2.0, PHP 5)

mb_ereg — マルチバイト文字列に正規表現マッチを行う

説明

int **mb_ereg** (string $pattern , string $string [, array $regs])

マルチバイト対応の正規表現マッチングを行います。

パラメータ

pattern

検索パターン。

string

検索文字列。

regs

マッチした string の部分文字列。

返り値

マルチバイト対応の正規表現マッチを行い、 一致した場合は 1 を返します。オプションの 3 番目の引数を 指定した場合は、一致した部分のバイト数を返し、一致した部分文字列が 配列 regs に格納されます。空文字に 一致した場合は 1 が返されます。正規表現に一致しないか、 エラーを発生した場合に **FALSE** を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_eregi()

---

mb_eregi_replace

(PHP 4 >= 4.2.0, PHP 5)

mb_eregi_replace — マルチバイト文字列に大文字小文字を区別せずに正規表現による置換を行う

説明

string **mb_eregi_replace** (string $pattern , string $replace , string $string [, string $option])

string において、 正規表現 pattern にマッチする文字列を replacement に置換します。

パラメータ

pattern

正規表現パターン。マルチバイト文字を使用できます。 大文字小文字は区別しません。

replace

置換する文字列。

string

検索対象となる文字列。

option
option の意味は、mb_ereg_replace() の場合と同じです。

返り値

結果の文字列、あるいはエラー時に **FALSE** を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg_replace()

---

mb_eregi

(PHP 4 >= 4.2.0, PHP 5)

mb_eregi — マルチバイト文字列に大文字小文字を区別しない正規表現マッチを行う

説明

int **mb_eregi** (string $pattern , string $string [, array $regs])

マルチバイト対応の大文字小文字を区別しない正規表現マッチングを行います。

パラメータ

pattern

正規表現パターン。

string

検索対象の文字列。

regs

マッチした string の部分文字列を格納します。

返り値

マルチバイト対応の大文字小文字を 区別しない正規表現マッチを行い、一致した場合は 1 を返します。 オプションの 3 番目の引数を指定した場合は、一致した部分のバイト数を返し、 一致した部分文字列が配列 regs に格納されます。 空文字に一致した場合は 1 が返されます。 正規表現に一致しないか、エラーを発生した場合に **FALSE** を返します。

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_regex_encoding()
- mb_ereg()

---

mb_get_info

(PHP 4 >= 4.2.0, PHP 5)

mb_get_info — mbstring の内部設定値を取得する

説明

mixed **mb_get_info** ([string $type])

mb_get_info() は、mbstring の内部設定パラメータを返します。

パラメータ

type

type が指定されない場合または "all" が指定された場合、"internal_encoding", "http_output", "http_input", "func_overload", "mail_charset", "mail_header_encoding", "mail_body_encoding" の設定値を有する連想配列が返されます。

type に "http_output", "http_input", "internal_encoding", "func_overload" が指定された場合、 指定された設定パラメータが返されます。

返り値

type が指定されていない場合は型情報を含む配列、 それ以外の場合は指定した type の値を返します。

変更履歴

バージョン	説明
5.1.3	要素型 "mail_charset", "mail_header_encoding" および "mail_body_encoding" が使用できるようになりました。

参考

- mb_regex_encoding()
- mb_http_output()

---

mb_http_input

(PHP 4 >= 4.0.6, PHP 5)

mb_http_input — HTTP 入力文字エンコーディングを検出する

説明

mixed **mb_http_input** ([string $type])

HTTP 入力文字エンコーディングを検出します。

パラメータ

type

HTTP 入力の型を表す文字列を入力してください。 GET の場合は "G"、POST の場合は "P"、 COOKIE の場合は "C"、文字列の場合は "S" 、 リストの場合は "L" 、リスト全体 (array を返す) の場合は "I" です。 type が省略された場合、 直近に処理された入力型が返されます。

返り値

type の文字エンコーディング名を返します。 **mb_http_input()** が指定した HTTP 入力の処理を行っていない場合、**FALSE** を返します。

参考

- mb_internal_encoding()
- mb_http_output()
- mb_detect_order()

---

mb_http_output

(PHP 4 >= 4.0.6, PHP 5)

mb_http_output — HTTP 出力文字エンコーディングを設定あるいは取得する

説明

mixed **mb_http_output** ([string $encoding])

HTTP 出力文字エンコーディングを設定あるいは取得します。 この関数を実行した後、 出力は encoding に変換されます。

パラメータ

encoding

encoding が設定された場合、 **mb_http_output()** は HTTP 出力文字エンコーディングを encoding に設定します。

encoding が省略された場合、 **mb_http_output()** は現在の HTTP 出力文字エンコーディングを返します。

返り値

encoding が省略された場合、 **mb_http_output()** は現在の HTTP 出力文字エンコーディングを返します。それ以外の場合、 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mb_internal_encoding()
- mb_http_input()
- mb_detect_order()

---

mb_internal_encoding

(PHP 4 >= 4.0.6, PHP 5)

mb_internal_encoding — 内部文字エンコーディングを設定あるいは取得する

説明

mixed **mb_internal_encoding** ([string $encoding])

内部文字エンコーディングを設定あるいは取得します。

パラメータ

encoding

encoding は、HTTP 入力文字エンコーディング変換、 HTTP 出力文字エンコーディング変換および mbstring モジュールの文字列関数においてデフォルトの文字エンコーディングとして使用されます。

返り値

encoding が設定された場合、 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 encoding が省略された場合、 現在の文字エンコーディング名を返します。

例

Example#1 mb_internal_encoding() の例

```
<?php
/* 内部文字エンコーディングをUTF-8に設定 */
mb_internal_encoding("UTF-8");

/* 現在の内部文字エンコーディングを表示 */
echo mb_internal_encoding();
?>
```

注意

> **注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- mb_http_input()
- mb_http_output()
- mb_detect_order()

---

mb_language

(PHP 4 >= 4.0.6, PHP 5)

mb_language — 現在の言語を設定あるいは取得する

説明

mixed **mb_language** ([string $language])

現在の言語を設定あるいは取得します。

パラメータ

language

e-mail メッセージのエンコーディングとして使用します。有効な言語は、"Japanese", "ja","English","en", "uni" (UTF-8) です。mb_send_mail() は、e-mail をエンコードする際にこの設定を使用します。

言語とその設定は、Japanese の場合は ISO-2022-JP/Base64、uni の場合は UTF-8/Base64、English の場合は ISO-8859-1/quoted printable です。

返り値

language が設定され、 language が有効な場合、**TRUE**が返されます。 そうでない場合、**FALSE**が返されます。 language が省略された場合、 言語の名前が文字列として返されます。事前に言語が設定されていない場合、 **FALSE** が返されます。

参考

- mb_send_mail()

---

mb_output_handler

(PHP 4 >= 4.0.6, PHP 5)

mb_output_handler — 出力バッファ内で文字エンコーディングを変換するコールバック関数

説明

string **mb_output_handler** (string $contents , int $status)

mb_output_handler() は、 ob_start() のコールバック関数です。 **mb_output_handler()** は、出力バッファの文字を 内部文字エンコーディングから HTTP 出力文字エンコーディングに変換します。

パラメータ

contents

出力バッファの中身。

status

出力バッファの状態。

返り値

変換後の文字列を返します。

変更履歴

バージョン	説明
4.1.0	以下の条件が満たされた場合に、このハンドラは charset HTTP ヘッダを設定します。 • header() で *Content-Type* が設定されていない • デフォルトの MIME 型が *text/* で始まる • mbstring.http_input の設定が *pass* 以外である

例

Example#1 mb_output_handler() の例

```
<?php
mb_http_output("UTF-8");
ob_start("mb_output_handler");
?>
```

注意

> **注意**: PHP 4.3.0 以降において、 イメージのようなバイナリデータを PHP スクリプトから出力したい場合、 バイナリデータを送信する前に header() により Content-Type: ヘッダ(例：header("Content-Type: image/png"))を 送信する必要があります。 Content-Type: ヘッダが送信されると出力文字 コード変換は無効となります。
> ただし、header() により"Content-Type: text/*" を送信した場合には、テキストが送信されるとみなし、文字コード設定に 基づいて出力文字コード変換を行います。
> なお、PHP 4.2.x あるいはそれ以前のバージョンで画像のようなバイナリデータを 出力する場合には、mb_http_output() を用いて出力エンコーディングを "pass" に設定する必要があります。

参考

- ob_start()

---

mb_parse_str

(PHP 4 >= 4.0.6, PHP 5)

mb_parse_str — GET/POST/COOKIE データをパースし、グローバル変数を設定する

説明

bool **mb_parse_str** (string $encoded_string [, array &$result])

GET/POST/COOKIE データをパースし、 グローバル変数を設定します。PHPは、生の POST/COOKIE を提供しないため、 現状では GET データでのみ使用可能です。この関数は、URL エンコードされたデータをパース、エンコーディングを検出、 内部エンコーディングへ変換し、配列result または グローバル配列に設定します。

パラメータ

encoded_string

URL エンコードされたデータ。

result

デコードされ、文字エンコーディング変換された文字列が含まれます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mb_detect_order()
- mb_internal_encoding()

---

mb_preferred_mime_name

(PHP 4 >= 4.0.6, PHP 5)

mb_preferred_mime_name — MIME 文字設定を文字列で得る

説明

string **mb_preferred_mime_name** (string $encoding)

指定したエンコーディングの MIME charset 文字列を取得します。

パラメータ

encoding

調べるエンコーディング。

返り値

文字エンコーディング encoding 用の MIME charset 文字列を返します。

例

Example#1 mb_preferred_mime_string() の例

```
<?php
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
?>
```

---

mb_regex_encoding

(PHP 4 >= 4.2.0, PHP 5)

mb_regex_encoding — 現在の正規表現用のエンコーディングを文字列として返す

説明

mixed **mb_regex_encoding** ([string $encoding])

現在のマルチバイト正規表現用のエンコーディングを文字列として返す

パラメータ

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

マルチバイト対応の正規表現関数で用いる文字エンコーディングを返します。

参考

- mb_internal_encoding()
- mb_ereg()

---

mb_regex_set_options

(PHP 4 >= 4.3.0, PHP 5)

mb_regex_set_options — マルチバイト正規表現関数のデフォルトオプションを取得または設定する

説明

string **mb_regex_set_options** ([string $options])

options で表されるオプションを、 マルチバイト対応の正規表現関数のデフォルトに設定します。

パラメータ

options

設定するオプション。

返り値

以前設定されていたオプション文字列を返します。もし options が省略されていた場合、 現在のデフォルトのオプションを返します。

参考

- mb_split()
- mb_ereg()
- mb_eregi()

---

mb_send_mail

(PHP 4 >= 4.0.6, PHP 5)

mb_send_mail — エンコード変換を行ってメールを送信する

説明

*bool **mb_send_mail** (string $to , string $subject , string $message [, string $additional_headers [, string $additional_parameter]])*

email を送信します。ヘッダと本文は mb_language() の設定に基づき変換、エンコードされます。 これは mail() のラッパー関数です。詳細は、mail() を参照ください。

パラメータ

to

送信先のメールアドレス。 各アドレスをカンマで区切ると、複数の宛先を to に指定できます。 このパラメータは、自動的にはエンコードされません。

subject

メールの件名。

message

メールの本文。

additional_headers

additional_headers は、 ヘッダの最後に挿入されます。これは通常、ヘッダを追加する際に使用されます。 改行 ("\n") で区切ることにより複数のヘッダを指定可能です。

additional_parameter

additional_parameter は、MTA へ渡す コマンドライン引数です。sendmail を利用する際に正しい Return-Path を設定するためなどに利用すると便利です。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

変更履歴

バージョン	説明
5.0.0	PHP 5.0.0 以降では、*Content-Type* および *Content-Transfer-Encoding* ヘッダの内容が 再定義可能となりました。PHP 4 では、常に mb_language() で定義された値が用いられていました。

参考

- *mail()*
- ***mb_encode_mimeheade()***
- *mb_language()*

---

mb_split

(PHP 4 >= 4.2.0, PHP 5)

mb_split — マルチバイト文字列を正規表現により分割する

説明

*array **mb_split** (string $pattern , string $string [, int $limit])*

マルチバイト文字列 string において、正規表現 pattern により文字列を分割し、 結果を配列として返します。

パラメータ

pattern

正規表現パターン。

string

分割する文字列。

limit
オプションの引数 limit を指定した場合は、 最大 limit 個の要素に分割されます。

返り値

結果を配列で返します。

注意

> ***注意**: 内部エンコーディングあるいは mb_regex_encoding() で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。*

参考

- mb_regex_encoding()
- mb_ereg()

---

mb_strcut

(PHP 4 >= 4.0.6, PHP 5)

mb_strcut — 文字列の一部を得る

説明

string **mb_strcut** (string $str , int $start [, int $length [, string $encoding]])

mb_strcut() は、mb_substr() と同じ処理を異なった方法で行います。位置 start がマルチバイト文字の 2 バイト目以降で ある場合、マルチバイト文字の最初のバイトから開始されます。

この関数は、length より短く、 かつマルチバイト文字列の一部でないかシフトシーケンスの中にない文字を 取りだします。

パラメータ

str

取り出しの対象となる文字列。

start

取り出しの開始位置。

length

取り出す長さ。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

mb_strcut() は、 start および length パラメータで指定した str の一部を返します。

参考

- mb_substr()
- mb_internal_encoding()

---

mb_strimwidth

(PHP 4 >= 4.0.6, PHP 5)

mb_strimwidth — 指定した幅で文字列を丸める

説明

string **mb_strimwidth** (string $str , int $start , int $width [, string $trimmarker [, string $encoding]])

文字列 str を指定した幅 width で丸めます。

パラメータ

str

丸めたい文字列。

start

開始位置のオフセット。文字列の始めからの文字数（最初の文字は 0）です。

width

丸める幅。

trimmarker

丸めた後にその文字列の最後に追加される文字列。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

丸められた文字列を返します。 trimmarker が設定された場合、 trimmarker が丸められた文字列に追加されます。

例

Example#1 mb_strimwidth() の例

```
<?php
$str = mb_strimwidth($str, 0, 40, "..>");
?>
```

参考

- mb_strwidth()
- mb_internal_encoding()

---

mb_stripos

(PHP 5 >= 5.2.0)

mb_stripos — 大文字小文字を区別せず、 文字列の中で指定した文字列が最初に現れる位置を探す

説明

int **mb_stripos** (string $haystack , string $needle [, int $offset [, string $encoding]])

mb_stripos() は、 needle が haystack の中で最初に現れる位置を返します。 mb_strpos() とは異なり、 **mb_stripos()** は大文字小文字を区別しません。 needle が見つからなかった場合は **FALSE** を返します。

パラメータ

haystack

needle が最初に現れる位置を見つける文字列。

needle

haystack の中で探す文字列。

offset

haystack の中で、検索を開始する位置。

encoding

使用する文字エンコーディング名。 省略した場合は内部文字エンコーディングが用いられます。

返り値

needle が haystack の中で最初に現れる位置を返します。needle が見つからない場合は **FALSE** を返します。

参考

- stripos()
- strpos()
- mb_strpos()

---

mb_stristr

(PHP 5 >= 5.2.0)

mb_stristr — 大文字小文字を区別せず、 文字列の中で指定した文字列が最初に現れる位置を探す

説明

string **mb_stristr** (string $haystack , string $needle [, bool $part [, string $encoding]])

mb_stristr() は、 haystack の中で最初に needle が現れる場所を探し、 haystack の部分文字列を返します。 mb_strstr() とは異なり、 **mb_stristr()** は大文字小文字を区別しません。 needle が見つからなかった場合は **FALSE** を返します。

パラメータ

haystack

needle が最初に現れる位置を見つける文字列。

needle

haystack の中で探す文字列。

part

この関数が haystack のどの部分を返すのかを指定します。 **TRUE** の場合は、haystack の先頭から needle が最初に現れる位置までを返します。 **FALSE** の場合は、needle が最初に現れる位置から haystack の最後までを返します。 デフォルト値は **FALSE** です。

encoding

使用する文字エンコーディング名。 省略した場合は内部文字エンコーディングが用いられます。

返り値

haystack の部分文字列を返します。 needle が見つからない場合は FALSE を返します。

参考

- stristr()
- strstr()
- mb_strstr()

---

mb_strlen

(PHP 4 >= 4.0.6, PHP 5)

mb_strlen — 文字列の長さを得る

説明

int **mb_strlen** (string $str [, string $encoding])

文字列の長さを取得します。

パラメータ

str

長さを調べたい文字列。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

文字エンコーディング encoding の文字列 str の文字数を返します。 マルチバイト文字の一文字は1個として数えられます。

参考

- mb_internal_encoding()
- strlen()

---

mb_strpos

(PHP 4 >= 4.0.6, PHP 5)

mb_strpos — 文字列の中に指定した文字列が最初に現れる位置を見つける

説明

int **mb_strpos** (string $haystack , string $needle [, int $offset [, string $encoding]])

ある文字列の中で別の文字列が最初に現れる位置を見つけます。

マルチバイト文字列に正しくマッチするように strpos() を拡張したもので、 最初の 1 文字目の位置が 0 、2 文字目の文字が 1 というようになります。

パラメータ

haystack

調べたい文字列。

needle

haystack の中での位置を調べる文字列。

offset

検索オフセット。 指定されない場合は、0 が使用されます。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

文字列 haystack の中で needle が最初に現れる位置を数字で返します。 needle が見付からなかった場合、FALSE を返します。

参考

- mb_()

- mb_internal_encoding()
- strpos()

---

mb_strrchr

(PHP 5 >= 5.2.0)

mb_strrchr — 別の文字列の中で、ある文字が最後に現れる場所を見つける

説明

string **mb_strrchr** (string $haystack , string $needle [, bool $part [, string $encoding]])

mb_strrchr() は、 haystack の中で最後に needle が現れる場所を探し、 haystack の部分文字列を返します。 needle が見つからなかった場合は **FALSE** を返します。

パラメータ

haystack

needle が最後に現れる位置を探す文字列。

needle

haystack の中で探す文字列。

part

この関数が haystack のどの部分を返すのかを指定します。 **TRUE** の場合は、haystack の先頭から needle が最後に現れる位置までを返します。 **FALSE** の場合は、needle が最後に現れる位置から haystack の最後までを返します。 デフォルト値は **FALSE** です。

encoding

使用する文字エンコーディングの名前。 省略した場合は内部文字エンコーディングを使用します。

返り値

haystack の部分文字列を返します。 needle が見つからない場合は **FALSE** を返します。

参考

- strrchr()
- mb_strstr()
- mb_strrichr()

---

mb_strrichr

(PHP 5 >= 5.2.0)

mb_strrichr — 大文字小文字を区別せず、 別の文字列の中である文字が最後に現れる場所を探す

説明

string **mb_strrichr** (string $haystack , string $needle [, bool $part [, string $encoding]])

mb_strrichr() は、 haystack の中で最後に needle が現れる場所を探し、 haystack の部分文字列を返します。 mb_strrchr() とは異なり、 **mb_strrichr()** は大文字小文字を区別しません。 needle が見つからなかった場合は **FALSE** を返します。

パラメータ

haystack

needle が最後に現れる位置を探す文字列。

needle

haystack の中で探す文字列。

part

この関数が haystack のどの部分を返すのかを指定します。 **TRUE** の場合は、haystack の先頭から needle が最後に現れる位置までを返します。 **FALSE** の場合は、needle が最後に現れる位置から haystack の最後までを返します。 デフォルト値は **FALSE** です。

encoding

使用する文字エンコーディングの名前。 省略した場合は内部文字エンコーディングを使用します。

返り値

haystack の部分文字列を返します。 needle が見つからない場合は **FALSE** を返します。

参考

- mb_stristr()
- mb_strrchr()

---

mb_strripos

(PHP 5 >= 5.2.0)

mb_strripos — 大文字小文字を区別せず、 文字列の中で指定した文字列が最後に現れる位置を探す

説明

int **mb_strripos** (string $haystack , string $needle [, int $offset [, string $encoding]])

mb_strripos() は、マルチバイト対応の strripos() 操作を、文字数に基づいて行います。 needle の位置を haystack の先頭から順に数えていきます。 最初の文字の位置は 0、二番目の文字の位置は 1 という具合です。 mb_strrpos() とは異なり、 **mb_strripos()** は大文字小文字を区別しません。

パラメータ

haystack

needle が最後に現れる位置を見つける文字列。

needle

haystack の中で探す文字列。

offset

haystack の中で、検索を開始する位置。

encoding

使用する文字エンコーディング名。 省略した場合は内部文字エンコーディングが用いられます。

返り値

needle が haystack の中で最後に現れる位置を返します。needle が見つからない場合は **FALSE** を返します。

参考

- strripos()
- strrpos()
- mb_strrpos()

---

mb_strrpos

(PHP 4 >= 4.0.6, PHP 5)

mb_strrpos — 文字列の中に指定した文字列が最後に現れる位置を見つける

説明

int **mb_strrpos** (string $haystack , string $needle [, int $offset [, string $encoding]])

mb_strrpos() は、マルチバイト対応の strrpos() 操作を、文字数に基づいて行います。 needle の位置を haystack の先頭から順に数えていきます。 最初の文字の位置は 0、二番目の文字の位置は 1 という具合です。

パラメータ

haystack

needle が最後に登場する場所を調べたい文字列。

needle

haystack の中で見つけたい文字列。

offset
指定すると、文字列中の任意の文字位置から検索を開始することができます。 負の値を指定すると、文字の終端より前の任意の位置で検索を終了します。
encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

文字列 haystack の中で needle が最後に現れる位置を数字で返します。 needle が見付からなかった場合、**FALSE** を返します。

変更履歴

バージョン	説明

バージョン	説明
5.2.0	オプションのパラメータ *offset* が追加されました。

注意

注意: *encoding* パラメータは、 *PHP 5.2.0* 以降は三番目のパラメータではなく四番目のパラメータに変わりました。 過去との互換性を保つために *encoding* を三番目の引数で指定することもできますが、これは推奨されません。 将来は削除される予定です。

注意: 内部エンコーディングあるいは *mb_regex_encoding()* で指定した文字エンコーディングを、この関数の文字エンコーディングとして使用します。

参考

- *mb_strpos()*
- *mb_internal_encoding()*
- *strrpos()*

---

mb_strstr

(PHP 5 >= 5.2.0)

mb_strstr — 文字列の中で、指定した文字列が最初に現れる位置を見つける

説明

string ***mb_strstr*** *(string $haystack , string $needle [, bool $part [, string $encoding]])*

mb_strstr() は、 *haystack* の中で最初に *needle* が現れる場所を探し、 *haystack* の部分文字列を返します。 *needle* が見つからなかった場合は ***FALSE*** を返します。

パラメータ

haystack

needle が最初に現れる位置を見つける文字列。

needle

haystack の中で探す文字列。

part

この関数が *haystack* のどの部分を返すのかを指定します。 ***TRUE*** の場合は、*haystack* の先頭から *needle* が最初に現れる位置までを返します。 ***FALSE*** の場合は、*needle* が最初に現れる位置から *haystack* の最後までを返します。 デフォルト値は ***FALSE*** です。

encoding

使用する文字エンコーディング名。 省略した場合は内部文字エンコーディングが用いられます。

返り値

haystack の部分文字列を返します。 *needle* が見つからない場合は ***FALSE*** を返します。

参考

- *stristr()*
- *strstr()*
- *mb_stristr()*

---

mb_strtolower

(PHP 4 >= 4.3.0, PHP 5)

mb_strtolower — 文字列を小文字にする

説明

string ***mb_strtolower*** *(string $str [, string $encoding])*

str のすべてのアルファベットを小文字にして返します。

パラメータ

str

小文字にしたい文字列。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

str のすべてのアルファベットを小文字にしたものを返します。

Unicode

Unicode 文字属性についての詳細は » http://www.unicode.org/unicode/reports/tr21/ を参照してください。

strtolower() と違い、アルファベットであるかどうかは Unicode 文字属性をもとに決定されます。したがって、この関数の挙動は ロケールの設定に影響されず、すべてのアルファベット、 例えば A ウムラウト (Ä) を変換することができます。

例

Example#1 mb_strtolower() の例

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtolower($str);
echo $str; // 結果は mary had a little lamb and she loved it so となります
?>
```

参考

- mb_strtoupper()
- mb_convert_case()
- strtolower()

---

mb_strtoupper

(PHP 4 >= 4.3.0, PHP 5)

mb_strtoupper — 文字列を大文字にする

説明

string **mb_strtoupper** (string $str [, string $encoding])

str のすべてのアルファベットを大文字にして返します。

パラメータ

str

大文字に変換したい文字列。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

str のすべてのアルファベットを大文字にしたものを返します。

Unicode

Unicode 文字属性についての詳細は » http://www.unicode.org/unicode/reports/tr21/ を参照してください。

strtoupper()と違い、アルファベットであるかどうかは Unicode 文字属性をもとに決定されます。したがって、この関数の挙動は ロケールの設定に影響されず、すべてのアルファベット、 例えば A ウムラウト (Ä) を変換することができます。

例

Example#1 mb_strtoupper() の例

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtoupper($str);
echo $str; // 結果は MARY HAD A LITTLE LAMB AND SHE LOVED IT SO となります
?>
```

参考

- mb_strtolower()
- mb_convert_case()
- strtoupper()

---

mb_strwidth

(PHP 4 >= 4.0.6, PHP 5)

mb_strwidth — 文字列の幅を返す

説明

int ***mb_strwidth*** *(string $str [, string $encoding])*

文字列 str の幅を返します。

マルチバイト文字は、通常はシングルバイト文字の倍の幅となります。

文字の幅

文字	幅
U+0000 - U+0019	0
U+0020 - U+1FFF	1
U+2000 - U+FF60	2
U+FF61 - U+FF9F	1
U+FFA0 -	2

パラメータ

str

幅を取得したい文字列。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

文字列 str の幅を返します。

参考

- *mb_strimwidth()*
- *mb_internal_encoding()*

---

mb_substitute_character

(PHP 4 >= 4.0.6, PHP 5)

mb_substitute_character — 置換文字を設定あるいは取得する

説明

mixed ***mb_substitute_character*** *([mixed $substrchar])*

入力文字エンコーディングが無効、 または出力文字エンコーディングに文字コードが存在しない場合の代替文字を指定します。 無効な文字は、 ***NULL*** *(出力しない)、文字列または整数値 (Unicode 文字コード値) に置換することが可能です。*

この設定は、mb_convert_encoding()、 mb_convert_variables()、 mb_output_handler()、 および mb_send_mail() に影響します。

パラメータ

substrchar

Unicode 値の整数または文字列を以下のように指定します。

- *"none" : 出力しない*
- *"long" : 文字コードの値 (例: U+3000,JIS+7E7E) を出力する*

返り値

substchar が設定された場合、成功時に ***TRUE****、そうでない場合に* ***FALSE*** *を返します。 substchar が設定されない場合、Unicode 値または "none" あるいは "long" を返します。*

例

Example#1 mb_substitute_character() の例

```
<?php
/* Unicode U+3013 (ゲタ記号)を設定 */
mb_substitute_character(0x3013);

/* HEX フォーマットを設定 */
mb_substitute_character("long");

/* 現在の設定を表示 */
echo mb_substitute_character();
?>
```

---

mb_substr_count

(PHP 4 >= 4.3.0, PHP 5)

mb_substr_count — 部分文字列の出現回数を数える

説明

int **mb_substr_count** (string $haystack , string $needle [, string $encoding])

文字列 haystack の中での部分文字列 needle の出現回数を数えます。

パラメータ

haystack

調べたい文字列。

needle

見つける文字列。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

文字列 haystack の中での部分文字列 needle の出現回数を返します。

例

Example#1 mb_substr_count() の例

```
<?php
echo mb_substr_count("これはてすとです。", "す"); // 2 を出力する
?>
```

参考

- mb_strpos()
- mb_substr()
- substr_count()

---

mb_substr

(PHP 4 >= 4.0.6, PHP 5)

mb_substr — 文字列の一部を得る

説明

string **mb_substr** (string $str , int $start [, int $length [, string $encoding]])

文字数に基づきマルチバイト対応の substr() 処理を行います。位置は、 str の始めから数えられます。 最初の文字の位置は 0、2 番目の文字の位置は 1、といったようになります。

パラメータ

str

調べたい文字列。

start

str の中の使用開始位置。

length

返す文字列の最大の長さ。

encoding

encoding パラメータには文字エンコーディングを指定します。省略した場合は、内部文字エンコーディングを使用します。

返り値

mb_substr() は、start および length パラメータで指定した str の一部を返します。

参考

- mb_strcut()
- mb_internal_encoding()

目次

---

muscat 関数

導入

この拡張モジュールは、Muscat 3.6 検索システムへのインターフェイスを提供します。 Muscat は現在は存在しませんが、その代わりとして » Xapian プロジェクトが使用できます。

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

インストール手順

これらの関数は、PHP が --with-muscat[=DIR] オプションをつけてコンパイルされている際にのみ使用可能です。

リソース型

この拡張モジュールでは Muscat セッションハンドラを定義しています。これは muscat_setup_net() および muscat_setup() が返すものです。

---

muscat_close

(PHP 4 >= 4.0.5)

muscat_close — muscat セッションをシャットダウンする

説明

void **muscat_close** (resource $muscat_handle)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

muscat セッションをシャットダウンし、メモリを PHP に解放します。

パラメータ

muscat_handle

muscat_setup() あるいは muscat_setup_net() が返す muscat セッションハンドラ。

返り値

値を返しません。

参考

- muscat_setup()
- muscat_setup_net()

---

muscat_get

(PHP 4 >= 4.0.5)

muscat_get — コア muscat API から 1 行分のデータを取得する

説明

string **muscat_get** (resource $muscat_handle)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

コア muscat API から 1 行分のデータを取得します。

パラメータ

muscat_handle

muscat_setup() あるいは muscat_setup_net() が返す muscat セッションハンドラ。

返り値

muscat レスポンスを文字列で返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

参考

- muscat_give()

---

muscat_give

(PHP 4 >= 4.0.5)

muscat_give — コア muscat API に文字列を送信する

説明

void **muscat_give** (resource $muscat_handle , string $string)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

string をコア muscat API に送信します。

パラメータ

muscat_handle

muscat_setup() あるいは muscat_setup_net() が返す muscat セッションハンドラ。

string

送信する文字列。

返り値

値を返しません。

参考

- muscat_get()

---

muscat_setup_net

(PHP 4 >= 4.0.5)

muscat_setup_net — 新規 muscat セッションを作成する

説明

resource **muscat_setup_net** (resource $socket)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しい muscat セッションを作成します。

パラメータ

socket

ソケットハンドル。

返り値

セッションハンドラ、あるいはエラー時に **FALSE** を返します。

参考

- muscat_setup()
- muscat_close()

---

muscat_setup

(PHP 4 >= 4.0.5)

muscat_setup — muscat セッションを新規に作成する

説明

resource **muscat_setup** (int $size [, string $muscat_dir])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

新しいローカルの muscat セッションを作成します。

パラメータ

size

muscat 用に確保するメモリサイズです。

muscat_dir

muscat のインストール ディレクトリで、例えば、"/usr/local/empower" となります。 デフォルトは、コンパイル時の muscat ディレクトリで

す。

返り値

セッションハンドラ、あるいはエラー時に **FALSE** を返します。

参考

- muscat_setup_net()
- muscat_close()

目次

MySQL 関数

導入

以下の関数は、MySQL データベースサーバへのアクセスを可能にします。 MySQL に関するより詳細な情報は、» http://www.mysql.com/ にあります。

MySQL のドキュメントは、» http://dev.mysql.com/doc/にあります。

要件

以下の関数を利用可能とするには、MySQL サポートを指定して PHP を コンパイルする必要があります。

インストール手順

コンパイルするには、単純に --with-mysql[=DIR] 設定オプションを利用してください。ここで、オプションの [DIR] は MySQL がインストールされているディレクトリを表します。

この MySQL 拡張モジュールは MySQL 4.1.0 以降とも互換性がありますが、 それ以降のバージョンが提供する新機能はサポートしていません。 この機能を使用するには、 MySQLi 拡張モジュールを使用してください。

もし mysql 拡張モジュールと mysqli 拡張モジュールを同時にインストールしたい場合は、 衝突を避けるために両方で同じクライアントライブラリを用いる必要があります。

Linux へのインストール

PHP 4

オプション --with-mysql はデフォルトで有効となっています。configureオプション --without-mysql を使用すると、 この動作は無効になります。 MySQL インストールディレクトリへのパスを指定しなかった場合、 PHP はバンドルされた MySQL クライアントライブラリを使用します。

(auth-mysqlのような) MySQL を使用する他のアプリケーションを実行するユーザは、 バンドルされたライブラリを使わず --with-mysql=/path/to/mysql のように MySQL のインストールディレクトリを指定するべきです。 これにより、MySQL によりインストールされたクライアントライブラリを PHP が使用するようになり、衝突が回避されます。

PHP 5+

MySQL はデフォルトでは有効とならず、PHP に MySQL ライブラリも付属しません。 この理由の詳細については、 FAQ を参照してください。 ヘッダとライブラリ は、 » MySQL からダウンロードできます。

Windows へのインストール

PHP 4

MySQL 拡張モジュールは、PHP に組み込まれています。

PHP 5+

MySQL はもはやデフォルトでは有効となりません。したがって php.ini で php_mysql.dll DLL を有効にしておく必要があります。 また、PHP が MySQL クライアントライブラリにアクセスできなければなりません。 libmysql.dll というファイルが Windows 版の PHP 配布ファイルに含まれており、PHP が MySQL と話すためにはこのファイルが Windows の PATH にある必要があります。そのための方法については、 "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" という FAQ を参照してください。 libmysql.dll を Windows のシステムディレクトリにコピーしても動作しますが (システムディレクトリは、デフォルトでシステムの PATH に含まれています)、お勧めしません。

(php_mysql.dll も含めた) PHP 拡張モジュールを有効にするには、PHP ディレクティブ extension_dir に拡張モジュールの存在する場所を設定するべきです。 Windows へのマニュアルインストール方法 も参照してください。PHP 5 での extension_dir の例は c:¥php¥ext です。

> **注意**: Web サーバの起動時に以下のようなエラーが発生する場合： "Unable to load dynamic library './php_mysql.dll'" これは php_mysql.dll や libmysql.dll がシステムによって見つけられなかったことが原因です。

MySQL インストールの注意

警告

この拡張モジュールと recode 拡張モジュールを同時にロードした場合、 PHP のクラッシュと起動に関する問題が発生する可能性があります。 より詳細な情報については、recode 拡張モジュールを参照してください。

> **注意**: latin（デフォルト）以外の文字セットを必要とする場合、 使用する文字セットのサポートを有効にしてコンパイルした（バンドル版でない） libmysql をインストールする必要があります。

実行時設定

php.ini の設定により動作が変化します。

MySQL 設定オプション

名前	デフォルト	変更の可否	変更履歴
mysql.allow_persistent	"1"	PHP_INI_SYSTEM	
mysql.max_persistent	"-1"	PHP_INI_SYSTEM	
mysql.max_links	"-1"	PHP_INI_SYSTEM	
mysql.trace_mode	"0"	PHP_INI_ALL	PHP 4.3.0 以降で使用可能です。
mysql.default_port	NULL	PHP_INI_ALL	
mysql.default_socket	NULL	PHP_INI_ALL	PHP 4.0.1 以降で使用可能です。
mysql.default_host	NULL	PHP_INI_ALL	
mysql.default_user	NULL	PHP_INI_ALL	
mysql.default_password	NULL	PHP_INI_ALL	
mysql.connect_timeout	"60"	PHP_INI_ALL	PHP <= 4.3.2 では PHP_INI_SYSTEM で、PHP 4.3.0 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

mysql.allow_persistent boolean

MySQL への 持続的接続 を可能にするかどうか。

mysql.max_persistent integer

プロセス毎の持続的 MySQL 接続の最大数。

mysql.max_links integer

持続的接続を含むプロセス毎の MySQL 接続の最大数。

mysql.trace_mode boolean

トレースモード。mysql.trace_mode が有効の場合、 テーブル/インデックスのスキャン時の警告・結果セットの未開放・SQL エラー などが画面に表示されます（PHP 4.3.0 以降で使用可能です）。

mysql.default_port string

他のポートが指定されない場合、データベースサーバ接続時に使用される デフォルトの TCP ポート番号。デフォルトが指定されない場合は、 環境変数 MYSQL_TCP_PORT・ /etc/services の mysql-tcpエントリ・コンパイル時の **MYSQL_PORT** 定数の順番でポートが取得されます。 Win32 では、**MYSQL_PORT** 定数のみが使用されます。

mysql.default_socket string

他にソケット名が指定されない場合、ローカルなデータベースサーバに 接続する時のデフォルトのソケット名。

mysql.default_host string

他のサーバ名が指定されない場合に、データベースサーバへの接続時に 使用されるデフォルトのサーバ名。 SQL セーフモード では適用されません。

mysql.default_user string

他のユーザ名が指定されない場合に、データベースサーバへの接続時に 使用されるデフォルトのユーザ名。 SQL セーフモード では適用されません。

mysql.default_password string

他のパスワードが指定されない場合に、データベースサーバへの接続時に 使用されるデフォルトのパスワード。 SQL セーフモード では適用されません。

mysql.connect_timeout integer

接続の有効時間(単位:秒)。Linux では、この有効時間はサーバからの 最初の応答の待ち時間としても使用されます。

リソース型

MySQL モジュールでは、2 種類のリソース型が使用されています。 最初のリソースはデータベース接続のリンク ID で、2 番目のリソースは クエリ結果を保持するリソースです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

PHP 4.3.0 以降では、mysql_connect() や mysql_pconnect() で追加のクライアントフラグを 指定できるようになりました。以下の定数が定義されています。

MySQL クライアント定数

定数	説明
MYSQL_CLIENT_COMPRESS	圧縮プロトコルを利用します。
MYSQL_CLIENT_IGNORE_SPACE	関数名の後のスペースを許可します。
MYSQL_CLIENT_INTERACTIVE	interactive_timeout で指定された秒数（wait_timeout のかわり） の無通信が続くまで接続を閉じません。
MYSQL_CLIENT_SSL	SSL による暗号化を使用します。このフラグは、バージョン 4.x 以降の MySQL クライアントライブラリを利用している場合にのみ有効です。 PHP 4 や、Windows 版の PHP 5 にバンドルされているのは、バージョン 3.23.x のライブラリです。

mysql_fetch_array() 関数は、結果の配列の形式を 指定するための定数を使用します。以下の定数が定義されています。

MySQL フェッチ定数

定数	説明
MYSQL_ASSOC	カラムは、フィールド名を添字とする配列形式で返されます。
MYSQL_BOTH	カラムは、数値の添字とフィールド名の添字のどちらでもアクセスできる 配列形式で返されます。
MYSQL_NUM	カラムは、数値の添字を持つ配列形式で返されます。添字は 0 からはじまり、 結果の最初のフィールドです。

注意

注意: ほとんどの MySQL 関数は、link_identifier を 最後のパラメータとしてオプションで受け付けます。これを指定しなかった場合、 直近にオープンされた接続が使用されます。もし直近の接続が存在しない場合、 php.ini のデフォルトパラメータを使用して接続を確立しようとします。 この試行にも失敗した場合、関数は **FALSE** を返します。

例

以下は、MySQL データベースに接続し、クエリを実行し、結果レコードを 出力、接続を切断する例です。

Example#1 MySQL 拡張モジュールに関する例

```
<?php
// データベースに接続し、選択する
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
    or die('Could not connect: ' . mysql_error());
echo 'Connected successfully';
mysql_select_db('my_database') or die('Could not select database');

// SQL クエリを実行する
$query = 'SELECT * FROM my_table';
$result = mysql_query($query) or die('Query failed: ' . mysql_error());

// HTML に結果を出力する
echo "<table>¥n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "¥t<tr>¥n";
    foreach ($line as $col_value) {
        echo "¥t¥t<td>$col_value</td>¥n";
    }
    echo "¥t</tr>¥n";
}
echo "</table>¥n";

// 結果セットを開放する
mysql_free_result($result);

// 接続を閉じる
mysql_close($link);
?>
```

---

mysql_affected_rows

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_affected_rows — 一番最近の操作で変更された行の数を得る

説明

int **mysql_affected_rows** ([resource $link_identifier])

link_identifier と関連付けられた直近の INSERT、UPDATE、REPLACE、DELETE クエリによって変更された行の数を取得します。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に変更された行の数を、直近のクエリが失敗した場合に -1 を返します。

直近のクエリが WHERE 句を含まない DELETE だった場合、テーブルから すべてのレコードが削除されますが、MySQL 4.1.2 以前のバージョンではこの関数はゼロを返します。

UPDATE を使用する場合、MySQL では新旧の値が同じときには更新処理を行いません。 このことから、必ずしも **mysql_affected_rows()** の返す値が マッチする行の数と一致するとは限りません。返す値は実際に更新処理が行われた 行の数です。

REPLACE ステートメントは、まず最初に同じ主キーのレコードを削除した後に 新しいレコードを挿入します。この関数は、削除された行の数と 挿入された行の数を足したものを返します。

例

Example#1 mysql_affected_rows() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('mydb');

/* これは、削除されたレコードの正しい数をかえすはず */
mysql_query('DELETE FROM mytable WHERE id < 10');
printf("Records deleted: %d¥n", mysql_affected_rows());

/* 決して真にはならない where 条件なので、結果は 0 となるはず */
mysql_query('DELETE FROM mytable WHERE 0');
printf("Records deleted: %d¥n", mysql_affected_rows());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Records deleted: 10
Records deleted: 0
```

Example#2 トランザクションを利用した mysql_affected_rows() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('mydb');

/* レコードの更新 */
mysql_query("UPDATE mytable SET used=1 WHERE id < 10");
printf ("Updated records: %d¥n", mysql_affected_rows());
mysql_query("COMMIT");
?>
```

上の例の出力は、たとえば以下のようになります。

```
Updated Records: 10
```

注意

> **注意**: **トランザクション** トランザクションを使用する場合には、コミット後ではなく INSERT, UPDATE, DELETE クエリの後に **mysql_affected_rows()** をコールする必要があります。
>
> **注意**: **SELECT ステートメント** SELECT から返される行の数を得る際には、 mysql_num_rows() が利用できます。

参考

- mysql_num_rows()
- mysql_info()

---

mysql_change_user

(No version information available, might be only in CVS)

mysql_change_user — アクティブな接続でログイン中のユーザーを変更する

説明

int **mysql_change_user** (string $user , string $password [, string $database [, resource $link_identifier]])

mysql_change_user() は、現在アクティブな接続 またはオプションのパラメータ link_identifier が指定された場合にはその接続にログイン中のユーザーを変更します。 database が指定された場合、ユーザーが変更された後 このデータベースがカレントのデータベースとなります。 新規の user/password の組み合わせの認証に失敗した場合、 現在接続中のユーザーがアクティブなままとなります。

この関数は廃止されており、もはや PHP には存在しません。

パラメータ

user

新しい MySQL ユーザー名です。

password

新しい MySQL パスワードです。

database

MySQL データベース名です。指定しなかった場合は、 現在選択されているデータベースが利用されます。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
3.0.14	この関数は PHP から取り除かれました。

注意

> **注意**: **条件** この関数は MySQL 3.23.3 以降が必要です。

参考

- mysql_connect()
- mysql_select_db()
- mysql_query()

---

mysql_client_encoding

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_client_encoding — 文字セット名を返す

説明

string **mysql_client_encoding** ([resource $link_identifier])

MySQL の character_set 変数の値を取得します。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

カレントの接続から、デフォルトの文字セット名を返します。

例

Example#1 mysql_client_encoding() の例

```
<?php
$link     = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$charset  = mysql_client_encoding($link);

echo "カレントの文字セットは: $charset\n";
?>
```

上の例の出力は、たとえば以下のようになります。

```
カレントの文字セットは: latin1
```

参考

- mysql_real_escape_string()

---

mysql_close

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_close — MySQL 接続を閉じる

説明

bool **mysql_close** ([resource $link_identifier])

mysql_close() は、指定した link_identifier が指す MySQL データベースへの非持続的リンクを閉じます。 link_identifier が指定されない場合、最後に オープンされたリンクが使用されます。

持続的でないリンクはスクリプトの実行終了時に自動的に閉じられるの で、通常は **mysql_close()** を使用する必要はありません。 リソースの解放 を参照ください。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_close() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できませんでした: ' . mysql_error());
}
echo '接続に成功しました';
mysql_close($link);
?>
```

上の例の出力は以下となります。

```
接続に成功しました
```

注意

> **注意**: **mysql_close()** は、mysql_pconnect() により生成された持続的リンクを閉じません。

参考

- mysql_connect()
- mysql_free_result()

---

mysql_connect

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_connect — MySQL サーバへの接続をオープンする

説明

resource **mysql_connect** ([string $server [, string $username [, string $password [, bool $new_link [, int $client_flags]]]]])

MySQL サーバへの接続をオープンあるいは再利用します。

パラメータ

server

MySQL サーバ。"hostname:port" のようにポート番号を 指定することが可能で、localhost では ":/path/to/socket" のようにソケットへのパスを指定することも可能です。

PHP ディレクティブ mysql.default_host が指定されない場合（デフォルト）、 'localhost:3306' が使用されます。 SQL セーフモード の場合はこのパラメータは無視され、常に 'localhost:3306' が用いられます。

username

ユーザ名。デフォルト値は mysql.default_user で定義されている値です。SQL セーフモード の場合はこのパラメータは無視され、 サーバプロセスの所有ユーザ名が用いられます。

password

パスワード。デフォルト値は mysql.default_password で定義されている値です。SQL セーフモード の場合はこのパラメータは無視され、 空のパスワードが用いられます。

new_link

同じ引数で 2 回 **mysql_connect()** をコールした場合、 2 回目は新規のリンクが確立されるのではなく、代わりにすでにオープンされた リンクのリンク ID が返されます。 パラメータ new_link はこの動作を変更し、既に **mysql_connect()** が同じパラメータでコールされて いる場合でも常に新規のリンクがオープンされるようにします。 SQL セーフモード の場合はこのパラメータは無視されます。

client_flags

パラメータ client_flags は、 以下の定数の組み合わせです： 128 (LOAD DATA LOCAL の処理を有効にする)、 **MYSQL_CLIENT_SSL**、 **MYSQL_CLIENT_COMPRESS**、 **MYSQL_CLIENT_IGNORE_SPACE** または **MYSQL_CLIENT_INTERACTIVE**。 詳細な情報については MySQL を参照ください。 SQL セーフモード の場合はこのパラメータは無視されます。

返り値

成功した場合に MySQL リンク ID を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	パラメータ client_flags が追加されました。
4.2.0	パラメータ new_link が追加されました。
3.0.10	server に、":/path/to/socket" のサポートが追加されました。
3.0.0	server に、":port" のサポートが追加されました。

例

Example#1 mysql_connect() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できませんでした: ' . mysql_error());
}
echo '接続に成功しました';
mysql_close($link);
?>
```

Example#2 mysql_connect() で hostname:port 構文を使用する例

```
<?php
// example.com のポート 3307 に接続します
$link = mysql_connect('example.com:3307', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できませんでした: ' . mysql_error());
}
echo '接続に成功しました';
mysql_close($link);

// localhost のポート 3307 に接続します
$link = mysql_connect('127.0.0.1:3307', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できませんでした: ' . mysql_error());
}
echo '接続に成功しました';
mysql_close($link);
?>
```

Example#3 mysql_connect() で ":/path/to/socket" 構文を使用する例

```
<?php
// localhost のソケット (例: /tmp/mysql.sock) に接続します

// 方法 1: localhost を省略する
$link = mysql_connect('/tmp/mysql', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できませんでした: ' . mysql_error());
}
echo '接続に成功しました';
mysql_close($link);


// 方法 2: localhost を指定する
$link = mysql_connect('localhost:/tmp/mysql.sock', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できませんでした: ' . mysql_error());
}
echo '接続に成功しました';
mysql_close($link);
?>
```

注意

注意: サーバ名に "localhost" や "localhost:port" を指定した場合、MySQL クライアントライブラリはそれをオーバーライドし、ローカルソケット (Windows では名前つきパイプ) に接続しようとします。 TCP/IP を用いたい場合は、"localhost" のかわりに "127.0.0.1" を用いてください。もし MySQL クライアントライブラリが 間違ったローカルソケットへ接続しようとしている場合、これを修正するには PHP 設定の MySQL に正しいパスを 指定したうえでサーバ名を空白にしておくべきです。

注意: サーバへのリンクは、mysql_close() のコールにより 明示的に閉じられない限り、スクリプトの実行終了と同時に閉じられます。

注意: 関数名の前に @ を付けることで 接続に失敗した場合のエラーメッセージを出力しないようにできます。

注意: エラー "Can't create TCP/IP socket (10106)" が発生するのは、たいていは variables_order 設定ディレクティブに E が含まれていない場合です。 Windows では、これが含まれていなければ SYSTEMROOT 環境変数が使用できず、PHP が Winsock の読み込みに失敗します。

参考

- mysql_pconnect()
- mysql_close()

---

mysql_create_db

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_create_db — MySQL データベースを作成する

説明

bool **mysql_create_db** (string $database_name [, resource $link_identifier])

mysql_create_db() は、指定したリンク ID が指す サーバ上に新規のデータベースを作成します。

パラメータ

database_name

作成されるデータベースの名前です。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_create_db() の別の例

mysql_create_db() は廃止予定です。代わりに mysql_query() を用いて CREATE DATABASE ステートメントを発行する方法が 推奨されます。

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できません: ' . mysql_error());
}

$sql = 'CREATE DATABASE my_db';
if (mysql_query($sql, $link)) {
    echo "データベース my_db の作成に成功しました\n";
} else {
    echo 'データベースの作成に失敗しました: ' . mysql_error() . "¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
データベース my_db の作成に成功しました
```

注意

> **注意**: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_createdb()**
>
> **注意**: MySQL 4.x クライアントライブラリを用いて MySQL 拡張モジュールを ビルドした場合、この関数は利用できません。

参考

- mysql_query()
- mysql_select_db()

---

mysql_data_seek

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_data_seek — 内部的な結果ポインタを移動する

説明

bool **mysql_data_seek** (resource $result , int $row_number)

mysql_data_seek() は、指定した結果 ID (result_identifier) が指す MySQL 結果の内部ポインタが指定した行番号 (row_number) を指すように移動します。 この後、たとえば mysql_fetch_assoc() のような MySQL のフェッチ関数をコールした場合には、 ここで指定した行の内容が返されます。

row_number は 0 から始まります。 row_number は 0 から mysql_num_rows() - 1 までの範囲にあるべきです。 しかし、もし結果セットが空 (mysql_num_rows() == 0) の場合、0 へのシークは E_WARNING を発生して失敗し、**mysql_data_seek()** は **FALSE** を返します。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

row_number

新しく結果ポインタを設定したい行番号です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_data_seek() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('接続できませんでした: ' . mysql_error());
}
$db_selected = mysql_select_db('sample_db');
if (!$db_selected) {
    die('データベースを選択できませんでした: ' . mysql_error());
}
$query = 'SELECT last_name, first_name FROM friends';
$result = mysql_query($query);
if (!$result) {
    die('クエリは失敗しました: ' . mysql_error());
}
/* 行を逆順で取得する */
for ($i = mysql_num_rows($result) - 1; $i >= 0; $i--) {
    if (!mysql_data_seek($result, $i)) {
        echo "行 $i をシークできません: " . mysql_error() . "¥n";
        continue;
    }

    if (!($row = mysql_fetch_assoc($result))) {
        continue;
    }

    echo $row['last_name'] . ' ' . $row['first_name'] . "<br />¥n";
}

mysql_free_result($result);
?>
```

注意

注意: **mysql_data_seek()** は、 mysql_query() との組み合わせでのみ利用可能です。 mysql_unbuffered_query() と組み合わせることはできません。

参考

- mysql_query()
- mysql_num_rows()
- mysql_fetch_row()
- mysql_fetch_assoc()
- mysql_fetch_array()
- mysql_fetch_object()

---

mysql_db_name

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_db_name — データベース名を得る

説明

string **mysql_db_name** (resource $result , int $row [, mixed $field])

mysql_list_dbs() をコールした結果からデータベース名を取得します。

パラメータ

result

mysql_list_dbs() をコールして 得られた結果ポインタ。

row

結果セット内のインデックス。

field

フィールド名。

返り値

成功した場合にデータベース名を、失敗した場合に ***FALSE*** *を返します。* ***FALSE*** *が返された場合、エラーの発生源を特定するには* *mysql_error()* *を使用してください。*

例

Example#1 mysql_db_name() の例

```
<?php
error_reporting(E_ALL);

$link = mysql_connect('dbhost', 'username', 'password');
$db_list = mysql_list_dbs($link);

$i = 0;
$cnt = mysql_num_rows($db_list);
while ($i < $cnt) {
    echo mysql_db_name($db_list, $i) . "¥n";
    $i++;
}
?>
```

注意

> ***注意***: *下位互換のために、次の非推奨別名を使用してもいいでしょう。* ***mysql_dbname()***

参考

- *mysql_list_dbs()*
- *mysql_tablename()*

---

mysql_db_query

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_db_query — MySQL クエリーを送信する

説明

resource ***mysql_db_query*** *(string $database , string $query [, resource $link_identifier])*

mysql_db_query() *はデータベースを選択し、 そこでクエリーを実行します。*

パラメータ

database

選択するデータベース名。

query

MySQL クエリー。

link_identifier

*MySQL 接続。指定されない場合、*mysql_connect()* により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに* mysql_connect() *がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、* ***E_WARNING*** *レベルの警告が生成されます。*

返り値

クエリーの結果を指す MySQL 結果リソースを正の値で返します。 エラー時には ***FALSE*** *を返します。また、 INSERT/UPDATE/DELETE クエリーの場合には* ***TRUE*** *または* ***FALSE*** *を返し、これらはそれぞれ クエリーが 成功した / 失敗した ことを示します。*

変更履歴

バージョン	説明
4.0.6	この関数は廃止予定です。使用しないでください。かわりに mysql_select_db() あるいは mysql_query() を使用してください。

例

Example#1 mysql_db_query() の別の例

```
<?php

if (!$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')) {
    echo 'Could not connect to mysql';
    exit;
}

if (!mysql_select_db('mysql_dbname', $link)) {
    echo 'Could not select database';
```

```
    exit;
}

$sql    = 'SELECT foo FROM bar WHERE id = 42';
$result = mysql_query($sql, $link);

if (!$result) {
    echo "DB Error, could not query the database¥n";
    echo 'MySQL Error: ' . mysql_error();
    exit;
}

while ($row = mysql_fetch_assoc($result)) {
    echo $row['foo'];
}

mysql_free_result($result);

?>
```

注意

注意: この関数の終了後、直前に選択されていたデータベースに自動的に戻ることは ないということに注意してください。 言い換えれば、一時的に別のデータベース上でクエリーを 実行するという目的でこの関数を利用することはできないということです。 もしそのような場合は、もとのデータベースに改めて接続しなおす必要があります。 この関数のかわりに、SQL クエリーの中で database.table 構文を用いたり mysql_select_db() を利用したりすることが 強く推奨されています。

参考

- mysql_query()
- mysql_select_db()

---

mysql_drop_db

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_drop_db — MySQLデータベースを破棄(削除)する

説明

bool **mysql_drop_db** (string $database_name [, resource $link_identifier])

mysql_drop_db() は、指定した link_identifier が指す データベース全体をサーバから破棄(削除)しようとします。 この関数は非推奨です。かわりに、 mysql_query() を用いて DROP DATABASE 文を発行する方法が推奨されます。

パラメータ

database_name

削除するデータベースの名前。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_drop_db() の別の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}

$sql = 'DROP DATABASE my_db';
if (mysql_query($sql, $link)) {
    echo "Database my_db was successfully dropped¥n";
} else {
    echo 'Error dropping database: ' . mysql_error() . "¥n";
}
?>
```

注意

警告

MySQL 4.x のクライアントライブラリを用いて MySQL 拡張モジュールを ビルドした場合、この関数は利用できません。

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_dropdb()**

参考

- mysql_query()

---

mysql_errno

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_errno — 直近の MySQL 処理からエラーメッセージのエラー番号を返す

説明

int **mysql_errno** ([resource $link_identifier])

直近の MySQL 関数で発生したエラーの番号を返します。

MySQL データベースバックエンドから返ってくるエラーは、 警告を発生しません。代わりに **mysql_errno()** を用いて エラー番号を取得してください。この関数が返すのは、直近に実行された MySQL 関数 (mysql_error() と **mysql_errno()** は除く) のエラーコードだけであることに 注意しましょう。この関数を利用するなら、別の MySQL 関数をコールする前に 値を調べることを忘れないようにしましょう。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

直近の MySQL 関数からのエラー番号を返します。エラーが発生していない 場合は、0 (ゼロ)を返します。

例

Example#1 mysql_errno() の例

```
<?php
$link = mysql_connect("localhost", "mysql_user", "mysql_password");

if (!mysql_select_db("nonexistentdb", $link)) {
    echo mysql_errno($link) . ": " . mysql_error($link). "¥n";
}

mysql_select_db("kossu", $link);
if (!mysql_query("SELECT * FROM nonexistenttable", $link)) {
    echo mysql_errno($link) . ": " . mysql_error($link) . "¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

参考

- mysql_error()
- » MySQL エラーコード

---

mysql_error

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_error — 直近に実行された MySQL 操作のエラーメッセージを返す

説明

string **mysql_error** ([resource $link_identifier])

直近の MySQL 関数からのエラー文字列を返します。 MySQL データベースバックエンドから返ってくるエラーは、 警告を発生しません。代わりに **mysql_error()** を用いて エラー文字列を取得してください。この関数が返すのは、直近に実行された MySQL 関数 (**mysql_error()** と mysql_errno() は除く) のエラー文字列だけであることに 注意しましょう。この関数を利用するなら、別の MySQL 関数をコールする前に 値を調べることを忘れないようにしましょう。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

直近の MySQL 関数からのエラー文字列を返します。エラーが発生していない 場合には、''（空文字列）を返します。

例

Example#1 mysql_error() の例

```
<?php
$link = mysql_connect("localhost", "mysql_user", "mysql_password");

mysql_select_db("nonexistentdb", $link);
echo mysql_errno($link) . ": " . mysql_error($link). "¥n";

mysql_select_db("kossu", $link);
mysql_query("SELECT * FROM nonexistenttable", $link);
echo mysql_errno($link) . ": " . mysql_error($link) . "¥n";
?>
```

上の例の出力は、たとえば以下のようになります。

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

参考

- mysql_errno()
- » MySQL エラーコード

---

mysql_escape_string

(PHP 4 >= 4.0.3, PHP 5, PECL mysql:1.0)

mysql_escape_string — mysql_query で使用するために文字列をエスケープする

説明

```
string mysql_escape_string ( string $unescaped_string )
```

この関数は、mysql_query() で指定可能なように unescaped_string をエスケープします。 この関数は非推奨です。

この関数は mysql_real_escape_string() とほぼ同じです。ただ mysql_real_escape_string() はコネクションハンドラを用い、カレントの文字セットを考慮したエスケープを行うという点が違います。 **mysql_escape_string()** はコネクションに関する引数を 持たず、カレントの文字セット設定を考慮しません。

パラメータ

unescaped_string

エスケープされる文字列。

返り値

エスケープされた文字列を返します。

変更履歴

バージョン	説明
4.3.0	この関数は非推奨となりました。利用しないでください。代わりに mysql_real_escape_string() を利用してください。

例

Example#1 mysql_escape_string() の例

```
<?php
$item = "Zak's Laptop";
$escaped_item = mysql_escape_string($item);
printf("Escaped string: %s¥n", $escaped_item);
?>
```

上の例の出力は以下となります。

```
Escaped string: Zak¥'s Laptop
```

注意

注意: **mysql_escape_string()** は、 % および _ をエスケープしません。

参考

- mysql_real_escape_string()
- addslashes()

- magic_quotes_gpc ディレクティブ

---

mysql_fetch_array

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_fetch_array — 連想配列、添字配列、またはその両方として結果の行を取得する

説明

array **mysql_fetch_array** (resource $result [, int $result_type])

取得した行に対応する配列を返し、内部のデータポインタを前に進めます。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

result_type

取得する配列の形式です。以下の定数値をとります。: **MYSQL_ASSOC**, **MYSQL_NUM**, そしてデフォルト値である **MYSQL_BOTH**

返り値

取得した行をあらわす文字列の配列を返します。もし行が存在しない場合は **FALSE** を返します。返される配列の形式は、result_type がどのように指定されているかによります。**MYSQL_BOTH**（デフォルト） を利用すると、連想添字と数値添字を共に持つ配列を取得します。 **MYSQL_ASSOC** を利用すると（ **mysql_fetch_assoc()** の動作と同様に）連想添字のみが取得され、 **MYSQL_NUM** を利用すると （mysql_fetch_row() の動作と同様に）数値添字のみが 取得されます。

結果の中で同じフィールド名のカラムが 2 つ以上ある場合、 最後のカラムが優先されます。 同名の他のカラムにアクセスするには、そのカラムの数値インデックスを 使うかまたはカラムの別名を定義する必要があります。 カラムの別名を定義した場合は、本来の列名でそのカラムにアクセスすることは できません。

例

Example#1 重複した列名に対して別名を定義する問い合わせ

```
SELECT table1.field AS foo, table2.field AS bar FROM table1, table2
```

Example#2 mysql_fetch_array() を MYSQL_NUM とともに利用する

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_NUM)) {
    printf("ID: %s  Name: %s", $row[0], $row[1]);
}

mysql_free_result($result);
?>
```

Example#3 mysql_fetch_array() を MYSQL_ASSOC とともに利用する

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    printf("ID: %s  Name: %s", $row["id"], $row["name"]);
}

mysql_free_result($result);
?>
```

Example#4 mysql_fetch_array() を MYSQL_BOTH とともに利用する

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
    die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
    printf ("ID: %s  Name: %s", $row[0], $row["name"]);
}

mysql_free_result($result);
?>
```

注意

注意: **パフォーマンス** 特筆すべき点として、**mysql_fetch_array()** が 著しい付加価値があるにもかかわらず、 mysql_fetch_row()

より それほど遅くはないということが言えます。

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- mysql_fetch_row()
- mysql_fetch_assoc()
- mysql_data_seek()
- mysql_query()

---

mysql_fetch_assoc

(PHP 4 >= 4.0.3, PHP 5, PECL mysql:1.0)

mysql_fetch_assoc — 連想配列として結果の行を取得する

説明

array **mysql_fetch_assoc** (resource $result)

取得した行に対応する連想配列を返し、内部のデータポインタを前に進めます。 **mysql_fetch_assoc()** は、 mysql_fetch_array() の 2 番目のパラメータを MYSQL_ASSOC に指定してコールするのと同じ働きをします。つまり、 連想配列のみを返します。

パラメータ

result

評価された結果 リソース。この結果は、 mysql_query() のコールにより得られたものです。

返り値

取得した行に対応する文字列の連想配列を返します。行がもうない場合には **FALSE** を返します。

結果の複数のカラムが同じフィールド名を有している場合、最後のカラ ムが優先されます。同じ名前を有する他のカラムにアクセスするには、 mysql_fetch_row()を使用して数値添字を返すか、 エイリアス名を追加する必要があります。エイリアスの説明については、 mysql_fetch_array()の例を参照ください。

例

Example#1 mysql_fetch_assoc() のいろいろな例

```
<?php

$conn = mysql_connect("localhost", "mysql_user", "mysql_password");

if (!$conn) {
    echo "Unable to connect to DB: " . mysql_error();
    exit;
}

if (!mysql_select_db("mydbname")) {
    echo "Unable to select mydbname: " . mysql_error();
    exit;
}

$sql = "SELECT id as userid, fullname, userstatus
        FROM   sometable
        WHERE  userstatus = 1";

$result = mysql_query($sql);

if (!$result) {
    echo "Could not successfully run query ($sql) from DB: " . mysql_error();
    exit;
}

if (mysql_num_rows($result) == 0) {
    echo "No rows found, nothing to print so am exiting";
    exit;
}

// データ行が存在する間、それを $row に連想配列形式でセットする
// 注: 結果が 1 行であることがわかっているのなら、ループを利用しなくてもよい
// 注: ループ内で extract($row); を実行すれば、
//       $userid, $fullname, そして $userstatus を利用できる
while ($row = mysql_fetch_assoc($result)) {
    echo $row["userid"];
    echo $row["fullname"];
    echo $row["userstatus"];
}

mysql_free_result($result);

?>
```

注意

注意: **パフォーマンス** 特筆すべき点として、**mysql_fetch_assoc()** が 著しい付加価値があるにもかかわらず、 mysql_fetch_row() より それほど遅くはないということが言えます。

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- mysql_fetch_row()
- mysql_fetch_array()
- mysql_data_seek()
- mysql_query()
- mysql_error()

---

mysql_fetch_field

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_fetch_field — 結果からカラム情報を取得し、オブジェクトとして返す

説明

object **mysql_fetch_field** (resource $result [, int $field_offset])

フィールド情報を含むオブジェクトを返します。特定のクエリー結果の中の フィールドに関する情報を得るために使用可能です。

パラメータ

result

評価された結果 リソース。この結果は、 mysql_query() のコールにより得られたものです。

field_offset

数字で表したフィールドの位置です。もし指定されなければ、 まだこの関数で情報を取得していないフィールドのうち最初のものが 選択されます。 field_offset は、 0 からはじまります。

返り値

フィールド情報を含むobjectを返します。オブジェクトの プロパティは次のとおりです。

- name - カラム名
- table - カラムが属しているテーブルの名前
- def - カラムのデフォルト値
- max_length - カラムの最大長
- not_null - カラムが **NULL** 値をとることができない場合 1
- primary_key - カラムが主キーであれば 1
- unique_key - カラムがユニークキーであれば 1
- multiple_key - カラムが非ユニークキーであれば 1
- numeric - カラムが数値(numeric)であれば 1
- blob - カラムがBLOBであれば 1
- type - カラムの型
- unsigned - カラムが符号無し(unsigned)であれば 1
- zerofill - カラムがゼロで埋められている(zero-filled)場合に 1

例

Example#1 mysql_fetch_field() の例

```
<?php
$conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$conn) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('database');
$result = mysql_query('select * from table');
if (!$result) {
    die('Query failed: ' . mysql_error());
}
/* カラムのメタデータを取得する */
$i = 0;
while ($i < mysql_num_fields($result)) {
    echo "Information for column $i:<br />¥n";
    $meta = mysql_fetch_field($result, $i);
    if (!$meta) {
        echo "No information available<br />¥n";
    }
    echo "<pre>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
```

```
not_null:       $meta->not_null
numeric:        $meta->numeric
primary_key:    $meta->primary_key
table:          $meta->table
type:           $meta->type
default:        $meta->def
unique_key:     $meta->unique_key
unsigned:       $meta->unsigned
zerofill:       $meta->zerofill
</pre>";
    $i++;
}
mysql_free_result($result);
?>
```

注意

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

参考

- mysql_field_seek()

---

mysql_fetch_lengths

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_fetch_lengths — 結果における各出力の長さを得る

説明

array **mysql_fetch_lengths** (resource $result)

MySQL により一番最近に取得された行における各フィールドの長さを 格納した配列を返します。

mysql_fetch_lengths()は、 mysql_fetch_row(), mysql_fetch_assoc(), mysql_fetch_array(), そして mysql_fetch_object() により一番最近に返された 各結果カラムの長さを格納した配列を返します。この配列のオフセットは 0 から始まります。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

返り値

成功した場合に長さの配列 (array) を、 失敗した場合に **FALSE** を返します。

例

Example#1 mysql_fetch_lengths() の例

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
$row      = mysql_fetch_assoc($result);
$lengths  = mysql_fetch_lengths($result);

print_r($row);
print_r($lengths);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [id] => 42
    [email] => user@example.com
)
Array
(
    [0] => 2
    [1] => 16
)
```

参考

- mysql_field_len()
- mysql_fetch_row()
- strlen()

---

mysql_fetch_object

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_fetch_object — 結果の行をオブジェクトとして取得する

説明

object **mysql_fetch_object** (resource $result [, string $class_name [, array $params]])

取得された行を表すプロパティを有するオブジェクトを返し、 内部のデータポインタを前に進めます。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

class_name

インスタンス化し、プロパティを設定して返すクラスの名前。 指定しなかった場合は stdClass オブジェクトが返されます。

params

class_name オブジェクトのコンストラクタに渡す オプションのパラメータの配列。

返り値

取得された行を表す文字列プロパティを有するオブジェクト(object)を返します。 もう行が残っていない場合は、**FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	さまざまな型のオブジェクトで結果を返せるようになりました。

例

Example#1 mysql_fetch_object() の例

```
<?php
mysql_connect("hostname", "user", "password");
mysql_select_db("mydb");
$result = mysql_query("select * from mytable");
while ($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

Example#2 mysql_fetch_object() の例

```
<?php

$row = mysql_fetch_object($result);

/* これが正しい方法 */
echo $row->field;
/* これは間違い */
// echo $row->0;

?>
```

注意

> **注意**: **パフォーマンス** 速度面では、この関数は mysql_fetch_array() と同等で、 mysql_fetch_row() とほぼ同等です(違いはわずかです)。
>
> **注意**: **mysql_fetch_object()**は、配列の代わりに オブジェクトが返されるという一つの違いを除いて mysql_fetch_array()と類似しています。 つまり、オフセットによってではなく、フィールド名によってのみ データにアクセスすることができます (数字は、プロパティ名として使用できません)。
>
> **注意**: この関数により返されるフィールド名は 大文字小文字を区別 します。
>
> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- mysql_fetch_array()
- mysql_fetch_assoc()
- mysql_fetch_row()
- mysql_data_seek()
- mysql_query()

---

mysql_fetch_row

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_fetch_row — 結果を添字配列として取得する

説明

array **mysql_fetch_row** (resource $result)

取得された行に対応する配列を返し、内部のデータポインタを前に進めます。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

返り値

取得された行に対応する文字列の配列を返します。もう行がない場合は、 **FALSE** を返します。

mysql_fetch_row()は、指定した結果 ID が指す結果から 1 行分のデータを取得します。 各結果カラムは、オフセット 0 から始まる配列に格納されます。

例

Example#1 mysql_fetch_row() で行を取得する

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
$row = mysql_fetch_row($result);

echo $row[0]; // 42
echo $row[1]; // 'email' の値
?>
```

注意

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- mysql_fetch_array()
- mysql_fetch_assoc()
- **mysql_data_object()**
- mysql_data_seek()
- mysql_fetch_lengths()
- mysql_result()

---

mysql_field_flags

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_field_flags — 結果において指定したフィールドのフラグを取得する

説明

string **mysql_field_flags** (resource $result , int $field_offset)

mysql_field_flags()は、指定したフィールドの フィールドフラグを返します。個々のフラグは、空白一つで区切られた 形式で返されます。このため、返された値を explode() で分割することができます。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は 0 から始まります。 field_offset が存在しない場合、 **E_WARNING** レベルのエラーが発行されます。

返り値

結果についてのフラグを文字列で返します。失敗した場合に **FALSE** を返します。

運用システム上のMySQLがサポートしている場合、次のフラグがレポート されます。 "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment" そして "timestamp"

例

Example#1 mysql_field_flags() の例

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
$flags = mysql_field_flags($result, 0);

echo $flags;
print_r(explode(' ', $flags));
?>
```

上の例の出力は、たとえば以下のようになります。

```
not_null primary_key auto_increment
Array
(
    [0] => not_null
    [1] => primary_key
    [2] => auto_increment
)
```

注意

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_fieldflags()**

参考

- mysql_field_type()
- mysql_field_len()

---

mysql_field_len

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_field_len — 指定したフィールドの長さを返す

説明

```
int mysql_field_len ( resource $result , int $field_offset )
```

mysql_field_len()は指定したフィールドの長さを 返します。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は 0 から始まります。 field_offset が存在しない場合、 **E_WARNING** レベルのエラーが発行されます。

返り値

成功した場合には指定したフィールドの長さ、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_field_len() の例

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}

// データベーススキーマで定義されている、id フィールドの
// 長さを取得する
$length = mysql_field_len($result, 0);
echo $length;
?>
```

注意

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_fieldlen()**

参考

- mysql_fetch_lengths()
- strlen()

---

mysql_field_name

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_field_name — 結果において指定したフィールド名を取得する

説明

string **mysql_field_name** (resource $result , int $field_offset)

mysql_field_name()は、指定したフィールドの 名前を返します。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は 0 から始まります。 field_offset が存在しない場合、 **E_WARNING** レベルのエラーが発行されます。

返り値

成功した場合に指定したフィールドの名前を、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_field_name() の例

```
<?php
/* users テーブルには以下の 3 つのフィールドがある
*    user_id
*    username
*    password.
*/
$link = @mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect to MySQL server: ' . mysql_error());
}
$dbname = 'mydb';
$db_selected = mysql_select_db($dbname, $link);
if (!$db_selected) {
    die("Could not set $dbname: " . mysql_error());
}
$res = mysql_query('select * from users', $link);

echo mysql_field_name($res, 0) . "¥n";
echo mysql_field_name($res, 2);
?>
```

上の例の出力は以下となります。

```
user_id
password
```

注意

> **注意**: この関数により返されるフィールド名は 大文字小文字を区別 します。
>
> **注意**: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_fieldname()**

参考

- mysql_field_type()
- mysql_field_len()

---

mysql_field_seek

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_field_seek — 結果ポインタを指定したフィールドオフセットにセットする

説明

bool **mysql_field_seek** (resource $result , int $field_offset)

指定したフィールドオフセットに移動します。 **mysql_field_seek()** をコールした後、 mysql_fetch_field() をフィールドオフセットを付けずに コールした場合、このフィールドが返されます。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は 0 から始まります。 field_offset が存在しない場合、 E_WARNING レベルのエラーが発行されます。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- mysql_fetch_field()

---

mysql_field_table

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_field_table — 指定したフィールドが含まれるテーブルの名前を取得する

説明

string **mysql_field_table** (resource $result , int $field_offset)

指定したフィールドが含まれるテーブルの名前を返します。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は 0 から始まります。 field_offset が存在しない場合、 E_WARNING レベルのエラーが発行されます。

返り値

成功した場合にテーブルの名前を返します。

例

Example#1 mysql_field_table() の例

```
<?php

$query = "SELECT account.*, country.* FROM account, country WHERE country.name = 'Portugal' AND account.country_id = cou

// 結果を DB から取得します
$result = mysql_query($query);

// テーブル名とフィールド名を一覧表示します
for ($i = 0; $i < mysql_num_fields($result); ++$i) {
    $table = mysql_field_table($result, $i);
    $field = mysql_field_name($result, $i);

    echo "$table: $field¥n";
}

?>
```

注意

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_fieldtable()**

参考

- mysql_list_tables()

---

mysql_field_type

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_field_type — 結果において指定したフィールドの型を取得する

説明

string **mysql_field_type** (resource $result , int $field_offset)

mysql_field_type() は、 mysql_field_name() 関数に似ています。 引数は同じですが、この関数ではフィールドの型が返されます。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

field_offset

数値フィールドオフセット。 field_offset は 0 から始まります。 field_offset が存在しない場合、 E_WARNING レベルのエラーが発行されます。

返り値

返されるフィールド型は "int", "real", "string", "blob", そして その他 » MySQL ドキュメント で詳細が規定されている型のうちのひとつになります。

例

Example#1 mysql_field_type() の例

```
<?php
mysql_connect("localhost", "mysql_username", "mysql_password");
mysql_select_db("mysql");
$result = mysql_query("SELECT * FROM func");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$table  = mysql_field_table($result, 0);
echo "Your '" . $table . "' table has " . $fields . " fields and " . $rows . " record(s)¥n";
echo "The table has the following fields:¥n";
for ($i=0; $i < $fields; $i++) {
    $type  = mysql_field_type($result, $i);
    $name  = mysql_field_name($result, $i);
    $len   = mysql_field_len($result, $i);
    $flags = mysql_field_flags($result, $i);
    echo $type . " " . $name . " " . $len . " " . $flags . "¥n";
}
mysql_free_result($result);
mysql_close();
?>
```

上の例の出力は、たとえば以下のようになります。

```
Your 'func' table has 4 fields and 1 record(s)
The table has the following fields:
string name 64 not_null primary_key binary
int ret 1 not_null
string dl 128 not_null
string type 9 not_null enum
```

注意

> **注意**: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_fieldtype()**

参考

- mysql_field_name()
- mysql_field_len()

---

mysql_free_result

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_free_result — 結果保持用メモリを開放する

説明

```
bool mysql_free_result ( resource $result )
```

mysql_free_result() は、結果 ID result に関するすべてのメモリを開放します。

mysql_free_result() は、 スクリプト実行のメモリの使用量が多すぎると懸念される場合にのみ 必要になります。指定した結果 ID に関する全ての結果保持用メモリは、 スクリプトの実行後に自動的に開放されます。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

result がリソースではなかった場合、 E_WARNING レベルのエラーが発生します。 mysql_query() が resource を返すのは SELECT, SHOW, EXPLAIN, そして DESCRIBE の場合だけであることに注意しましょう。

例

Example#1 mysql_free_result() の例

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}
/* 結果を利用する。ここでは、その後結果を利用したものと仮定する */
$row = mysql_fetch_assoc($result);

/* 結果を開放し、さらにスクリプトの処理を進める */
mysql_free_result($result);

echo $row['id'];
echo $row['email'];
?>
```

注意

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_freeresult()**

参考

- mysql_query()
- is_resource()

---

mysql_get_client_info

(PHP 4 >= 4.0.5, PHP 5, PECL mysql:1.0)

mysql_get_client_info — MySQL クライアント情報を取得する

説明

string **mysql_get_client_info** (void)

mysql_get_client_info() は、 クライアントライブラリのバージョンを表す文字列を返します。

返り値

MySQL クライアントのバージョンを返します。

例

Example#1 mysql_get_client_info() の例

```
<?php
printf("MySQL client info: %s¥n", mysql_get_client_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
MySQL client info: 3.23.39
```

参考

- mysql_get_host_info()
- mysql_get_proto_info()
- mysql_get_server_info()

---

mysql_get_host_info

(PHP 4 >= 4.0.5, PHP 5, PECL mysql:1.0)

mysql_get_host_info — MySQL ホスト情報を取得する

説明

string **mysql_get_host_info** ([resource $link_identifier])

使用されている接続の型を返します。その中にはサーバのホスト名も含まれます。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

使用されている MySQL 接続の型を表す文字列を返します。失敗した場合に **FALSE** を返します。

例

Example#1 mysql_get_host_info() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
printf("MySQL host info: %s¥n", mysql_get_host_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
MySQL host info: Localhost via UNIX socket
```

参考

- mysql_get_client_info()
- mysql_get_proto_info()
- mysql_get_server_info()

---

mysql_get_proto_info

(PHP 4 >= 4.0.5, PHP 5, PECL mysql:1.0)

mysql_get_proto_info — MySQL プロトコル情報を取得する

説明

int **mysql_get_proto_info** ([resource $link_identifier])

MySQL プロトコルを取得します。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に MySQL プロトコル、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_get_proto_info() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
printf("MySQL protocol version: %s¥n", mysql_get_proto_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
MySQL protocol version: 10
```

参考

- mysql_get_client_info()
- mysql_get_host_info()
- mysql_get_server_info()

---

mysql_get_server_info

(PHP 4 >= 4.0.5, PHP 5, PECL mysql:1.0)

mysql_get_server_info — MySQL サーバ情報を取得する

説明

```
string mysql_get_server_info ([ resource $link_identifier ] )
```

MySQL サーバのバージョンを取得します。

パラメータ

`link_identifier`

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、E_WARNING レベルの警告が生成されます。

返り値

成功した場合に MySQL サーバのバージョン、失敗した場合に FALSE を返します。

例

Example#1 mysql_get_server_info() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
printf("MySQL server version: %s¥n", mysql_get_server_info());
?>
```

上の例の出力は、たとえば以下のようになります。

```
MySQL server version: 4.0.1-alpha
```

参考

- mysql_get_client_info()
- mysql_get_host_info()
- mysql_get_proto_info()
- phpversion()

---

mysql_info

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_info — 直近のクエリについての情報を得る

説明

```
string mysql_info ([ resource $link_identifier ] )
```

直近のクエリについての詳細な情報を返します。

パラメータ

`link_identifier`

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、E_WARNING レベルの警告が生成されます。

返り値

成功した場合に文についての情報、失敗した場合に FALSE を返します。 どんな文が情報を返し、またそれがどのように見えるのかについては、以下の 例を参照ください。ここに挙げられていない文では FALSE が返されます。

例

Example#1 該当する MySQL 文

情報を返す文の例です。数値はあくまで説明用の例で、実際の値は クエリの内容によって変わります。

```
INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0
```

注意

注意: INSERT ... VALUES 文で **mysql_info()** が非 **FALSE** 値を 返すのは、文中で複数の値のリストが指定された場合に限ります。

参考

- mysql_affected_rows()
- mysql_insert_id()
- mysql_stat()

---

mysql_insert_id

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_insert_id — 直近の INSERT 操作で生成された ID を得る

説明

int **mysql_insert_id** ([resource $link_identifier])

直近の INSERT クエリにより AUTO_INCREMENT カラム用に生成された ID を取得します。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

直近の INSERT クエリにより AUTO_INCREMENT カラム用に生成された ID を返します。直近のクエリが AUTO_INCREMENT な値を生成しなかった場合に 0、MySQL 接続が確立されていなかった 場合に **FALSE** を返します。

例

Example#1 mysql_insert_id() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('mydb');

mysql_query("INSERT INTO mytable (product) values ('kossu')");
printf("最後に挿入されたレコードの ID は、%d\n", mysql_insert_id());
?>
```

注意

警告

mysql_insert_id() は、MySQL C API 関数 mysql_insert_id() の返り値の型を long 型 (PHP では int と呼ばれる) に変換します。AUTO_INCREMENT カラムが BIGINT 型である場合、 **mysql_insert_id()** で返される型は不正確になります。 かわりに、MySQL の SQL 内部関数 LAST_INSERT_ID() を使用してください。

注意: **mysql_insert_id()** は直近のクエリに対して働くので、 値を生成したクエリの直後に **mysql_insert_id()** をコールすることを忘れないようにしてください。

注意: MySQL の SQL 関数 LAST_INSERT_ID() の値には、 常に直近の AUTO_INCREMENT の値が含まれており、クエリの間では リセットされません。

参考

- mysql_query()
- mysql_info()

---

mysql_list_dbs

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_list_dbs — MySQL サーバ上で利用可能なデータベースのリストを得る

説明

resource **mysql_list_dbs** ([resource $link_identifier])

カレントの mysql デーモンから、利用可能なデータベースを含む 結果ポインタを返します。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、E_WARNING レベルの警告が生成されます。

返り値

成功した場合に結果ポインタ resource を、失敗した場合に FALSE を返します。結果ポインタの中身を調べるために mysql_tablename() 関数を利用し、取得したテーブルを 利用するには mysql_fetch_array() などの関数を 利用してください。

例

Example#1 mysql_list_dbs() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
     echo $row->Database . "¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
database1
database2
database3
```

注意

> **注意**: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_listdbs()**

参考

- mysql_db_name()
- mysql_select_db()

---

mysql_list_fields

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_list_fields — MySQL テーブルのフィールドのリストを得る

説明

resource **mysql_list_fields** (string $database_name , string $table_name [, resource $link_identifier])

指定された名前のテーブルについての情報を取得します。

この関数は非推奨です。かわりに mysql_query() を利用して SHOW COLUMNS FROM table [LIKE 'name'] 文を発行することを推奨します。

パラメータ

database_name

クエリの対象となるデータベース名。

table_name

クエリの対象となるテーブル名。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、E_WARNING レベルの警告が生成されます。

返り値

成功した場合に結果ポインタ resource 、失敗した場合に FALSE を返します。

返された結果は mysql_field_flags(), mysql_field_len(), mysql_field_name() そして mysql_field_type() で利用可能です。

例

Example#1 非推奨である mysql_list_fields() の代替例

```
<?php
$result = mysql_query("SHOW COLUMNS FROM sometable");
if (!$result) {
     echo 'Could not run query: ' . mysql_error();
     exit;
}
if (mysql_num_rows($result) > 0) {
     while ($row = mysql_fetch_assoc($result)) {
         print_r($row);
```

```
    }
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [Field] => id
     [Type] => int(7)
     [Null] =>
     [Key] => PRI
     [Default] =>
      [Extra] => auto_increment
)
Array
(
     [Field] => email
     [Type] => varchar(100)
     [Null] =>
     [Key] =>
     [Default] =>
     [Extra] =>
)
```

注意

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_listfields()**

参考

- mysql_field_flags()
- mysql_info()

---

mysql_list_processes

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_list_processes — MySQL プロセスのリストを得る

説明

resource **mysql_list_processes** ([resource $link_identifier])

カレントの MySQL サーバのスレッドを取得します。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に結果ポインタ resource 、 失敗した場合に **FALSE** を返します。

例

Example#1 mysql_list_processes() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$result = mysql_list_processes($link);
while ($row = mysql_fetch_assoc($result)){
    printf("%s %s %s %s %s¥n", $row["Id"], $row["Host"], $row["db"],
        $row["Command"], $row["Time"]);
}
mysql_free_result($result);
?>
```

上の例の出力は、たとえば以下のようになります。

```
1 localhost test Processlist 0
4 localhost mysql sleep 5
```

参考

- mysql_thread_id()
- mysql_stat()

---

mysql_list_tables

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_list_tables — MySQL データベース上のテーブルのリストを得る

説明

resource **mysql_list_tables** (string $database [, resource $link_identifier])

MySQL データベースから、テーブル名のリストを取得します。

この関数は廃止されました。かわりに mysql_query() を利用して SHOW TABLES [FROM db_name] [LIKE 'pattern'] 文を発行することを推奨します。

パラメータ

database

データベース名。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に結果ポインタ resource 、 失敗した場合に **FALSE** を返します。

結果ポインタの中身を調べるためには mysql_tablename() 関数を利用し、 取得したテーブルを利用するには mysql_fetch_array() などの関数を利用してください。

変更履歴

バージョン	説明
4.3.7	この関数は廃止されました。

例

Example#1 mysql_list_tables() の別の例

```
<?php
$dbname = 'mysql_dbname';

if (!mysql_connect('mysql_host', 'mysql_user', 'mysql_password')) {
    echo 'Could not connect to mysql';
    exit;
}

$sql = "SHOW TABLES FROM $dbname";
$result = mysql_query($sql);

if (!$result) {
    echo "DB Error, could not list tables¥n";
    echo 'MySQL Error: ' . mysql_error();
    exit;
}

while ($row = mysql_fetch_row($result)) {
    echo "Table: {$row[0]}¥n";
}

mysql_free_result($result);
?>
```

注意

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_listtables()**

参考

- mysql_list_dbs()
- mysql_tablename()

---

mysql_num_fields

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_num_fields — 結果におけるフィールドの数を得る

説明

int **mysql_num_fields** (resource $result)

クエリから、フィールドの数を取得します。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

返り値

成功した場合に結果セット resource のフィールド数、 失敗した場合に **FALSE** を返します。

例

Example#1 mysql_num_fields() の例

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
    echo 'Could not run query: ' . mysql_error();
    exit;
}

/* id,email の 2 つのフィールドがあるので、2 を返す */
echo mysql_num_fields($result);
?>
```

注意

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_numfields()**

参考

- mysql_select_db()
- mysql_query()
- mysql_fetch_field()
- mysql_num_rows()

---

mysql_num_rows

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_num_rows — 結果における行の数を得る

説明

int **mysql_num_rows** (resource $result)

結果セットから行の数を取得します。このコマンドは、 SELECT や SHOW のような、実際に結果セットを返す文に対してのみ有効です。 INSERT、UPDATE、REPLACE、DELETE クエリで変更された行の数を得るには、 mysql_affected_rows() を使用してください。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

返り値

成功した場合に結果セットの行の数、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_num_rows() の例

```
<?php

$link = mysql_connect("localhost", "mysql_user", "mysql_password");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);

echo "$num_rows Rows¥n";

?>
```

注意

注意: mysql_unbuffered_query() を使用した場合、 結果セットのすべての行を取得するまで **mysql_num_rows()** は正しい値を返しません。

注意: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_numrows()**

参考

- mysql_affected_rows()
- mysql_connect()
- mysql_data_seek()
- mysql_select_db()
- mysql_query()

---

mysql_pconnect

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_pconnect — MySQL サーバへの持続的な接続をオープンする

説明

resource **mysql_pconnect** ([string $server [, string $username [, string $password [, int $client_flags]]]])

MySQL サーバとの持続的な接続を確立します。

mysql_pconnect()は、 mysql_connect()とよく似た動作をしますが、 2 つの大きな違いがあります。

1 番目の違いとして、この関数は接続時にまず 同じホスト、ユーザ名、パスワードを有する(持続的)リンクが すでにオープンされていないかどうかを調べます。 それがみつかった場合、新規の接続をオープンする代わりに そのリンクの ID が返されます。

2 番目の違いは、スクリプトの実行が終了しても SQL サーバとの接続が 閉じられないということです。そのかわりに、将来再利用されるために リンクがオープンされたままとなります (mysql_close() は、**mysql_pconnect()** によって確立されたリンクを 閉じません)。

このため、この型のリンクは、'持続的(persistent)'であると言われます。

パラメータ

server

MySQL サーバ。"hostname:port" のようにポート番号を 指定することが可能で、localhost では ":/path/to/socket" のようにソケットへのパスを指定することも可能です。

PHP ディレクティブ mysql.default_host が指定されない場合(デフォルト)、 'localhost:3306' が使用されます。

username

ユーザ名。デフォルト値はサーバプロセスの所有ユーザ名です。

password

パスワード。デフォルト値は空のパスワードです。

client_flags

パラメータ client_flags は、 以下の定数の組み合わせです: 128 (LOAD DATA LOCAL の処理を有効にする)、 **MYSQL_CLIENT_SSL**、**MYSQL_CLIENT_COMPRESS**、 **MYSQL_CLIENT_IGNORE_SPACE** または **MYSQL_CLIENT_INTERACTIVE**

返り値

成功した場合に MySQL 持続的リンク ID を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	パラメータ client_flags が追加されました。
3.0.10	server に、":/path/to/socket" のサポートが追加されました。
3.0.0	server に、":port" のサポートが追加されました。

注意

> **注意**: この接続方法は、モジュールバージョンの PHP でのみ使用可能であることに 注意しましょう。詳しい情報は 持続的 データベース接続 を参照ください。

警告

持続的接続を利用する場合、MySQL の同時接続数の制限をこえないように Apache や MySQL の設定を多少変更する必要があるかも知れません。

> **注意**: 関数名の前に @ を追加すると、失敗した場合のエラーメッセージを抑制できます。

参考

- mysql_connect()
- Persistent Database Connections

---

mysql_ping

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_ping — サーバとの接続状況を調べ、接続されていない場合は再接続する

説明

bool **mysql_ping** ([resource $link_identifier])

サーバとの接続が有効かどうかを調べます。 もし接続が切れていた場合、自動的に再接続が試みられます。 この関数は、アイドル期間が長いスクリプトで利用し、 サーバが接続を切断したかどうかを確認するために用いられます。

注意: MySQL 5.0.13 以降、自動再接続機能は使えなくなりました。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

MySQL サーバとの接続が有効な場合に **TRUE** そうでない場合に **FALSE** を返します。

例

Example#1 mysql_ping() の例

```
<?php
set_time_limit(0);

$conn = mysql_connect('localhost', 'mysqluser', 'mypass');
$db   = mysql_select_db('mydb');

/* このクエリは非常に時間がかかるものと仮定する */
$result = mysql_query($sql);
if (!$result) {
    echo 'Query #1 failed, exiting.';
    exit;
}

/* 接続が有効かどうかを確かめる。切断されていたら再接続する */
if (!mysql_ping($conn)) {
    echo 'Lost connection, exiting after query #1';
    exit;
}
mysql_free_result($result);

/* 接続が有効であることが確かめられたので、別のクエリを実行する */
$result2 = mysql_query($sql2);
?>
```

参考

- mysql_thread_id()
- mysql_list_processes()

---

mysql_query

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_query — MySQL クエリを送信する

説明

resource **mysql_query** (string $query [, resource $link_identifier])

mysql_query() は、 ひとつのクエリを送信します (複数クエリの送信はサポートしません)。 送信先は、link_identifier で指定したサーバ上にある、現在アクティブなデータベースです。

パラメータ

query

SQL クエリ。

クエリ文字列は、セミコロンで終えてはいけません。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

SELECT, SHOW, DESCRIBE や EXPLAIN 文、その他結果セットを返す文では、 **mysql_query()** は成功した場合に resource を返します。エラー時には **FALSE** を返します。

それ以外の SQL 文 UPDATE, DELETE, DROP などでは、 **mysql_query()** は成功した場合に **TRUE** 、エラー時に **FALSE** を返します。

返された結果にアクセスするためには、結果リソースを mysql_fetch_array() やその他の関数に渡します。

SELECT 文によって返された行の数を知るには mysql_num_rows() を用います。また DELETE, INSERT, REPLACE, または UPDATE 文で変更された行の数を 知るには mysql_affected_rows() を用います。

クエリが参照するテーブルにアクセスする権限がない場合も **mysql_query()**は失敗し、**FALSE** が返されます。

例

Example#1 間違ったクエリ

次のクエリは文法的に間違っているので、 **mysql_query()**は失敗し **FALSE**を返します。

```
<?php
$result = mysql_query('SELECT * WHERE 1=1');
if (!$result) {
    die('Invalid query: ' . mysql_error());
}

?>
```

Example#2 正しいクエリ

次のクエリは正しいので、**mysql_query()** は resource を返します。

```
<?php
// これはユーザが指定する。たとえば
$firstname = 'fred';
$lastname  = 'fox';

// クエリの作成
// これは SQL クエリを実行する最良の方法です。
// さらなる例は、mysql_real_escape_string() を参照ください。
$query = sprintf("SELECT firstname, lastname, address, age FROM friends WHERE firstname='%s' AND lastname='%s'",
    mysql_real_escape_string($firstname),
    mysql_real_escape_string($lastname));

// クエリの実行
$result = mysql_query($query);

// 結果のチェック
// MySQL に送られたクエリと返ってきたエラーをそのまま表示します。デバッグに便利です。
if (!$result) {
    $message  = 'Invalid query: ' . mysql_error() . "¥n";
    $message .= 'Whole query: ' . $query;
    die($message);
}

// 結果の利用
// $result をそのまま出力してもリソースの内部の情報にはアクセスできません。
// 結果にたいして MySQL の関数を適用する必要があります。
// mysql_result(), mysql_fetch_array(), mysql_fetch_row() なども参照ください。
while ($row = mysql_fetch_assoc($result)) {
    echo $row['firstname'];
    echo $row['lastname'];
    echo $row['address'];
    echo $row['age'];
}

// 結果セットに関連付けられているリソースの開放
// これは、スクリプトが終了する際に自動的に実行されます。
mysql_free_result($result);
?>
```

参考

- mysql_connect()
- mysql_error()
- mysql_real_escape_string()
- mysql_result()
- mysql_fetch_assoc()
- mysql_unbuffered_query()

---

mysql_real_escape_string

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_real_escape_string — SQL 文中で用いる文字列の特殊文字をエスケープする

説明

string **mysql_real_escape_string** (string $unescaped_string [, resource $link_identifier])

現在の接続の文字セットで unescaped_string の特殊文字をエスケープし、 mysql_query() で安全に利用できる形式に変換します。バイナリデータを挿入しようとしている場合、 必ずこの関数を利用しなければなりません。

mysql_real_escape_string() は、MySQL のライブラリ関数 mysql_real_escape_string をコールしています。 これは以下の文字について先頭にバックスラッシュを付加します。 ¥x00, ¥n, ¥r, ¥, ', " そして ¥x1a.

データの安全性を確保するため、MySQL へクエリを送信する場合には （わずかな例外を除いて）常にこの関数を用いなければなりません。

パラメータ

unescaped_string

エスケープされる文字列。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合にエスケープ後の文字列、失敗した場合に **FALSE** を返します。

例

Example#1 単純な mysql_real_escape_string() の例

```
<?php
// 接続
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
    OR die(mysql_error());

// クエリ
$query = sprintf("SELECT * FROM users WHERE user='%s' AND password='%s'",
            mysql_real_escape_string($user),
            mysql_real_escape_string($password));
?>
```

Example#2 SQL インジェクション攻撃の例

```
<?php
// データベース上のユーザに一致するかどうかを調べる
$query = "SELECT * FROM users WHERE user='{$_POST['username']}' AND password='{$_POST['password']}'";
mysql_query($query);

// $_POST['password'] をチェックしなければ、このような例でユーザに望みどおりの情報を取得されてしまう
$_POST['username'] = 'aidan';
$_POST['password'] = "' OR ''='";

// MySQL に送信されたクエリは、
echo $query;
?>
```

MySQL に送信されたクエリは次のとおり:

```
SELECT * FROM users WHERE user='aidan' AND password='' OR ''=''
```

これでは、パスワードを知らなくても誰でもログインできてしまいます。

Example#3 "うまいやり方" のクエリ

それぞれの変数に **mysql_real_escape_string()** を適用し、 SQL インジェクションを防ぎます。この例では、 データベースにクエリを送信する場合の "うまいやり方" を示します。これは、 マジッククオート の設定に依存しません。

```
<?php

if (isset($_POST['product_name']) && isset($_POST['product_description']) && isset($_POST['user_id'])) {
    // 接続します

    $link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password');

    if(!is_resource($link)) {

        echo "サーバへの接続に失敗しました\n";
        // ... エラーを適切にログ出力します

    } else {

        // magic_quotes_gpc/magic_quotes_sybase が ON になっている場合に、その処理内容を元に戻します

        if(get_magic_quotes_gpc()) {
            $product_name        = stripslashes($_POST['product_name']);
            $product_description = stripslashes($_POST['product_description']);
        } else {
            $product_name        = $_POST['product_name'];
            $product_description = $_POST['product_description'];
        }

        // 安全なクエリを作成します
        $query = sprintf("INSERT INTO products (`name`, `description`, `user_id`) VALUES ('%s', '%s', %d)",
                    mysql_real_escape_string($product_name, $link),
                    mysql_real_escape_string($product_description, $link),
                    $_POST['user_id']);

        mysql_query($query, $link);

        if (mysql_affected_rows($link) > 0) {
            echo "製品を追加しました\n";
        }
    }
} else {
    echo "フォームの内容を適切に入力してください\n";
```

```
}
?>
```

これでクエリは正しく実行され、SQL インジェクション攻撃が機能しなくなります。

注意

注意: **mysql_real_escape_string()** を利用する前に、MySQL 接続が確立されている必要があります。もし存在しなければ、E_WARNING レベルのエラーが生成され、**FALSE** が返されます。link_identifier が指定されなかった場合は、 直近の MySQL 接続が用いられます。

注意: magic_quotes_gpc が有効な場合は、 まず最初に stripslashes() を適用します。そうしないと、 すでにエスケープされているデータに対してさらにエスケープ処理を してしまうことになります。

注意: この関数を用いてデータをエスケープしなければ、クエリは SQL インジェクション攻撃 に対しての脆弱性を持ったものになります。

注意: **mysql_real_escape_string()** は % や _ をエスケープしません。 MySQL では、これらの文字を LIKE, GRANT, または REVOKE とともに用いることで、 ワイルドカードを表現します。

参考

- mysql_client_encoding()
- addslashes()
- stripslashes()
- The magic_quotes_gpc ディレクティブ
- The magic_quotes_runtime ディレクティブ

---

mysql_result

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_result — 結果データを得る

説明

string **mysql_result** (resource $result , int $row [, mixed $field])

MySQL の結果セットからひとつのセルの内容を取得します。

大量の結果セットで作業を行う際は、行全体を取り込む関数のうちひとつを 使用することを検討するべきです（以下で説明します）。 これらの関数は一回の関数コールで複数のセルの内容を返すので、 **mysql_result()** よりもかなり高速です。 また、フィールド引数としてオフセット数値を指定する方が フィールド名やテーブル名.フィールド名のように指定するよりも かなり高速です。

パラメータ

result

評価された結果 リソース。この結果は、mysql_query() のコールにより得られたものです。

row

結果から取得する行の番号。行番号は 0 からはじまります。

field

取得したいフィールド名またはフィールドのオフセット。

フィールドのオフセット、フィールド名またはテーブル名.フィールド名を 指定可能です。カラム名のエイリアスが定義されている ('select foo as bar from...') 場合、そのカラム名の代わりに エイリアスを使用してください。指定しなかった場合は最初のフィールドを 取得します。

返り値

成功した場合に MySQL 結果セットのひとつのセルの内容、 失敗した場合に **FALSE** を返します。

例

Example#1 mysql_result() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
$result = mysql_query('SELECT name FROM work.employee');
if (!$result) {
    die('Could not query:' . mysql_error());
}
echo mysql_result($result, 2); // 3 番目の employee の name を出力する

mysql_close($link);
?>
```

注意

注意: **mysql_result()** は、 結果セットを処理するほかの関数と混用することはできません。

参考

- mysql_fetch_row()
- mysql_fetch_array()
- mysql_fetch_assoc()
- mysql_fetch_object()

---

mysql_select_db

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_select_db — MySQL データベースを選択する

説明

bool **mysql_select_db** (string $database_name [, resource $link_identifier])

指定したリンク ID が指すサーバ上のデータベースを、アクティブな データベースに設定します。それ以降にコールされる mysql_query() は、すべてアクティブなデータベース上で 実行されます。

パラメータ

database_name

選択するデータベース名。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_select_db() の例

```
<?php

$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Not connected : ' . mysql_error());
}

// foo をカレントの db に指定する
$db_selected = mysql_select_db('foo', $link);
if (!$db_selected) {
    die ('Can¥'t use foo : ' . mysql_error());
}
?>
```

注意

> **注意**: 下位互換のために、次の非推奨別名を使用してもいいでしょう。 **mysql_selectdb()**

参考

- mysql_connect()
- mysql_pconnect()
- mysql_query()

---

mysql_set_charset

(PHP 5 >= 5.2.3)

mysql_set_charset — クライアントの文字セットを設定する

説明

bool **mysql_set_charset** (string $charset [, resource $link_identifier])

現在の接続における、デフォルトの文字セットを設定します。

パラメータ

charset

有効な文字セット名。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場

合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、E_WARNING レベルの警告が生成されます。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

注意

注意: この関数は、MySQL 5.0.7 以降でないと使用できません。

---

mysql_stat

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_stat — 現在のシステムの状態を取得する

説明

```
string mysql_stat ([ resource $link_identifier ] )
```

mysql_stat() は現在のサーバの状態を返します。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、E_WARNING レベルの警告が生成されます。

返り値

稼働時間、スレッド、クエリ、オープンされているテーブル、フラッシュされた テーブル、そして 1 秒あたりのクエリ数を文字列で返します。その他のデータも 含めた完全な状態を得るには、SHOW STATUS SQL コマンドを 実行する必要があります。link_identifier が不正な 場合には NULL が返されます。

例

Example#1 mysql_stat() の例

```php
<?php
$link   = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$status = explode('  ', mysql_stat($link));
print_r($status);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [0] => Uptime: 5380
    [1] => Threads: 2
    [2] => Questions: 1321299
    [3] => Slow queries: 0
    [4] => Opens: 26
    [5] => Flush tables: 1
    [6] => Open tables: 17
    [7] => Queries per second avg: 245.595
)
```

Example#2 mysql_stat() の別の例

```php
<?php
$link   = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$result = mysql_query('SHOW VARIABLES', $link);
while ($row = mysql_fetch_assoc($result)) {
    echo $row['Variable_name'] . ' = ' . $row['Value'] . "¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
back_log = 50
basedir = /usr/local/
bdb_cache_size = 8388600
bdb_log_buffer_size = 32768
bdb_home = /var/db/mysql/
bdb_max_lock = 10000
bdb_logdir =
bdb_shared_data = OFF
bdb_tmpdir = /var/tmp/
...
```

参考

- mysql_get_server_info()
- mysql_list_processes()

---

mysql_tablename

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_tablename — フィールドのテーブル名を得る

説明

string **mysql_tablename** (resource $result , int $i)

result からテーブル名を取得します。

この関数は非推奨です。かわりに mysql_query() を利用して SHOW TABLES [FROM db_name] [LIKE 'pattern'] 文を発行することを推奨します。

パラメータ

result

mysql_list_tables() から返される 結果ポインタ resource 。

i

整数のインデックス（行/テーブル 番号）。

返り値

成功した場合にテーブル名、失敗した場合に **FALSE** を返します。

結果ポインタの中身を調べるために **mysql_tablename()** 関数を利用し、取得したテーブルを 利用するには mysql_fetch_array() などの関数を利用してください。

例

Example#1 mysql_tablename() の例

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password");
$result = mysql_list_tables("mydb");
$num_rows = mysql_num_rows($result);
for ($i = 0; $i < $num_rows; $i++) {
    echo "Table: ", mysql_tablename($result, $i), "¥n";
}

mysql_free_result($result);
?>
```

注意

> **注意**: 結果ポインタに含まれるテーブル数を調べるには mysql_num_rows() 関数を利用します。

参考

- mysql_list_tables()
- mysql_field_table()
- mysql_db_name()

---

mysql_thread_id

(PHP 4 >= 4.3.0, PHP 5, PECL mysql:1.0)

mysql_thread_id — カレントのスレッド ID を返す

説明

int **mysql_thread_id** ([resource $link_identifier])

カレントのスレッド ID を取得します。接続が一度切断され、 mysql_ping() によって再接続された場合には、 スレッド ID は変わります。つまり、スレッド ID は必要になったその時点で 取得すべきだということです。

パラメータ

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

成功した場合にスレッド ID 、失敗した場合に **FALSE** を返します。

例

Example#1 mysql_thread_id() の例

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$thread_id = mysql_thread_id($link);
if ($thread_id){
    printf("current thread id is %d¥n", $thread_id);
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
current thread id is 73
```

参考

- mysql_ping()
- mysql_list_processes()

---

mysql_unbuffered_query

(PHP 4 >= 4.0.6, PHP 5, PECL mysql:1.0)

mysql_unbuffered_query — MySQL に SQL クエリを送信するが、結果に対してのフェッチやバッファリングは行わない

説明

```
resource mysql_unbuffered_query ( string $query [, resource $link_identifier ] )
```

mysql_unbuffered_query() は SQL クエリ query を MySQL に送信します。その際、 mysql_query() が行っているような自動バッファリングを 行いません。一方、この挙動により SQL クエリが消費するメモリの量を おさえられます。また、最初の 1 行目が取得されたらすぐに処理を はじめることができます。SQL の処理が完全に終わるまで待つ必要がありません。 複数の DB 接続を利用する場合には、オプションのパラメータ link_identifier を指定する必要があります。

パラメータ

query

SQL クエリ。

link_identifier

MySQL 接続。指定されない場合、mysql_connect() により直近にオープンされたリンクが指定されたと仮定されます。そのようなリンクがない場合、引数を指定せずに mysql_connect() がコールした時と同様にリンクを確立します。リンクが見付からない、または、確立できない場合、**E_WARNING** レベルの警告が生成されます。

返り値

SELECT, SHOW, DESCRIBE あるいは EXPLAIN では、 **mysql_unbuffered_query()** は 成功した場合に resource 、エラー時に **FALSE** を返します。

UPDATE, DELETE, DROP, などその他の SQL 文では、 **mysql_unbuffered_query()** は 成功した場合に **TRUE** 、エラー時に **FALSE** を返します。

注意

> **注意**: **mysql_unbuffered_query()** の利点には、以下のような 代償があります: **mysql_unbuffered_query()** から返される結果セットには mysql_num_rows() や mysql_data_seek() が使用できません。また、結果の行をすべてフェッチするまで、MySQL に 新しいクエリを送信することができません。

参考

- mysql_query()

目次

- mysql_errno — 直近の MySQL 処理からエラーメッセージのエラー番号を返す
- mysql_error — 直近に実行された MySQL 操作のエラーメッセージを返す
- mysql_escape_string — mysql_query で使用するために文字列をエスケープする
- mysql_fetch_array — 連想配列、添字配列、またはその両方として結果の行を取得する
- mysql_fetch_assoc — 連想配列として結果の行を取得する
- mysql_fetch_field — 結果からカラム情報を取得し、オブジェクトとして返す
- mysql_fetch_lengths — 結果における各出力の長さを得る
- mysql_fetch_object — 結果の行をオブジェクトとして取得する
- mysql_fetch_row — 結果を添字配列として取得する
- mysql_field_flags — 結果において指定したフィールドのフラグを取得する
- mysql_field_len — 指定したフィールドの長さを返す
- mysql_field_name — 結果において指定したフィールド名を取得する
- mysql_field_seek — 結果ポインタを指定したフィールドオフセットにセットする
- mysql_field_table — 指定したフィールドが含まれるテーブルの名前を取得する
- mysql_field_type — 結果において指定したフィールドの型を取得する
- mysql_free_result — 結果保持用メモリを開放する
- mysql_get_client_info — MySQL クライアント情報を取得する
- mysql_get_host_info — MySQL ホスト情報を取得する
- mysql_get_proto_info — MySQL プロトコル情報を取得する
- mysql_get_server_info — MySQL サーバ情報を取得する
- mysql_info — 直近のクエリについての情報を得る
- mysql_insert_id — 直近の INSERT 操作で生成された ID を得る
- mysql_list_dbs — MySQL サーバ上で利用可能なデータベースのリストを得る
- mysql_list_fields — MySQL テーブルのフィールドのリストを得る
- mysql_list_processes — MySQL プロセスのリストを得る
- mysql_list_tables — MySQL データベース上のテーブルのリストを得る
- mysql_num_fields — 結果におけるフィールドの数を得る
- mysql_num_rows — 結果における行の数を得る
- mysql_pconnect — MySQL サーバへの持続的な接続をオープンする
- mysql_ping — サーバとの接続状況を調べ、接続されていない場合は再接続する
- mysql_query — MySQL クエリを送信する
- mysql_real_escape_string — SQL 文中で用いる文字列の特殊文字をエスケープする
- mysql_result — 結果データを得る
- mysql_select_db — MySQL データベースを選択する
- mysql_set_charset — クライアントの文字セットを設定する
- mysql_stat — 現在のシステムの状態を取得する
- mysql_tablename — フィールドのテーブル名を得る
- mysql_thread_id — カレントのスレッド ID を返す
- mysql_unbuffered_query — MySQL に SQL クエリを送信するが、結果に対してのフェッチやバッファリングは行わない

MySQL 関数 (PDO_MYSQL)

導入

PDO_MYSQL は、PHP から MySQL 3.x、4.x および 5.x データベースへのアクセスを可能にするための PHP Data Objects (PDO) インターフェース を実装したドライバです。

PDO_MYSQL は、MySQL 4.1 以降に存在するプリペアドステートメントを ネイティブにサポートしているという利点があります。 古いバージョンの mysql クライアントライブラリを使用している場合は、 PDO がこの機能をエミュレートします。

警告

注意: MySQL のテーブル型 (ストレージエンジン) の中には、 トランザクションをサポートしていないものがあります。 トランザクションをサポートしていないテーブル型を使用して トランザクションを使用するコードを書くと、 MySQL はトランザクションが正常に使用できたかのように振舞います。 さらに、DDL クエリを実行する際には、 実行中のトランザクションが暗黙的にコミットされます。

定義済み定数

このドライバでは以下の定数が定義されています。これは拡張モジュールが PHP に組み込まれているか、実行時に動的にロードされている場合のみ使用可能です。さらに、これらのドライバ固有の定数はそのドライバを使用している場合にのみ使用されます。 postgres ドライバで mysql 固有の属性を使用すると、予期せぬ結果を引き起こします。もし複数のドライバを使用しているコードを実行している場合、 **PDO::getAttribute()** で **PDO_ATTR_DRIVER_NAME** 属性を使用することで、使用中のドライバ名を調べることが可能です。

PDO::MYSQL_ATTR_USE_BUFFERED_QUERY (integer)
PDOStatement でこの属性を **TRUE** に設定すると、 MySQL ドライバはバッファ版の MySQL API を使用します。 移植性の高いコードを書くには、代わりに **PDOStatement::fetchAll()** を使用すべきです。

Example#1 mysql でクエリのバッファリングを強制する

```
<?php
if ($db->getAttribute(PDO::ATTR_DRIVER_NAME) == 'mysql') {
    $stmt = $db->prepare('select * from foo',
        array(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY => true));
```

```
} else {
    die("このアプリケーションは mysql でしか動作しません。代わりに \$stmt->fetchAll() を使用すべきです");
}
?>
```

PDO::MYSQL_ATTR_LOCAL_INFILE (integer)

LOAD LOCAL INFILE を有効にします。

PDO::MYSQL_ATTR_INIT_COMMAND (integer)

MySQL サーバへの接続時に実行するコマンドを指定します。 再接続の際には自動的に再実行されます。

PDO::MYSQL_ATTR_READ_DEFAULT_FILE (integer)

my.cnf ではなく、 指定した名前のファイルからオプションを読み込みます。

PDO::MYSQL_ATTR_READ_DEFAULT_GROUP (integer)

my.cnf あるいは 別のファイル (**MYSQL_READ_DEFAULT_FILE** で指定したもの) の中の、指定した名前のグループからオプションを読み込みます。

PDO::MYSQL_ATTR_MAX_BUFFER_SIZE (integer)

バッファの最大サイズ。デフォルトは 1 MiB です。

PDO::MYSQL_ATTR_DIRECT_QUERY (integer)

プリペアドステートメントではなく、直接クエリを実行します。

PDO_MYSQL DSN

(No version information available, might be only in CVS)

PDO_MYSQL DSN — MySQL データベースに接続する

説明

PDO_MYSQL データソース名 (DSN) は以下の要素で構成されます。

DSN 接頭辞

DSN 接頭辞は **mysql:** です。

host

データベースサーバが存在するホスト名を指定します。

port

データベースサーバが待機しているポートを指定します。

dbname

データベース名を指定します。

unix_socket

MySQL の unix ソケットを指定します (host あるいは port と同時に使用することはできません)。

例

Example#1 PDO_MYSQL DSN の例

以下の例は、MySQL データベースに接続するための PDO_MYSQL DSN を表します。

```
mysql:host=localhost;dbname=testdb
```

より完全な例は、このようになります。

```
mysql:host=localhost;port=3307;dbname=testdb
mysql:unix_socket=/tmp/mysql.sock;dbname=testdb
```

MySQL 改良版拡張サポート (mysqli)

導入

mysqli 拡張サポートによって MySQL 4.1 以上で提供される機能を利用することが できるようになります。MySQL データベースサーバに関する詳細は » http://www.mysql.com/ をご覧ください。

MySQL に関するドキュメントは » http://dev.mysql.com/doc/ にあります。

このドキュメントの一部は、MySQL AB の許可を得て MySQL マニュアルから引用 したものです。

要件

これらの関数を有効にするには、mysqli 拡張サポートを 有効にして PHP をコンパイルする必要があります。

注意: mysqli 拡張サポートは MySQL 4.1.3 以上で動作するよう設計されています。 それ以前のバージョンについては MySQL 拡張サ

ポートのドキュメントをご覧ください。

インストール手順

PHP に mysqli 拡張サポートを追加するには、 --with-mysqli=mysql_config_path/mysql_config オプションを指定して PHP をコンパイルする必要があります。 mysql_config_path には MySQL 4.1 以上に付属する mysql_config プログラムが存在するパスを 指定してください。

mysqli 拡張サポートと標準の mysql 拡張を共存させる形でインストールしたい 場合には、衝突を避けるために同じクライアントライブラリを利用する必要があります。

Windows へのインストール

MySQLi はデフォルトでは有効となりません。したがって php.ini で php_mysqli.dll DLL を有効にしておく必要があります。 また、PHP が MySQL クライアントライブラリにアクセスできなければなりません。 libmysql.dll というファイルが Windows 版の PHP 配布ファイルに含まれており、PHP が MySQL と話すためにはこのファイルが Windows の PATH にある必要があります。そのための方法については、 "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" という FAQ を参照してください。 libmysql.dll を Windows のシステムディレクトリにコピーしても動作しますが (システムディレクトリは、デフォルトでシステムの PATH に含まれています)、お勧めしません。

> **注意**: MySQL 5 に接続する際は、 » http://dev.mysql.com/downloads/connector/php/ からバイナリをダウンロードすることを推奨します。

(php_mysqli.dll も含めた) PHP 拡張モジュールを有効にするには、PHP ディレクティブ extension_dir に拡張モジュールの存在する場所を設定するべきです。 Windows へのマニュアルインストール方法 も参照してください。PHP 5 での extension_dir の例は c:¥php¥ext です。

> **注意**: Web サーバの起動時に以下のようなエラーが発生する場合： "Unable to load dynamic library './php_mysqli.dll'" これは php_mysqli.dll や libmysql.dll がシステムによって見つけられなかったことが原因です。

実行時設定

php.ini の設定により動作が変化します。

MySQLi 設定オプション

名前	デフォルト	変更の可否	変更履歴
mysqli.max_links	"-1"	PHP_INI_SYSTEM	PHP 5.0.0 以降で使用可能です。
mysqli.default_port	"3306"	PHP_INI_ALL	PHP 5.0.0 以降で使用可能です。
mysqli.default_socket	NULL	PHP_INI_ALL	PHP 5.0.0 以降で使用可能です。
mysqli.default_host	NULL	PHP_INI_ALL	PHP 5.0.0 以降で使用可能です。
mysqli.default_user	NULL	PHP_INI_ALL	PHP 5.0.0 以降で使用可能です。
mysqli.default_pw	NULL	PHP_INI_ALL	PHP 5.0.0 以降で使用可能です。

PHP_INI_* 定数の詳細と定義については、 設定の変更 を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

mysqli.max_links integer

プロセス毎の MySQL 接続の最大数。

mysqli.default_port string

他のポートが指定されない場合、データベースサーバ接続時に使用される デフォルトの TCP ポート番号。デフォルトが指定されない場合は、 環境変数 MYSQL_TCP_PORT・ /etc/services の mysql-tcpエントリ・コンパイル時の **MYSQL_PORT** 定数の順番でポートが取得されます。 Win32 では、**MYSQL_PORT** 定数のみが使用されます。

mysqli.default_socket string

他にソケット名が指定されない場合、ローカルな データベースサーバに接続する時のデフォルトのソケット名。

mysqli.default_host string

他のサーバ名が指定されない場合に、データベースサーバへの接続時に 使用されるデフォルトのサーバ名。 safe mode では適用されません。

mysqli.default_user string

他のユーザ名が指定されない場合に、データベースサーバへの接続時に 使用されるデフォルトのユーザ名。 safe mode では適用されません。

mysqli.default_password string

他のパスワードが指定されない場合に、データベースサーバへの接続時に 使用されるデフォルトのパスワード。 safe mode では適用されません。

定義済みクラス

mysqli

PHP と MySQL データベースの間の接続を表します。

コンストラクタ

- mysqli - 新たに mysqli オブジェクトを作成します

メソッド

- autocommit - データベース変更時のオートコミットをオンまたはオフにします

- change_user - 指定したデータベース接続のユーザを変更します
- character_set_name - データベース接続のデフォルトの文字セットを返します
- close - オープンされている接続をクローズします
- commit - カレントのトランザクションをコミットします
- connect - MySQL データベースサーバーへの新規接続をオープンします
- debug - デバッグ処理を行います
- dump_debug_info - デバッグ情報をダンプします
- get_client_info - クライアントのバージョンを返します
- get_host_info - 使用されている接続の型を返します
- get_server_info - MySQL サーバーのバージョンを返します
- get_server_version - MySQL サーバーのバージョンを返します
- init - mysqli オブジェクトを初期化します
- info - 直近に実行されたクエリに関する情報を取得します
- kill - 指定した MySQL スレッドをキルするようサーバーに指示します
- multi_query - 複数のクエリを実行します
- more_results - 実行した複数のクエリについて結果がまだ残ってるかどうかを確認します
- next_result - 実行した複数のクエリから次の結果を読み込みます
- options - オプションを設定します
- ping - 指定したサーバー接続に ping を行い、接続がない場合には再接続します
- prepare - SQL クエリをパースします
- query - クエリを実行します
- real_connect - MySQL データベースサーバーへの接続をオープンします
- escape_string - 接続に使用する文字セットを考慮して SQL 命令が含まれる文字列の中の特殊文字をエスケープします
- rollback - カレントのトランザクションをロールバックします
- select_db - デフォルトのデータベースを選択します
- set_charset - デフォルトのクライアント文字セットを設定します
- ssl_set - SSL パラメータを設定します
- stat - カレントのシステムステータスを取得します
- stmt_init- mysqli_stmt_prepare で利用するステートメントを初期化します
- store_result - 直近のクエリから結果セットを伝送します
- thread_safe - スレッドセーフかどうかを返します
- use_result - 直近のクエリからバッファリングされていない結果セットを伝送します

プロパティ

- affected_rows - 直近の MySQL 操作で変更された行の数を取得します
- client_info - MySQL クライアントのバージョンを文字列で返します
- client_version - MySQL クライアントのバージョンを整数で返します
- errno - 直近にコールされた関数のエラーコードを返します
- error - 直近にコールされた関数のエラー文字列を返します
- field_count - 直近のクエリのカラム数を返します
- host_info - 使用されている接続の型を返します
- info - 直近に実行されたクエリについての情報を取得します
- insert_id - 直近のクエリで使用された、自動生成 ID を返します
- protocol_version - 使用されている MySQL プロトコルのバージョンを返します
- server_info - サーバのバージョン番号を表す文字列を返します
- server_version - サーバのバージョン番号を整数値で返します
- sqlstate - 直近のエラーについて、SQLSTATE エラーコードを含む文字列を返します
- thread_id - カレントの接続のスレッド ID を返します
- warning_count - 直近の SQL ステートメントの中で発生した警告の数を返します

mysqli_stmt

プリペアドステートメントを表します。

メソッド

- bind_param - 変数をプリペアドステートメントにバインドします
- bind_result - 結果を保存するため、変数をプリペアドステートメントにバインドします
- close - プリペアドステートメントを閉じます
- data_seek - ステートメント結果セットの任意の行に移動します
- execute - プリペアドステートメントを実行します
- fetch - プリペアドステートメントから結果を取得し、バインドした変数に保存します
- free_result - 指定したステートメントハンドルの結果を保存しているメモリを開放します
- prepare - SQL クエリを準備します
- reset - プリペアドステートメントをリセットします
- result_metadata - プリペアドステートメントから、メタ情報の結果セットを取得します
- send_long_data - データを分割して送信します
- store_result - プリペアドステートメントの結果をすべてメモリに保存します

プロパティ

- affected_rows - 直近のステートメントの実行により更新された行数を返します
- errno - 直近のステートメントのエラーコードを返します
- error - 直近のステートメントのエラー文字列を返します
- field_count - 結果セットのカラムの数を返します
- id - ステートメントの ID を返します
- insert_id - プリペアドステートメントで AUTO_INCREMENT のカラムに設定された値を返します
- num_rows - 結果セットの行の数を返します
- param_count - 指定したプリペアドステートメントのパラメータ数を返します
- sqlstate - 直近のステートメントについて SQLSTATE エラーコードを含む文字列を返します

mysqli_result

データベースへのクエリにより得られた結果セットを表します。

メソッド

- close - 結果セットを閉じます
- data_seek - 内部の結果ポインタを移動します
- fetch_array - 結果の行を連想配列・数値添字配列あるいはその両方で取得します
- fetch_assoc - 結果の行を連想配列で取得します
- fetch_field - 結果セットからカラムの情報を取得します
- fetch_fields - 結果セットからすべてのカラムの情報を取得します
- fetch_field_direct - 指定したカラムの情報を取得します
- fetch_object - 結果の行をオブジェクトで取得します
- fetch_row - 結果の行を数値添字の配列で取得します
- field_seek - 結果ポインタを指定したフィールドオフセットに移動します
- free_result - 結果のメモリを開放します

プロパティ

- current_field - 現在のフィールドポインタのオフセットを返します
- field_count - 結果セットのフィールド数を返します
- lengths - カラムの長さの配列を返します
- num_rows - 結果の行数を返します
- type - **MYSQLI_STORE_RESULT** あるいは **MYSQLI_USE_RESULT** を返します

定義済み定数

MySQLi 定数

定数名	説明
MYSQLI_READ_DEFAULT_GROUP (integer)	`my.cnf' の指定した名前のグループか、あるいは *MYSQLI_READ_DEFAULT_FILE* で指定したファイルからオプションを読み込みます。
MYSQLI_READ_DEFAULT_FILE (integer)	*my.cnf* のかわりに、指定したファイルからオプションを 読み込みます。
MYSQLI_OPT_CONNECT_TIMEOUT (integer)	接続のタイムアウトまでの秒数。
MYSQLI_OPT_LOCAL_INFILE (integer)	*LOAD LOCAL INFILE* コマンドを有効にします。
MYSQLI_INIT_COMMAND (integer)	MySQL サーバへの接続時に実行するコマンド。 再接続時にも自動的に再実行されます。
MYSQLI_CLIENT_SSL (integer)	SSL（暗号化プロトコル）を使用します。このオプションは、アプリケーション プログラムで指定することはできず、MySQL クライアントライブラリの内部で設定します。
MYSQLI_CLIENT_COMPRESS (integer)	圧縮プロトコルを使用します。
MYSQLI_CLIENT_INTERACTIVE (integer)	（wait_timeout のかわりに）interactive_timeout の秒数を使用できるように します。クライアントセッションの wait_timeout 変数の値は、interactive_timeout 変数の値に設定されます。
MYSQLI_CLIENT_IGNORE_SPACE (integer)	関数名に続く空白文字を許可します。すべての関数名を予約語とします。
MYSQLI_CLIENT_NO_SCHEMA (integer)	db_name.tbl_name.col_name 形式の使用を禁止します。
MYSQLI_CLIENT_MULTI_QUERIES (integer)	
MYSQLI_STORE_RESULT (integer)	結果セットをバッファに格納します。
MYSQLI_USE_RESULT (integer)	結果セットをバッファに格納しません。
MYSQLI_ASSOC (integer)	行データを、カラム名をインデックスとする配列に格納して返します。
MYSQLI_NUM (integer)	行データを、数値インデックスの配列に格納して返します。
MYSQLI_BOTH (integer)	行データを、数値インデックス・カラム名インデックスの両方を 保持する配列に格納して返します。
MYSQLI_NOT_NULL_FLAG (integer)	フィールドは、*NOT NULL* と定義されています。
MYSQLI_PRI_KEY_FLAG (integer)	フィールドは、プライマリキーの一部です。
MYSQLI_UNIQUE_KEY_FLAG (integer)	フィールドは、ユニークキーの一部です。
MYSQLI_MULTIPLE_KEY_FLAG (integer)	フィールドは、インデックスの一部です。
MYSQLI_BLOB_FLAG (integer)	フィールドは *BLOB* と定義されています。
MYSQLI_UNSIGNED_FLAG (integer)	フィールドは *UNSIGNED* と定義されています。
MYSQLI_ZEROFILL_FLAG (integer)	フィールドは *ZEROFILL* と定義されています。
MYSQLI_AUTO_INCREMENT_FLAG (integer)	フィールドは *AUTO_INCREMENT* と定義されています。
MYSQLI_TIMESTAMP_FLAG (integer)	フィールドは *TIMESTAMP* と定義されています。
MYSQLI_SET_FLAG (integer)	フィールドは *SET* と定義されています。
MYSQLI_NUM_FLAG (integer)	フィールドは *NUMERIC* と定義されています。
MYSQLI_PART_KEY_FLAG (integer)	フィールドは、マルチインデックスの一部です。
MYSQLI_GROUP_FLAG (integer)	フィールドは *GROUP BY* の一部です。
MYSQLI_TYPE_DECIMAL (integer)	フィールドは *DECIMAL* と定義されています。
MYSQLI_TYPE_NEWDECIMAL (integer)	精密な *DECIMAL* あるいは *NUMERIC* のフィールドです (MySQL 5.0.3 以降)。
MYSQLI_TYPE_BIT (integer)	フィールドは *BIT* と定義されています (MySQL 5.0.3 以降)。
MYSQLI_TYPE_TINY (integer)	フィールドは *TINYINT* と定義されています。
MYSQLI_TYPE_SHORT (integer)	フィールドは *INT* と定義されています。
MYSQLI_TYPE_LONG (integer)	フィールドは *INT* と定義されています。
MYSQLI_TYPE_FLOAT (integer)	フィールドは *FLOAT* と定義されています。
MYSQLI_TYPE_DOUBLE (integer)	フィールドは *DOUBLE* と定義されています。
MYSQLI_TYPE_NULL (integer)	フィールドは *DEFAULT NULL* と定義されています。
MYSQLI_TYPE_TIMESTAMP (integer)	フィールドは *TIMESTAMP* と定義されています。
MYSQLI_TYPE_LONGLONG (integer)	フィールドは *BIGINT* と定義されています。
MYSQLI_TYPE_INT24 (integer)	フィールドは *MEDIUMINT* と定義されています。
MYSQLI_TYPE_DATE (integer)	フィールドは *DATE* と定義されています。

定数名	説明

例

MySQLI ドキュメンテーションのすべての例では、MySQL AB の world データベースを使用しています。これは » http://dev.mysql.com/get/Downloads/Manual/world.sql.gz/from/pick にあります。

---

mysqli_affected_rows

mysqli->affected_rows

(PHP 5)

mysqli->affected_rows — 直前の MySQL の操作で変更された行の数を得る

説明

手続き型:

int **mysqli_affected_rows** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
int$affected_rows;

直近の INSERT、 UPDATE、REPLACE あるいは DELETE クエリにより変更された行の数を返します。

SELECT 文の場合は、**mysqli_affected_rows()** は mysqli_num_rows() と同じように動作します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

正の整数が返された場合、それは変更された行数かあるいは取得された行数を 示します。ゼロが返された場合、それは UPDATE 文でレコードが更新されなかったか WHERE 条件に当てはまる行がなかった、またはクエリが実行されなかったことを 示します。-1 は、クエリがエラーを返したことを示します。

> **注意**: 変更された行数が整数型の最大値をこえた場合、結果の行数は 文字列として返されます。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 行を挿入します */
$mysqli->query("CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d¥n", $mysqli->affected_rows);

$mysqli->query("ALTER TABLE Language ADD Status int default 0");

/* 行を更新します */
$mysqli->query("UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d¥n", $mysqli->affected_rows);

/* 行を削除します */
$mysqli->query("DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d¥n", $mysqli->affected_rows);

/* すべての行を選択します */
$result = $mysqli->query("SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d¥n", $mysqli->affected_rows);

$result->close();

/* Language テーブルを削除します */
$mysqli->query("DROP TABLE Language");

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
```

```
    printf("Can't connect to localhost. Error: %s¥n", mysqli_connect_error());
    exit();
}

/* 行を挿入します */
mysqli_query($link, "CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d¥n", mysqli_affected_rows($link));

mysqli_query($link, "ALTER TABLE Language ADD Status int default 0");

/* 行を更新します */
mysqli_query($link, "UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d¥n", mysqli_affected_rows($link));

/* 行を削除します */
mysqli_query($link, "DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d¥n", mysqli_affected_rows($link));

/* すべての行を選択します */
$result = mysqli_query($link, "SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d¥n", mysqli_affected_rows($link));

mysqli_free_result($result);

/* Language テーブルを削除します */
mysqli_query($link, "DROP TABLE Language");

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Affected rows (INSERT): 984
Affected rows (UPDATE): 168
Affected rows (DELETE): 815
Affected rows (SELECT): 169
```

参考

- mysqli_num_rows()
- mysqli_info()

---

mysqli_autocommit

mysqli->autocommit()

(PHP 5)

mysqli->autocommit() — データベース更新の自動コミットをオンまたはオフにする

説明

手続き型:

```
bool mysqli_autocommit ( mysqli $link , bool $mode )
```

オブジェクト指向型（メソッド）:

```
mysqli
bool autocommit ( bool $mode )
```

指定されたデータベース接続リソースに対するクエリの 自動コミットモードをオンまたはオフにします。

現在の自動コミットモードを知るには、SQL コマンド SELECT @@autocommit を使用します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

mode

自動コミットを有効にするかどうか。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

注意

注意: この関数は、トランザクションに 対応していないテーブル型（MyISAM あるいは ISAM など）では 動作しません。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 自動コミットを on にします */
$mysqli->autocommit(TRUE);

if ($result = $mysqli->query("SELECT @@autocommit")) {
    $row = $result->fetch_row();
    printf("Autocommit is %s¥n", $row[0]);
    $result->free();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
    printf("Can't connect to localhost. Error: %s¥n", mysqli_connect_error());
    exit();
}

/* 自動コミットを on にします */
mysqli_autocommit($link, TRUE);

if ($result = mysqli_query($link, "SELECT @@autocommit")) {
    $row = mysqli_fetch_row($result);
    printf("Autocommit is %s¥n", $row[0]);
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Autocommit is 1
```

参考

- mysqli_commit()
- mysqli_rollback()

---

mysqli_bind_param

(PHP 5)

mysqli_bind_param — mysqli_stmt_bind_param() のエイリアス

説明

この関数は mysqli_stmt_bind_param() のエイリアスです。

注意

注意: **mysqli_bind_param()** は非推奨で、将来は削除される予定です。

参考

- mysqli_stmt_bind_param()

---

mysqli_bind_result

(PHP 5)

mysqli_bind_result — mysqli_stmt_bind_result() のエイリアス

説明

この関数は mysqli_stmt_bind_result() のエイリアスです。

注意

注意: **mysqli_bind_result()** は非推奨で、廃止予定です。

参考

- mysqli_stmt_bind_result()

---

mysqli_change_user

mysqli->change_user()

(PHP 5)

mysqli->change_user() — 指定されたデータベース接続のユーザ名を変更する

説明

手続き型:

bool **mysqli_change_user** (mysqli $link , string $user , string $password , string $database)

オブジェクト指向型（メソッド）:

mysqli
bool **change_user** (string $user , string $password , string $database)

指定されたデータベース接続のユーザ名を変更し、 現在のデータベースを設定します。

ユーザを正しく変更するには、username と password 引数を正しく渡す必要があります。 またそのユーザが対象のデータベースに対する適切なパーミッションを 持っている必要があります。どんな理由であれ、認証に失敗するとカレントユーザの 認証が継続されます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

user

MySQL のユーザ名。

password

MySQL のパスワード。

database

変更するデータベース。

引数には **NULL** 値を渡すこともできます。 その場合ユーザの変更だけでデータベースの選択はされません。 そのようなケースでデータベースを選択したい場合には mysqli_select_db() 関数を使用してください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: このコマンドを使用すると、常に、カレントのデータベース接続は あたかも完全に新しいデータベース接続であるかのようになります。 これにより、全てのアクティブなトランザクションはロールバックされ、 一時テーブルは全てクローズされ、ロックされたテーブルはすべて 開放されます。

例

Example#1 オブジェクト指向型

```php
<?php

/* データベース test に接続します */
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

/* 接続をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 変数 a を設定します */
$mysqli->query("SET @a:=1");

/* すべてをリセットし、新しいデータベースを選択します */
$mysqli->change_user("my_user", "my_password", "world");

if ($result = $mysqli->query("SELECT DATABASE()")) {
    $row = $result->fetch_row();
    printf("デフォルトデータベース: %s\n", $row[0]);
    $result->close();
}

if ($result = $mysqli->query("SELECT @a")) {
    $row = $result->fetch_row();
```

```
    if ($row[0] === NULL) {
        printf("変数 a の値は NULL\n");
    }
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* データベース test に接続します */
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* 接続をチェックします */
if (!$link) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 変数 a を設定します */
mysqli_query($link, "SET @a:=1");

/* すべてをリセットし、新しいデータベースを選択します */
mysqli_change_user($link, "my_user", "my_password", "world");

if ($result = mysqli_query($link, "SELECT DATABASE()")) {
    $row = mysqli_fetch_row($result);
    printf("デフォルトデータベース: %s\n", $row[0]);
    mysqli_free_result($result);
}

if ($result = mysqli_query($link, "SELECT @a")) {
    $row = mysqli_fetch_row($result);
    if ($row[0] === NULL) {
        printf("変数 a の値は NULL\n");
    }
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
デフォルトデータベース: world
変数 a の値は NULL
```

参考

- mysqli_connect()
- mysqli_select_db()

---

mysqli_character_set_name

mysqli->character_set_name()

(PHP 5)

mysqli->character_set_name() — データベース接続のデフォルトの文字コードセットを返す

説明

手続き型:

```
string mysqli_character_set_name ( mysqli $link )
```

オブジェクト指向型（メソッド）:

```
mysqli
string character_set_name ( void )
```

データベース接続の現在の文字コードセットを返します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

現在の接続のデフォルト文字セットを返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 現在の文字セットを表示します */
$charset = $mysqli->character_set_name();
printf ("Current character set is %s¥n", $charset);

$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (!$link) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 現在の文字セットを表示します */
$charset = mysqli_character_set_name($link);
printf ("Current character set is %s¥n",$charset);

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Current character set is latin1_swedish_ci
```

参考

- mysqli_client_encoding()
- mysqli_real_escape_string()

---

mysqli_client_encoding

(PHP 5)

mysqli_client_encoding — mysqli_character_set_name() のエイリアス

説明

この関数は mysqli_character_set_name() のエイリアスです。

参考

- mysqli_real_escape_string()

---

mysqli_close

mysqli->close()

(PHP 5)

mysqli->close() — 事前にオープンしているデータベース接続を閉じる

説明

手続き型:

```
bool mysqli_close ( mysqli $link )
```

オブジェクト指向型（メソッド）:

```
mysqli
bool close ( void )
```

既に開いているデータベース接続を閉じます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mysqli_connect()
- mysqli_init()
- mysqli_real_connect()

---

mysqli_commit

mysqli->commit()

(PHP 5)

mysqli->commit() — 現在のトランザクションをコミットする

説明

手続き型形式:

bool **mysqli_commit** (mysqli $link)

オブジェクト指向型（メソッド）

mysqli
bool **commit** (void)

データベース接続の現在のトランザクションをコミットします。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向形式

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE Language LIKE CountryLanguage Type=InnoDB");

/* autocommit を off にします */
$mysqli->autocommit(FALSE);

/* データを挿入します */
$mysqli->query("INSERT INTO Language VALUES ('DEU', 'Bavarian', 'F', 11.2)");
$mysqli->query("INSERT INTO Language VALUES ('DEU', 'Swabian', 'F', 9.4)");

/* トランザクションをコミットします */
$mysqli->commit();

/* テーブルを削除します */
$mysqli->query("DROP TABLE Language");

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型形式

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* 接続をチェックします */
if (!$link) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}
```

```
/* autocommit を off にします */
mysqli_autocommit($link, FALSE);

mysqli_query($link, "CREATE TABLE Language LIKE CountryLanguage Type=InnoDB");

/* データを挿入します */
mysqli_query($link, "INSERT INTO Language VALUES ('DEU', 'Bavarian', 'F', 11.2)");
mysqli_query($link, "INSERT INTO Language VALUES ('DEU', 'Swabian', 'F', 9.4)");

/* トランザクションをコミットします */
mysqli_commit($link);

/* 接続を閉じます */
mysqli_close($link);
?>
```

参考

- mysqli_autocommit()
- mysqli_rollback()

---

mysqli_connect_errno

(PHP 5)

mysqli_connect_errno — 直近の接続コールに関するエラーコードを返す

説明

int **mysqli_connect_errno** (void)

直近の mysqli_connect() コールのエラー番号を返します。

> **注意**: クライアントのエラーメッセージ番号は MySQL の errmsg.h ヘッダファイルで、そしてサーバのエラーメッセージ番号は mysqld_error.h で定義されています。MySQL のソース配布の中には、エラーメッセージの 完全なリストが Docs/mysqld_error.txt に含まれています。

返り値

直近の mysqli_connect() コールが失敗した場合、 エラーコードを返します。ゼロは、何もエラーが発生しなかったことを示します。

例

Example#1 mysqli_connect_errno() の例

```
<?php

$link = @mysqli_connect("localhost", "nonexisting_user", "");

if (!$link) {
    printf("Can't connect to localhost. Errorcode: %d¥n", mysqli_connect_errno());
}
?>
```

参考

- mysqli_connect()
- mysqli_connect_error()
- mysqli_errno()
- mysqli_error()
- mysqli_sqlstate()

---

mysqli_connect_error

(PHP 5)

mysqli_connect_error — 直近の接続エラーの内容を文字列で返す

説明

string **mysqli_connect_error** (void)

mysqli_connect() が指すデータベースの 直近のエラーについての文字列表現を返します。

返り値

エラーの内容を表す文字列を返します。エラーが発生しなかった場合は 空文字列を返します。

例

Example#1 mysqli_connect_error() の例

```php
<?php

$link = @mysqli_connect("localhost", "nonexisting_user", "");

if (!$link) {
    printf("Can't connect to localhost. Error: %s¥n", mysqli_connect_error());
}
?>
```

参考

- mysqli_connect()
- mysqli_connect_errno()
- mysqli_errno()
- mysqli_error()
- mysqli_sqlstate()

---

mysqli_connect

mysqli->__construct()

(No version information available, might be only in CVS)

mysqli->__construct() — 新規に MySQL サーバへの接続をオープンする

説明

手続き型

mysqli **mysqli_connect** ([string $host [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket]]]]]])

オブジェクト指向型（コンストラクタ）:

**mysqli
__construct** ([string $host [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket]]]]]])

実行中の MySQL サーバへの接続をオープンします。

パラメータ

host

ホスト名または IP アドレスです。この引数に **NULL** または "localhost" を渡すと ローカルホストとみなされます。もし可能な場合、TCP/IP プロトコルの代わりに パイプが使用されます。

username

MySQL のユーザ名。

passwd

省略したり **NULL** を渡したりした場合、MySQL サーバは パスワードを持たないユーザレコードについてのみ認証を試みます。 これによってひとつのユーザ名において(パスワードが指定されたか 否かによって)違うパーミッションを与えることができます。

dbname

指定した場合は、 クエリが行われるデフォルトのデータベースとなります。

port

MySQL サーバに接続する際のポート番号を指定します。

socket

使用するソケットあるいは名前つきパイプを指定します。

> **注意**: socket 引数を指定しても、MySQL サーバへの 接続時の型を明示的に定義することにはなりません。MySQL サーバへの 接続方法については host 引数で定義されます。

返り値

MySQL サーバへの接続を表すオブジェクトを返します。接続に失敗した場合には **FALSE** を返します。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続の状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}
```

```
printf("Host information: %s¥n", $mysqli->host_info);

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続の状況をチェックします */
if (!$link) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

printf("Host information: %s¥n", mysqli_get_host_info($link));

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Host information: Localhost via UNIX socket
```

注意

> **注意**: エラー "Can't create TCP/IP socket (10106)" が発生するのは、たいていは variables_order 設定ディレクティブに E が含まれていない場合です。 Windows では、これが含まれていなければ SYSTEMROOT 環境変数が使用できず、PHP が Winsock の読み込みに失敗します。

---

mysqli_data_seek

result->data_seek()

(No version information available, might be only in CVS)

result->data_seek() — 結果の任意の行にポインタを移動する

説明

手続き型:

```
bool mysqli_data_seek ( mysqli_result $result , int $offset )
```

オブジェクト指向型（メソッド）:

```
mysqli_result
bool data_seek ( int $offset )
```

mysqli_data_seek() 関数は、 結果セットの任意の行 offset にポインタを移動します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

offset

フィールドオフセット。ゼロから全行数 - 1 までの間 （0..mysqli_num_rows() - 1）である必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数は、mysqli_store_result() あるいは mysqli_query() によって取得した、 バッファに格納されている結果に対してのみ使用可能です。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
```

```
if ($result = $mysqli->query( $query)) {

    /* 行番号 400 に移動します */
    $result->data_seek(399);

    /* 行を取得します */
    $row = $result->fetch_row();

    printf ("City: %s  Countrycode: %s¥n", $row[0], $row[1]);

    /* 結果セットを開放します */
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続を開きます */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (!$link) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
     exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";

if ($result = mysqli_query($link, $query)) {

    /* 行番号 400 に移動します */
    mysqli_data_seek($result, 399);

    /* 行を取得します */
    $row = mysqli_fetch_row($result);

    printf ("City: %s  Countrycode: %s¥n", $row[0], $row[1]);

    /* 結果セットを開放します */
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
City: Benin City  Countrycode: NGA
```

参考

- mysqli_store_result()
- mysqli_fetch_row()
- mysqli_fetch_array()
- mysqli_fetch_assoc()
- mysqli_fetch_object()
- mysqli_query()
- mysqli_num_rows()

---

mysqli_debug

mysqli->debug()

(PHP 5)

mysqli->debug() — デバッグ操作を行う

説明

手続き型:

```
bool mysqli_debug ( string $message )
```

オブジェクト指向型 (メソッド):

```
mysqli
bool debug ( string $message )
```

Fred Fish debugging library を使用してデバッグを行います。

パラメータ

message

実行するデバッグ操作を表す文字列。

返り値

Returns ***TRUE****.*

注意

> ***注意****:* ***mysqli_debug()*** *関数を使用するには、 MySQL クライアントライブラリを、デバッグ機能を有効にしてコンパイルする 必要があります。*

例

Example#1 トレースファイルの作成

```
<?php

/* ローカル（クライアント）マシンの '/tmp/client.trace' に
   トレースファイルを作成します */
mysqli_debug("d:t:0,/tmp/client.trace");

?>
```

参考

- *mysqli_dump_debug_info()*
- *mysqli_report()*

---

mysqli_disable_reads_from_master

mysqli->disable_reads_from_master()

(PHP 5)

mysqli->disable_reads_from_master() — マスタからの読み込みを無効にする

説明

手続き型:

bool ***mysqli_disable_reads_from_master*** *(mysqli $link)*

オブジェクト指向型（メソッド）:

mysqli
void ***disable_reads_from_master*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_disable_rpl_parse

(PHP 5)

mysqli_disable_rpl_parse — RPL のパースを無効にする

説明

bool ***mysqli_disable_rpl_parse*** *(mysqli $link)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_dump_debug_info

mysqli->dump_debug_info()

(PHP 5)

mysqli->dump_debug_info() — デバッグ情報をログに出力する

説明

Procedural style:

bool ***mysqli_dump_debug_info*** *(mysqli $link)*

Object oriented style (method):

mysqli
bool **dump_debug_info** (void)

この関数は SUPER 権限を持ったユーザによって実行されるされることを 想定しており、link で指定した接続に関連する MySQL サーバのデバッグ情報をログに出力します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mysqli_debug()

---

mysqli_embedded_server_end

(PHP 5 >= 5.1.0)

mysqli_embedded_server_end —

説明

void **mysqli_embedded_server_end** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_embedded_server_start

(PHP 5 >= 5.1.0)

mysqli_embedded_server_start —

説明

bool **mysqli_embedded_server_start** (bool $start , array $arguments , array $groups)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_enable_reads_from_master

(PHP 5)

mysqli_enable_reads_from_master — マスタからの読み込みを有効にする

説明

bool **mysqli_enable_reads_from_master** (mysqli $link)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_enable_rpl_parse

(PHP 5)

mysqli_enable_rpl_parse — RPL のパースを有効にする

説明

bool **mysqli_enable_rpl_parse** (mysqli $link)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_errno

mysqli->errno

(PHP 5)

mysqli->errno — 直近の関数コールによるエラーコードを返す

説明

手続き型:

int **mysqli_errno** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
int$errno;

直近の MySQLi 関数のコールが成功あるいは失敗した際のエラーコードを返します。

クライアントのエラーメッセージ番号は MySQL の errmsg.h ヘッダファイルで、そしてサーバのエラーメッセージ番号は mysqld_error.h で定義されています。MySQL のソース配布の中には、エラーメッセージの 完全なリストが Docs/mysqld_error.txt に含まれています。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

直近のコールが失敗した場合、エラーコードを返します。 ゼロは、何もエラーが発生しなかったことを示します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if (!$mysqli->query("SET a=1")) {
    printf("Errorcode: %d¥n", $mysqli->errno);
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if (!mysqli_query($link, "SET a=1")) {
    printf("Errorcode: %d¥n", mysqli_errno($link));
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Errorcode: 1193
```

参考

- mysqli_connect_errno()
- mysqli_connect_error()
- mysqli_error()
- mysqli_sqlstate()

---

mysqli_error

mysqli->error

(PHP 5)

mysqli->error — 直近のエラーの内容を文字列で返す

説明

手続き型:

string **mysqli_error** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
string$error;

直近の MySQLi 関数のコールが成功あるいは失敗した際のエラーメッセージを返します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

エラーの内容を表す文字列を返します。エラーが発生しなかった場合は空文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if (!$mysqli->query("SET a=1")) {
    printf("Errormessage: %s¥n", $mysqli->error);
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if (!mysqli_query($link, "SET a=1")) {
    printf("Errormessage: %s¥n", mysqli_error($link));
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Errormessage: Unknown system variable 'a'
```

参考

- mysqli_connect_errno()
- mysqli_connect_error()
- mysqli_errno()
- mysqli_sqlstate()

---

mysqli_escape_string

(PHP 5)

mysqli_escape_string — mysqli_real_escape_string() のエイリアス

説明

この関数は mysqli_real_escape_string() のエイリアスです。

参考

- mysqli_real_escape_string()

---

mysqli_execute

(PHP 5)

mysqli_execute — mysqli_stmt_execute() のエイリアス

説明

この関数は mysqli_stmt_execute() のエイリアスです。

注意

注意: **mysqli_execute()** は非推奨で、廃止予定です。

参考

- mysqli_stmt_execute()

---

mysqli_fetch_array

result->fetch_array()

(No version information available, might be only in CVS)

result->fetch_array() — 結果の行を連想配列・数値添字配列あるいはその両方の形式で取得する

説明

手続き型:

```
mixed mysqli_fetch_array ( mysqli_result $result [, int $resulttype ] )
```

オブジェクト指向型（メソッド）:

```
mysqli_result
mixed fetch_array ([ int $resulttype ] )
```

取得した行に対応する配列を返します。もし result が指す結果にもう行がない場合は **NULL** を返します。

mysqli_fetch_array() は mysqli_fetch_row() 関数の拡張版です。データを数値添字の 配列に格納することに加えて、
mysqli_fetch_array() 関数は、フィールド名をキーとする連想配列にもデータを格納します。

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

もし 2 つ以上のカラムが同じフィールド名であった場合は、最後に現れた カラムが優先され、以前のデータを上書きします。同名の複数のカラムに
アクセスする場合、数値添字版の行データを使用しなければなりません。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

resulttype

このオプションは、 結果の行データから返す配列の型を指定します。ここで指定可能な値は 定数 **MYSQLI_ASSOC**、 **MYSQLI_NUM** あるいは
MYSQLI_BOTH. のいずれかです。デフォルトは **MYSQLI_BOTH** です。

MYSQLI_ASSOC 定数を指定すると、この関数は mysqli_fetch_assoc() と同じ結果を返します。一方 **MYSQLI_NUM** を指定すると、
mysqli_fetch_row() 関数と同じ結果となります。最後の **MYSQLI_BOTH** を指定すると、 ひとつの配列にこれら両方の属性を含めます。

返り値

取得した行に対応する文字列の配列を返します。結果セットにもう行がない場合には **NULL** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
```

```
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID LIMIT 3";
$result = $mysqli->query($query);

/* 数値添字配列 */
$row = $result->fetch_array(MYSQLI_NUM);
printf ("%s (%s)¥n", $row[0], $row[1]);

/* 連想配列 */
$row = $result->fetch_array(MYSQLI_ASSOC);
printf ("%s (%s)¥n", $row["Name"], $row["CountryCode"]);

/* 連想配列および数値添字配列 */
$row = $result->fetch_array(MYSQLI_BOTH);
printf ("%s (%s)¥n", $row[0], $row["CountryCode"]);

/* 結果セットを開放します */
$result->close();

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID LIMIT 3";
$result = mysqli_query($link, $query);

/* 数値添字配列 */
$row = mysqli_fetch_array($result, MYSQLI_NUM);
printf ("%s (%s)¥n", $row[0], $row[1]);

/* 連想配列 */
$row = mysqli_fetch_array($result, MYSQLI_ASSOC);
printf ("%s (%s)¥n", $row["Name"], $row["CountryCode"]);

/* 連想配列および数値添字配列 */
$row = mysqli_fetch_array($result, MYSQLI_BOTH);
printf ("%s (%s)¥n", $row[0], $row["CountryCode"]);

/* 結果セットを開放します */
mysqli_free_result($result);

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Kabul (AFG)
Qandahar (AFG)
Herat (AFG)
```

参考

- *mysqli_fetch_assoc()*
- *mysqli_fetch_row()*
- *mysqli_fetch_object()*
- *mysqli_query()*
- *mysqli_data_seek()*

---

mysqli_fetch_assoc

mysqli->fetch_assoc()

(PHP 5)

mysqli->fetch_assoc() — 結果の行を連想配列で取得する

説明

手続き型:

*array **mysqli_fetch_assoc** (mysqli_result $result)*

オブジェクト指向型（メソッド）:

mysqli_result
array **fetch_assoc** (void)

取得した行に対応する連想配列を返します。もしもう行がない場合には **NULL** を返します。

> **注意**: この関数により返されるフィールド名は 大文字小文字を区別 します。
>
> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

取得した行に対応する文字列の連想配列を返します。 連想配列の各キーが、結果セットのカラムを表します。 結果セットにもう行がない場合には **NULL** を返します。

もし 2 つ以上のカラムが同じフィールド名であった場合は、最後に現れた カラムが優先され、以前のデータを上書きします。同名の複数のカラムにアクセスする場合、mysqli_fetch_row() を用いて 数値添字配列を使用するか、あるいはカラム名にエイリアスを指定する 必要があります。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

    /* 連想配列を取得します */
    while ($row = $result->fetch_assoc()) {
        printf ("%s (%s)¥n", $row["Name"], $row["CountryCode"]);
    }

    /* 結果セットを開放します */
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

    /* 連想配列を取得します */
    while ($row = mysqli_fetch_assoc($result)) {
        printf ("%s (%s)¥n", $row["Name"], $row["CountryCode"]);
    }

    /* 結果セットを開放します */
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

参考

- mysqli_fetch_array()
- mysqli_fetch_row()

- mysqli_fetch_object()
- mysqli_query()
- mysqli_data_seek()

---

mysqli_fetch_field_direct

result->fetch_field_direct()

(No version information available, might be only in CVS)

result->fetch_field_direct() — 単一のフィールドのメタデータを取得する

説明

手続き型:

object **mysqli_fetch_field_direct** (mysqli_result $result , int $fieldnr)

オブジェクト指向型（メソッド）:

mysqli_result
object **fetch_field_direct** (int $fieldnr)

指定した結果セットから、フィールド定義情報を含むオブジェクトを返します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

fieldnr

フィールド番号。この値は 0 から フィールド数 - 1 までの範囲でなければなりません。

返り値

フィールド定義情報を含むオブジェクトを返します。もし、指定した fieldnr のフィールドの情報が取得できない場合は **FALSE** を返します。

オブジェクトの属性

属性	説明
name	カラムの名前。
orgname	もしエイリアスが指定されている場合の、本来の名前。
table	フィールドが属するテーブルの名前。
orgtable	もしエイリアスが指定されている場合の、本来のテーブル名。
def	フィールドのデフォルト値。文字列形式。
max_length	結果セットにおけるフィールドの最大幅。
length	テーブルの定義で指定されているフィールド幅。
charsetnr	フィールドの文字セット番号。
flags	フィールドのビットフラグを整数型で表す。
type	フィールドのデータ型。
decimals	フィールドの桁数（integer 型のフィールド）。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Name LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* カラム 'SurfaceArea' のフィールド情報を取得します */
    $finfo = $result->fetch_field_direct(1);

    printf("Name:      %s¥n", $finfo->name);
    printf("Table:     %s¥n", $finfo->table);
    printf("max. Len:  %d¥n", $finfo->max_length);
    printf("Flags:     %d¥n", $finfo->flags);
```

```
    printf("Type:      %d¥n", $finfo->type);

    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
     exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Name LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* カラム 'SurfaceArea' のフィールド情報を取得します */
    $finfo = mysqli_fetch_field_direct($result, 1);

    printf("Name:      %s¥n", $finfo->name);
    printf("Table:     %s¥n", $finfo->table);
    printf("max. Len:  %d¥n", $finfo->max_length);
    printf("Flags:     %d¥n", $finfo->flags);
    printf("Type:      %d¥n", $finfo->type);

    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Name:       SurfaceArea
Table:      Country
max. Len:  10
Flags:     32769
Type:      4
```

参考

- mysqli_num_fields()
- mysqli_fetch_field()
- mysqli_fetch_fields()

---

mysqli_fetch_field

result->fetch_field()

(No version information available, might be only in CVS)

result->fetch_field() — 結果セットの次のフィールドを返す

説明

手続き型:

```
object mysqli_fetch_field ( mysqli_result $result )
```

オブジェクト指向型（メソッド）:

```
mysqli_result
object fetch_field ( void )
```

結果セットから ひとつのカラムの情報をオブジェクトとして返します。この関数を 繰り返しコールすることで、結果セットのすべてのカラムについての情報が 取得可能です。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

フィールド定義情報を含むオブジェクトを返します。もし フィールドの情報が取得できない場合は、FALSE を返します。

オブジェクトのプロパティ

プロパティ	説明
name	カラムの名前。
orgname	もしエイリアスが指定されている場合の、本来の名前。
table	フィールドが属するテーブルの名前。
orgtable	もしエイリアスが指定されている場合の、本来のテーブル名。
def	フィールドのデフォルト値。文字列形式。
max_length	結果セットにおけるフィールドの最大幅。
length	テーブルの定義で指定されているフィールド幅。
charsetnr	フィールドの文字セット番号。
flags	フィールドのビットフラグを整数型で表す。
type	フィールドのデータ型。
decimals	フィールドの桁数（integer 型のフィールド）。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* すべてのカラムのフィールド情報を取得します */
    while ($finfo = $result->fetch_field()) {

        printf("Name:      %s¥n", $finfo->name);
        printf("Table:     %s¥n", $finfo->table);
        printf("max. Len:  %d¥n", $finfo->max_length);
        printf("Flags:     %d¥n", $finfo->flags);
        printf("Type:      %d¥n¥n", $finfo->type);
    }
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* すべてのカラムのフィールド情報を取得します */
    while ($finfo = mysqli_fetch_field($result)) {

        printf("Name:      %s¥n", $finfo->name);
        printf("Table:     %s¥n", $finfo->table);
        printf("max. Len:  %d¥n", $finfo->max_length);
        printf("Flags:     %d¥n", $finfo->flags);
        printf("Type:      %d¥n¥n", $finfo->type);
    }
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Name:      Name
Table:     Country
max. Len:  11
Flags:     1
Type:      254

Name:      SurfaceArea
```

```
Table:      Country
max. Len:  10
Flags:     32769
Type:      4
```

参考

- mysqli_num_fields()
- mysqli_fetch_field_direct()
- mysqli_fetch_fields()
- mysqli_field_seek()

---

mysqli_fetch_fields

result->fetch_fields()

(No version information available, might be only in CVS)

result->fetch_fields() — 結果セットのフィールド情報をオブジェクトの配列で返す

説明

手続き型:

```
array mysqli_fetch_fields ( mysqli_result $result )
```

オブジェクト指向型（メソッド）:

```
mysqli_result
array fetch_fields ( void )
```

この関数は mysqli_fetch_field() 関数と同じ目的で 使用しますが、ひとつ違いがあります。一度にひとつずつフィールド情報を 取得するのではなく、複数のカラムの情報をオブジェクトの配列で返します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

フィールド定義情報を含むオブジェクトの配列を返します。もし フィールドの情報が取得できない場合は、FALSE を返します。

オブジェクトのプロパティ

プロパティ	説明
name	カラムの名前。
orgname	もしエイリアスが指定されている場合の、本来の名前。
table	フィールドが属するテーブルの名前。
orgtable	もしエイリアスが指定されている場合の、本来のテーブル名。
def	フィールドのデフォルト値。文字列形式。
max_length	結果セットにおけるフィールドの最大幅。
length	テーブルの定義で指定されているフィールド幅。
charsetnr	フィールドの文字セット番号。
flags	フィールドのビットフラグを整数型で表す。
type	フィールドのデータ型。
decimals	フィールドの桁数（integer 型のフィールド）。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";
```

```
if ($result = $mysqli->query($query)) {

    /* すべてのカラムのフィールド情報を取得します */
    $finfo = $result->fetch_fields();

    foreach ($finfo as $val) {
        printf("Name:       %s¥n", $val->name);
        printf("Table:      %s¥n", $val->table);
        printf("max. Len:  %d¥n", $val->max_length);
        printf("Flags:      %d¥n", $val->flags);
        printf("Type:       %d¥n¥n", $val->type);
    }
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* すべてのカラムのフィールド情報を取得します */
    $finfo = mysqli_fetch_fields($result);

    foreach ($finfo as $val) {
        printf("Name:       %s¥n", $val->name);
        printf("Table:      %s¥n", $val->table);
        printf("max. Len:  %d¥n", $val->max_length);
        printf("Flags:      %d¥n", $val->flags);
        printf("Type:       %d¥n¥n", $val->type);
    }
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Name:       Name
Table:      Country
max. Len:  11
Flags:     1
Type:       254

Name:       SurfaceArea
Table:      Country
max. Len:  10
Flags:     32769
Type:       4
```

参考

- mysqli_num_fields()
- mysqli_fetch_field_direct()
- mysqli_fetch_field()

---

mysqli_fetch_lengths

result->lengths()

(No version information available, might be only in CVS)

result->lengths() — 結果セットにおける現在の行のカラムの長さを返す

説明

手続き型:

array **mysqli_fetch_lengths** (mysqli_result $result)

オブジェクト指向型（プロパティ）:

mysqli_result
array$lengths;

mysqli_fetch_lengths() 関数は、result が指す結果セットの現在の行について、すべてのカラムの長さを含む 配列を返します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

各カラムのサイズ（終端の null 文字は含みません）を表す整数の配列を 返します。エラー時には **FALSE** を返します。

mysqli_fetch_lengths() は、結果セットの現在の行に ついてのみ有効です。mysqli_fetch_row/array/object をコールする前、あるいは 結果のすべての行を取得した後にこの関数をコールすると、**FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT * from Country ORDER BY Code LIMIT 1";

if ($result = $mysqli->query($query)) {

    $row = $result->fetch_row();

    /* カラムの長さを表示します */
    foreach ($result->lengths as $i => $val) {
        printf("Field %2d has Length %2d¥n", $i+1, $val);
    }
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT * from Country ORDER BY Code LIMIT 1";

if ($result = mysqli_query($link, $query)) {

    $row = mysqli_fetch_row($result);

    /* カラムの長さを表示します */
    foreach (mysqli_fetch_lengths($result) as $i => $val) {
        printf("Field %2d has Length %2d¥n", $i+1, $val);
    }
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Field  1 has Length  3
Field  2 has Length  5
Field  3 has Length 13
Field  4 has Length  9
Field  5 has Length  6
Field  6 has Length  1
Field  7 has Length  6
Field  8 has Length  4
Field  9 has Length  6
Field 10 has Length  6
Field 11 has Length  5
Field 12 has Length 44
Field 13 has Length  7
Field 14 has Length  3
Field 15 has Length  2
```

mysqli_fetch_object

result->fetch_object()

(No version information available, might be only in CVS)

result->fetch_object() — 結果セットの現在の行をオブジェクトとして返す

説明

手続き型:

object **mysqli_fetch_object** (mysqli_result $result [, string $class_name [, array $params]])

オブジェクト指向型（メソッド）:

mysqli_result
object **fetch_object** ([string $class_name [, array $params]])

mysqli_fetch_object() は、結果セットの現在の行を オブジェクトで返します。このオブジェクトは、結果セットのフィールド名を 属性として持ちます。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

class_name

params

返り値

取得した行に対応する文字列プロパティを有するオブジェクトを返します。 もし結果セットにもう行がない場合には **NULL** を返します。

注意: この関数により返されるフィールド名は 大文字小文字を区別 します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

変更履歴

バージョン	説明
5.0.0	さまざまな型のオブジェクトで結果を返せるようになりました。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

    /* オブジェクトの配列を取得します */
    while ($obj = $result->fetch_object()) {
        printf ("%s (%s)¥n", $obj->Name, $obj->CountryCode);
    }

    /* 結果セットを開放します */
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

    /* オブジェクトの配列を取得します */
    while ($obj = mysqli_fetch_object($result)) {
```

```
        printf ("%s (%s)¥n", $obj->Name, $obj->CountryCode);
    }

    /* 結果セットを開放します */
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

参考

- mysqli_fetch_array()
- mysqli_fetch_assoc()
- mysqli_fetch_row()
- mysqli_query()
- mysqli_data_seek()

---

mysqli_fetch_row

result->fetch_row()

(No version information available, might be only in CVS)

result->fetch_row() — 結果の行を数値添字配列で取得する

説明

手続き型:

```
mixed mysqli_fetch_row ( mysqli_result $result )
```

オブジェクト指向型（メソッド）:

```
mysqli_result
mixed fetch_row ( void )
```

結果セットからデータを 1 行取得し、それを数値添字の配列で 返します。各カラムは 0（ゼロ）から始まる添字に格納されます。
mysqli_fetch_row() 関数を続けてコールすると、 結果セットの次の行を返します。もしもう行がない場合には **NULL** を返します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

mysqli_fetch_row() は取得した行に対応する文字列の配列を 返します。もしもう行がない場合には **NULL** を返します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = $mysqli->query($query)) {

    /* 配列を取得します */
    while ($row = $result->fetch_row()) {
        printf ("%s (%s)¥n", $row[0], $row[1]);
    }

    /* 結果セットを開放します */
    $result->close();
}
```

```
/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 50,5";

if ($result = mysqli_query($link, $query)) {

    /* 配列を取得します */
    while ($row = mysqli_fetch_row($result)) {
        printf ("%s (%s)¥n", $row[0], $row[1]);
    }

    /* 結果セットを開放します */
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Pueblo (USA)
Arvada (USA)
Cape Coral (USA)
Green Bay (USA)
Santa Clara (USA)
```

参考

- *mysqli_fetch_array()*
- *mysqli_fetch_assoc()*
- *mysqli_fetch_object()*
- *mysqli_query()*
- *mysqli_data_seek()*

---

mysqli_fetch

(PHP 5)

mysqli_fetch — mysqli_stmt_fetch() のエイリアス

説明

この関数は mysqli_stmt_fetch() のエイリアスです。

注意

> ***注意**: **mysqli_fetch()** は非推奨で、廃止予定です。*

参考

- *mysqli_stmt_fetch()*

---

mysqli_field_count

mysqli->field_count

(PHP 5)

mysqli->field_count — 直近のクエリのカラムの数を返す

説明

手続き型:

*int **mysqli_field_count** (mysqli $link)*

オブジェクト指向型（プロパティ）:

mysqli_result
int$field_count;

link が指す接続における直近のクエリの カラムの数を返します。この関数は、クエリが空でない結果セットを 生成すべきなのかそうではないのかを判断するために mysqli_store_result() 関数を使用する際に有用です。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

結果セットのフィールド数を整数で返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

$mysqli->query( "DROP TABLE IF EXISTS friends");
$mysqli->query( "CREATE TABLE friends (id int, name varchar(20))");

$mysqli->query( "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");


$mysqli->real_query($HTTP_POST_VARS['query']);

if ($mysqli->field_count) {
    /* これは select/show あるいは describe クエリです */
    $result = $mysqli->store_result();

    /* 結果セットを処理します */
    $row = $result->fetch_row();

    /* 結果セットを開放します */
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

mysqli_query($link, "DROP TABLE IF EXISTS friends");
mysqli_query($link, "CREATE TABLE friends (id int, name varchar(20))");

mysqli_query($link, "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

mysqli_real_query($link, $HTTP_POST_VARS['query']);

if (mysqli_field_count($link)) {
    /* これは select/show あるいは describe クエリです */
    $result = mysqli_store_result($link);

    /* 結果セットを処理します */
    $row = mysqli_fetch_row($result);

    /* 結果セットを開放します */
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

---

mysqli_field_seek

result->field_seek()

(No version information available, might be only in CVS)

result->field_seek() — 結果ポインタを、指定したフィールドオフセットに設定する

説明

手続き型:

*bool **mysqli_field_seek** (mysqli_result $result , int $fieldnr)*

オブジェクト指向型（メソッド）:

mysqli_result

```
bool field_seek ( int $fieldnr )
```

フィールドカーソルを指定したオフセットに設定します。 mysqli_fetch_field() を次にコールした際に、 設定したオフセットに関連するカラムのフィールド定義を取得します。

注意: 行のはじめに移動するには、オフセットとしてゼロを指定します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

fieldnr

フィールド番号。これは 0 から フィールド数 - 1 までの範囲でなければなりません。

返り値

コール前のフィールドカーソルの値を返します。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* 2 番目のカラムのフィールド情報を取得します */
    $result->field_seek(1);
    $finfo = $result->fetch_field();

    printf("Name:      %s¥n", $finfo->name);
    printf("Table:     %s¥n", $finfo->table);
    printf("max. Len:  %d¥n", $finfo->max_length);
    printf("Flags:     %d¥n", $finfo->flags);
    printf("Type:      %d¥n¥n", $finfo->type);

    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* 2 番目のカラムのフィールド情報を取得します */
    mysqli_field_seek($result, 1);
    $finfo = mysqli_fetch_field($result);

    printf("Name:      %s¥n", $finfo->name);
    printf("Table:     %s¥n", $finfo->table);
    printf("max. Len:  %d¥n", $finfo->max_length);
    printf("Flags:     %d¥n", $finfo->flags);
    printf("Type:      %d¥n¥n", $finfo->type);

    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Name:       SurfaceArea
Table:      Country
max. Len:  10
Flags:     32769
Type:      4
```

参考

- mysqli_fetch_field()

---

mysqli_field_tell

result->current_field

(No version information available, might be only in CVS)

result->current_field — 結果ポインタにおける現在のフィールドオフセットを取得する

説明

手続き型:

int **mysqli_field_tell** (mysqli_result $result)

オブジェクト指向型（プロパティ）:

mysqli_result
int$current_field ;

直近の mysqli_fetch_field() のコールで使用した フィールドカーソルの位置を返します。この値は mysqli_field_seek() の引数として使用可能です。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

フィールドカーソルの現在のオフセットを返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = $mysqli->query($query)) {

    /* すべてのカラムのフィールド情報を取得します */
    while ($finfo = $result->fetch_field()) {

        /* フィールドポインタのオフセットを取得します */
        $currentfield = $result->current_field;

        printf("Column %d:¥n", $currentfield);
        printf("Name:      %s¥n", $finfo->name);
        printf("Table:     %s¥n", $finfo->table);
        printf("max. Len:  %d¥n", $finfo->max_length);
        printf("Flags:     %d¥n", $finfo->flags);
        printf("Type:      %d¥n¥n", $finfo->type);
    }
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, SurfaceArea from Country ORDER BY Code LIMIT 5";

if ($result = mysqli_query($link, $query)) {

    /* すべてのカラムのフィールド情報を取得します */
    while ($finfo = mysqli_fetch_field($result)) {
```

```
        /* フィールドポインタのオフセットを取得します */
        $currentfield = mysqli_field_tell($result);

        printf("Column  %d:¥n", $currentfield);
        printf("Name:       %s¥n", $finfo->name);
        printf("Table:      %s¥n", $finfo->table);
        printf("max. Len:  %d¥n", $finfo->max_length);
        printf("Flags:      %d¥n", $finfo->flags);
        printf("Type:       %d¥n¥n", $finfo->type);
    }
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Column  1:
Name:       Name
Table:      Country
max. Len:  11
Flags:      1
Type:       254

Column  2:
Name:       SurfaceArea
Table:      Country
max. Len:  10
Flags:      32769
Type:       4
```

参考

- mysqli_fetch_field()
- mysqli_field_seek()

---

mysqli_free_result

result->free()

(No version information available, might be only in CVS)

result->free() — 結果に関連付けられたメモリを開放する

説明

手続き型:

```
void mysqli_free_result ( mysqli_result $result )
```

オブジェクト指向型（すべてのメソッドが等価）:

```
mysqli_result
void free ( void )
void close ( void )
void free_result ( void )
```

結果に関連付けられたメモリを開放します。

注意: 結果オブジェクトが必要なくなった場合は、常に **mysqli_free_result()** でメモリを開放すべきです。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

値を返しません。

参考

- mysqli_query()
- mysqli_stmt_store_result()
- mysqli_store_result()
- mysqli_use_result()

---

mysqli_get_charset

(PHP 5 >= 5.1.0)

mysqli_get_charset — 文字セットオブジェクトを返す

説明

object **mysqli_get_charset** (mysqli $link)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_get_client_info

(PHP 5)

mysqli_get_client_info — MySQL クライアントのバージョンを文字列で返す

説明

string **mysqli_get_client_info** (void)

mysqli_get_client_info() 関数は、MySQLi 拡張モジュールが使用しているクライアントのバージョンを文字列で返します。

返り値

MySQL クライアントライブラリのバージョンを文字列で返します。

例

Example#1 mysqli_get_client_info

```
<?php

/* mysql クライアントライブラリのバージョンを
   調べるために、接続する必要はありません */

printf("Client library version: %s¥n", mysqli_get_client_info());
?>
```

参考

- mysqli_get_client_version()
- mysqli_get_server_info()
- mysqli_get_server_version()

---

mysqli_get_client_version

(PHP 5)

mysqli_get_client_version — MySQL クライアント情報を取得する

説明

int **mysqli_get_client_version** (void)

クライアントのバージョン番号を整数値で返します。

返り値

MySQL クライアントライブラリのバージョンを、以下の書式で返します。 main_version*10000 + minor_version *100 + sub_version 例えば、4.1.0 は 40100 となります。

これは、ある機能が使用可能かどうかを知るため、クライアントライブラリの バージョンを手っ取り早く調べたい場合に有用です。

例

Example#1 mysqli_get_client_version

```
<?php

/* mysql クライアントライブラリのバージョンを
   調べるために、接続する必要はありません */

printf("Client library version: %d¥n", mysqli_get_client_version());
?>
```

参考

- mysqli_get_client_info()
- mysqli_get_server_info()
- mysqli_get_server_version()

---

mysqli_get_host_info

mysqli->host_info

(No version information available, might be only in CVS)

mysqli->host_info — 使用している接続の型を文字列で返す

説明

手続き型:

string **mysqli_get_host_info** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
string$host_info;

mysqli_get_host_info() は、link が使用している接続の情報を文字列で返します（サーバのホスト名を 含みます）。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

サーバのホスト名と接続の型を文字列で返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* ホスト情報を表示します */
printf("Host info: %s¥n", $mysqli->host_info);

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* ホスト情報を表示します */
printf("Host info: %s¥n", mysqli_get_host_info($link));

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Host info: Localhost via UNIX socket
```

参考

- mysqli_get_proto_info()

---

mysqli_get_metadata

(PHP 5)

mysqli_get_metadata — mysqli_stmt_result_metadata() のエイリアス

説明

この関数は mysqli_stmt_result_metadata() のエイリアスです。

注意

注意: **mysqli_get_metadata()** は非推奨で、廃止予定です。

参考

- mysqli_stmt_result_metadata()

---

mysqli_get_proto_info

mysqli->protocol_version

(No version information available, might be only in CVS)

mysqli->protocol_version — 使用している MySQL プロトコルのバージョンを返す

説明

手続き型:

int **mysqli_get_proto_info** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
string$protocol_version;

link で表される接続で使用している MySQL プロトコルのバージョンを整数値で返します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

プロトコルバージョンを整数値で返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* プロトコルのバージョンを表示します */
printf("Protocol version: %d¥n", $mysqli->protocol_version);

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* プロトコルのバージョンを表示します */
printf("Protocol version: %d¥n", mysqli_get_proto_info($link));

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Protocol version: 10
```

参考

- mysqli_get_host_info()

mysqli_get_server_info

mysqli->server_info

(No version information available, might be only in CVS)

mysqli->server_info — MySQL サーバのバージョンを返す

説明

手続き型:

string **mysqli_get_server_info** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
string$server_info;

MySQLi 拡張モジュールが接続している MySQL サーバのバージョンを文字列で返します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

サーバのバージョンを文字列で返します。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* サーバのバージョンを表示します */
printf("Server version: %s¥n", $mysqli->server_info);

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* サーバのバージョンを表示します */
printf("Server version: %s¥n", mysqli_get_server_info($link));

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Server version: 4.1.2-alpha-debug
```

参考

- mysqli_get_client_info()
- mysqli_get_client_version()
- mysqli_get_server_version()

mysqli_get_server_version

mysqli->server_version

(No version information available, might be only in CVS)

mysqli->server_version — MySQL サーバのバージョンを整数値で返す

説明

手続き型:

int **mysqli_get_server_version** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
int$server_version;

mysqli_get_server_version() 関数は 接続している（link で表される）サーバの バージョンを整数値で返します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

サーバのバージョンを整数値で返します。

バージョン番号の書式は以下のようになります。 main_version * 10000 + minor_version * 100 + sub_version （すなわち、バージョン 4.1.0 は 40100 となります）

例

Example#1 オブジェクト指向

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password");

/* 接続の状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* サーバのバージョンを表示します */
printf("Server version: %d¥n", $mysqli->server_version);

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password");

/* 接続の状況をチェックする */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* サーバのバージョンを表示する */
printf("Server version: %d¥n", mysqli_get_server_version($link));

/* 接続を閉じる */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Server version: 40102
```

参考

- mysqli_get_client_info()
- mysqli_get_client_version()
- mysqli_get_server_info()

---

mysqli_get_warnings

(PHP 5 >= 5.1.0)

mysqli_get_warnings —

説明

```
object mysqli_get_warnings ( mysqli $link )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_info

mysqli->info

(PHP 5)

mysqli->info — 直近に実行されたクエリの情報を取得する

説明

手続き型:

```
string mysqli_info ( mysqli $link )
```

オブジェクト指向型（プロパティ）

```
mysqli
string$info;
```

mysqli_info() 関数は、直近に実行されたクエリに ついての情報を文字列で返します。文字列の詳細は以下のとおりです。

mysqli_info のとりうる返り値

クエリの型	結果文字列の例
INSERT INTO...SELECT...	Records: 100 Duplicates: 0 Warnings: 0
INSERT INTO...VALUES (...),(...),(...)	Records: 3 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...	Records: 1 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE ...	Records: 3 Duplicates: 0 Warnings: 0
UPDATE ...	Rows matched: 40 Changed: 40 Warnings: 0

注意: 上のどれにも当てはまらない形式のクエリはサポートされません。 そのような場合、**mysqli_info()** は空文字列を返します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

直近に実行されたクエリについての追加情報を文字列で返します。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TEMPORARY TABLE t1 LIKE City");

/* INSERT INTO .. SELECT */
$mysqli->query("INSERT INTO t1 SELECT * FROM City ORDER BY ID LIMIT 150");
printf("%s¥n", $mysqli->info);

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TEMPORARY TABLE t1 LIKE City");
```

```
/* INSERT INTO .. SELECT */
mysqli_query($link, "INSERT INTO t1 SELECT * FROM City ORDER BY ID LIMIT 150");
printf("%s¥n", mysqli_info($link));

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Records: 150  Duplicates: 0  Warnings: 0
```

参考

- mysqli_affected_rows()
- mysqli_warning_count()
- mysqli_num_rows()

---

mysqli_init

(PHP 5)

mysqli_init — MySQLi を初期化し、mysqli_real_connect() で使用するリソースを返す

説明

手続き型:

```
mysqli mysqli_init ( void )
```

オブジェクト指向型 (メソッド):

```
mysqli
mysqli init ( void )
```

mysqli_options() および mysqli_real_connect() で使用可能な MYSQL オブジェクトを割り当て、初期化します。

> **注意**: mysqli_real_connect() がコールされるまで、これ以降の あらゆる mysqli 関数コールは失敗します (mysqli_options() は除きます)。

返り値

オブジェクトを返します。

参考

- mysqli_options()
- mysqli_close()
- mysqli_real_connect()
- mysqli_connect()

---

mysqli_insert_id

mysqli->insert_id

(PHP 5)

mysqli->insert_id — 直近のクエリで使用した自動生成の ID を返す

説明

手続き型:

```
int mysqli_insert_id ( mysqli $link )
```

オブジェクト指向型 (プロパティ):

```
mysqli
int$insert_id;
```

mysqli_insert_id() 関数は、AUTO_INCREMENT 属性を 持つカラムがあるテーブル上でのクエリにより生成された ID を返します。 直近のクエリが INSERT あるいは UPDATE ではなかった場合、あるいは 変更されたテーブルに AUTO_INCREMENT 属性を持つカラムがなかった場合は この関数はゼロを返します。

> **注意**: LAST_INSERT_ID() 関数を使用して INSERT あるいは UPDATE ステートメントを 実行すると、**mysqli_insert_id()** 関数の返す値も 変更されます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

直前のクエリで更新された AUTO_INCREMENT フィールドの値を返します。接続での直前のクエリがない場合や クエリが AUTO_INCREMENT の値を更新しなかった場合は ゼロを返します。

注意: もし数値が int の最大値をこえた場合、**mysqli_insert_id()** は文字列で結果を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

$query = "INSERT INTO myCity VALUES (NULL, 'Stuttgart', 'DEU', 'Stuttgart', 617000)";
$mysqli->query($query);

printf ("New Record has id %d.¥n", $mysqli->insert_id);

/* テーブルを削除します */
$mysqli->query("DROP TABLE myCity");

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

$query = "INSERT INTO myCity VALUES (NULL, 'Stuttgart', 'DEU', 'Stuttgart', 617000)";
mysqli_query($link, $query);

printf ("New Record has id %d.¥n", mysqli_insert_id($link));

/* テーブルを削除します */
mysqli_query($link, "DROP TABLE myCity");

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
New Record has id 1.
```

---

mysqli_kill

mysqli->kill()

(PHP 5)

mysqli->kill() — サーバに MySQL スレッドの停止を問い合わせる

説明

手続き型:

```
bool mysqli_kill ( mysqli $link , int $processid )
```

オブジェクト指向型（メソッド）

```
mysqli
bool kill ( int $processid )
```

この関数は、processid で指定した MySQL スレッドの停止をサーバに問い合わせます。この値は、mysqli_thread_id() 関数で取得したものである 必要があります。

実行中のクエリを停止するには、SQL コマンド KILL QUERY processid を使用する必要があります。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* スレッド ID を取得します */
$thread_id = $mysqli->thread_id;

/* 接続を終了します */
$mysqli->kill($thread_id);

/* これはエラーとなります */
if (!$mysqli->query("CREATE TABLE myCity LIKE City")) {
    printf("Error: %s¥n", $mysqli->error);
    exit;
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* スレッド ID を取得します */
$thread_id = mysqli_thread_id($link);

/* 接続を終了します */
mysqli_kill($link, $thread_id);

/* これはエラーとなります */
if (!mysqli_query($link, "CREATE TABLE myCity LIKE City")) {
    printf("Error: %s¥n", mysqli_error($link));
    exit;
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Error: MySQL server has gone away
```

参考

- mysqli_thread_id()

---

mysqli_master_query

(PHP 5)

mysqli_master_query — マスタ/スレーブ設定で、マスタ側のクエリを実行する

説明

```
bool mysqli_master_query ( mysqli $link , string $query )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_more_results

mysqli->more_results

(PHP 5)

mysqli->more_results — マルチクエリからの結果がまだ残っているかどうかを調べる

説明

bool **mysqli_more_results** (mysqli $link)

直前の mysqli_multi_query() のコール結果に ひとつ以上の結果セットが残っているかどうかを調べます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

mysqli_multi_query() を参照ください。

参考

- mysqli_multi_query()
- mysqli_next_result()
- mysqli_store_result()
- mysqli_use_result()

---

mysqli_multi_query

mysqli->multi_query()

(PHP 5)

mysqli->multi_query() — データベース上でクエリを実行する

説明

手続き型:

bool **mysqli_multi_query** (mysqli $link , string $query)

オブジェクト指向型（メソッド）:

mysqli
bool **multi_query** (string $query)

セミコロンで連結されたひとつまたは複数のクエリを実行します。

最初のクエリの結果セットを取得するには、 mysqli_use_result() あるいは mysqli_store_result() を使用します。その他のクエリの結果は、mysqli_more_results() および mysqli_next_result() を使用して取得します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

query

クエリを表す文字列。

返り値

最初のステートメントが失敗した場合にのみ **FALSE** を返します。 その他のステートメントのエラーを取得するには、まず mysqli_next_result() をコールする必要があります。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
```

```
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if ($mysqli->multi_query($query)) {
    do {
        /* 最初の結果セットを格納します */
        if ($result = $mysqli->store_result()) {
            while ($row = $result->fetch_row()) {
                printf("%s¥n", $row[0]);
            }
            $result->close();
        }
        /* 区切り線を表示します */
        if ($mysqli->more_results()) {
            printf("-----------------¥n");
        }
    } while ($mysqli->next_result());
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query  = "SELECT CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* execute multi query */
if (mysqli_multi_query($link, $query)) {
    do {
        /* 最初の結果セットを格納します */
        if ($result = mysqli_store_result($link)) {
            while ($row = mysqli_fetch_row($result)) {
                printf("%s¥n", $row[0]);
            }
            mysqli_free_result($result);
        }
        /* 区切り線を表示します */
        if (mysqli_more_results($link)) {
            printf("-----------------¥n");
        }
    } while (mysqli_next_result($link));
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は、たとえば以下のようになります。

```
my_user@localhost
-----------------
Amersfoort
Maastricht
Dordrecht
Leiden
Haarlemmermeer
```

参考

- *mysqli_use_result()*
- *mysqli_store_result()*
- *mysqli_next_result()*
- *mysqli_more_results()*

---

mysqli_next_result

mysqli->next_result()

(PHP 5)

mysqli->next_result() — multi_query の、次の結果を準備する

説明

bool **mysqli_next_result** (mysqli $link)

直近の mysqli_multi_query() コールから次の結果セットを用意します。 これは mysqli_store_result() あるいは mysqli_use_result() で取得することが可能です。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

mysqli_multi_query() を参照ください。

参考

- mysqli_multi_query()
- mysqli_more_results()
- mysqli_store_result()
- mysqli_use_result()

---

mysqli_num_fields

result->field_count

(No version information available, might be only in CVS)

result->field_count — 結果のフィールド数を取得する

説明

手続き型:

int **mysqli_num_fields** (mysqli_result $result)

オブジェクト指向型（プロパティ）:

mysqli_result
int$field_count;

指定した結果セットからフィールドの数を返します。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

結果セットのフィールド数を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if ($result = $mysqli->query("SELECT * FROM City ORDER BY ID LIMIT 1")) {

    /* 結果セットのフィールド数を取得します */
    $field_cnt = $result->field_count;

    printf("Result set has %d fields.¥n", $field_cnt);

    /* 結果セットを閉じます */
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if ($result = mysqli_query($link, "SELECT * FROM City ORDER BY ID LIMIT 1")) {

    /* 結果セットのフィールド数を取得します */
    $field_cnt = mysqli_num_fields($result);

    printf("Result set has %d fields.¥n", $field_cnt);

    /* 結果セットを閉じます */
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Result set has 5 fields.
```

参考

- mysqli_fetch_field()

---

mysqli_num_rows

result->num_rows

(No version information available, might be only in CVS)

result->num_rows — 結果の行数を取得する

説明

手続き型:

int **mysqli_num_rows** (mysqli_result $result)

オブジェクト指向型（プロパティ）:

mysqli_result
int$num_rows;

結果セットの行数を返します。

mysqli_num_rows() が使用可能かどうかは、 結果セットをバッファに格納しているかどうかに依存します。結果セットを バッファに格納していない場合、すべての行を取得し終えるまで **mysqli_num_rows()** は正確な行数を返しません。

パラメータ

result

手続き型のみ: mysqli_query()、mysqli_store_result() あるいは mysqli_use_result() が返す結果セット ID。

返り値

結果セットの行数を返します。

注意: 行数が int の最大値をこえる場合、行数は文字列で返されます。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if ($result = $mysqli->query("SELECT Code, Name FROM Country ORDER BY Name")) {

    /* 結果セットの行数を調べます */
    $row_cnt = $result->num_rows;

    printf("Result set has %d rows.¥n", $row_cnt);
```

```
    /* 結果セットを閉じます */
    $result->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if ($result = mysqli_query($link, "SELECT Code, Name FROM Country ORDER BY Name")) {

    /* 結果セットの行数を調べます */
    $row_cnt = mysqli_num_rows($result);

    printf("Result set has %d rows.¥n", $row_cnt);

    /* 結果セットを閉じます */
    mysqli_free_result($result);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Result set has 239 rows.
```

参考

- mysqli_affected_rows()
- mysqli_store_result()
- mysqli_use_result()
- mysqli_query()

---

mysqli_options

mysqli->options()

(PHP 5)

mysqli->options() — オプションを設定する

説明

手続き型:

```
bool mysqli_options ( mysqli $link , int $option , mixed $value )
```

オブジェクト指向型（メソッド）

```
mysqli
bool options ( int $option , mixed $value )
```

接続に関する追加オプションを設定し、 接続の振る舞いに影響を与えるために使用します。

この関数は、いくつかのオプションを設定して複数回コールされることがあります。

mysqli_options() は、 mysqli_init() がコールされた後、 mysqli_real_connect() の前にコールしなければなりません。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

option

指定するオプション。以下の値のいずれかです。

使用可能なオプション

名前	説明
MYSQLI_OPT_CONNECT_TIMEOUT	接続のタイムアウト秒数。
MYSQLI_OPT_LOCAL_INFILE	*LOAD LOCAL INFILE* の使用可/不可。
MYSQLI_INIT_COMMAND	MySQL サーバへの接続後に実行するコマンド。
MYSQLI_READ_DEFAULT_FILE	*my.cnf* の代わりに、指定した名前のファイルから 設定を読み込みます。
MYSQLI_READ_DEFAULT_GROUP	*my.cnf* の指定した名前のグループ、あるいは **MYSQL_READ_DEFAULT_FILE** で指定したファイルから 設定を読み込みます。

value

オプションに設定する値。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

mysqli_real_connect() を参照ください。

参考

- *mysqli_init()*
- *mysqli_real_connect()*

---

mysqli_param_count

(PHP 5)

mysqli_param_count — mysqli_stmt_param_count() のエイリアス

説明

この関数は mysqli_stmt_param_count() のエイリアスです。

注意

> ***注意**: **mysqli_param_count()** は非推奨で、廃止予定です。*

参考

- *mysqli_stmt_param_count()*

---

mysqli_ping

mysqli->ping()

(PHP 5)

mysqli->ping() — サーバとの接続をチェックし、もし切断されている場合は再接続を試みる

説明

手続き型:

*bool **mysqli_ping** (mysqli $link)*

オブジェクト指向型（メソッド）:

mysqli
*bool **ping** (void)*

サーバとの接続が動作中かどうかを確かめます。もし切断されており、 グローバルオプション mysqli.reconnect が有効な場合は 再接続が試みられます。

この関数は、長期間アイドル状態にあるクライアントが、 サーバの状態を確認して必要なら再接続するために使用します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* サーバが稼動中かどうかを確認します */
if ($mysqli->ping()) {
    printf ("Our connection is ok!¥n");
} else {
    printf ("Error: %s¥n", $mysqli->error);
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* サーバが稼動中かどうかを確認します */
if (mysqli_ping($link)) {
    printf ("Our connection is ok!¥n");
} else {
    printf ("Error: %s¥n", mysqli_error($link));
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Our connection is ok!
```

---

mysqli_prepare

mysqli->prepare()

(PHP 5)

mysqli->prepare() — 実行するための SQL ステートメントを準備する

説明

手続き型:

```
mysqli_stmt mysqli_prepare ( mysqli $link , string $query )
```

オブジェクト指向型（メソッド）

```
mysqli
mysqli_stmt prepare ( string $query )
```

null で終わる文字列から SQL クエリを準備し、後でそのステートメントを操作するために使用する ステートメントハンドルを返します。 クエリは、単一の SQL 文である必要があります。

パラメータマーカは、ステートメントの実行や行の取得の前に mysqli_stmt_bind_param() や mysqli_stmt_bind_result() を使用して アプリケーション変数にバインドする必要があります。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

query

クエリを表す文字列。

> **注意**: ステートメントの最後にセミコロンや ¥g を追加してはいけません。

query にはひとつまたは複数のパラメータを 含めることが可能です。そのためには、適切な位置にクエスチョンマーク (?) を埋め込みます。

> **注意**: パラメータのマーカは、それが SQL 文の適切な位置にある場合のみ有効です。 例えば INSERT 文の VALUES() リストの中 (行に登録するカラム値を指定する) や WHERE 句で列のデータと比較する値などが 適切な位置の例です。
> しかし、識別子 (テーブルやカラムの名前) や SELECT 文で選択する項目の名前に指定したり、(等号 = のような) 二項演算子の両側

にパラメータを指定したりすることはできません。 後者の制限は、パラメータの型が判断できなくなることによるものです。 また、パラメータのマーカを NULL と比較して ? IS NULL のようにすることもできません。 一般に、パラメータが使用可能なのはデータ操作言語 (DML) ステートメントであり、データ定義言語 (DDL) ステートメントでは使用できません。

返り値

mysqli_prepare() はステートメントオブジェクトを返します。 エラー時には **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* プリペアドステートメントを作成します */
if ($stmt = $mysqli->prepare("SELECT District FROM City WHERE Name=?")) {

    /* マーカにパラメータをバインドします */
    $stmt->bind_param("s", $city);

    /* クエリを実行します */
    $stmt->execute();

    /* 結果変数をバインドします */
    $stmt->bind_result($district);

    /* 値を取得します */
    $stmt->fetch();

    printf("%s is in district %s¥n", $city, $district);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* プリペアドステートメントを作成します */
if ($stmt = mysqli_prepare($link, "SELECT District FROM City WHERE Name=?")) {

    /* マーカにパラメータをバインドします */
    mysqli_stmt_bind_param($stmt, "s", $city);

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    /* 結果変数をバインドします */
    mysqli_stmt_bind_result($stmt, $district);

    /* 値を取得します */
    mysqli_stmt_fetch($stmt);

    printf("%s is in district %s¥n", $city, $district);

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Amersfoort is in district Utrecht
```

参考

- mysqli_stmt_execute()
- mysqli_stmt_fetch()

- mysqli_stmt_bind_param()
- mysqli_stmt_bind_result()
- mysqli_stmt_close()

---

mysqli_query

mysqli->query()

(PHP 5)

mysqli->query() — データベース上でクエリを実行する

説明

手続き型:

```
mixed mysqli_query ( mysqli $link , string $query [, int $resultmode ] )
```

オブジェクト指向型（メソッド）:

```
mysqli
mixed query ( string $query [, int $resultmode ] )
```

データベースに対してクエリ query を実行します。

機能的には、この関数は mysqli_real_query() に続けて mysqli_use_result() あるいは mysqli_store_result() をコールすることと同等です。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

query

クエリ文字列。

resultmode

定数 MYSQLI_USE_RESULT あるいは MYSQLI_STORE_RESULT で、望みの挙動を指定します。 デフォルトでは MYSQLI_STORE_RESULT を使用します。

MYSQLI_USE_RESULT を使用すると、 mysqli_free_result() をコールするまでは それ以降のコールはすべて Commands out of sync エラーを返します。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。 SELECT, SHOW, DESCRIBE あるいは EXPLAIN の場合は、mysqli_query() は結果オブジェクトを返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* テーブルを作成します。これは結果セットを返しません */
if ($mysqli->query("CREATE TEMPORARY TABLE myCity LIKE City") === TRUE) {
    printf("Table myCity successfully created.¥n");
}

/* Select クエリを実行します。これは結果セットを返します */
if ($result = $mysqli->query("SELECT Name FROM City LIMIT 10")) {
    printf("Select returned %d rows.¥n", $result->num_rows);

    /* 結果セットを開放します */
    $result->close();
}

/* 大量のデータを取得する必要がある場合は MYSQLI_USE_RESULT を使用します */
if ($result = $mysqli->query("SELECT * FROM City", MYSQLI_USE_RESULT)) {

    /* この結果セットが閉じられるまで、サーバとやりとりする関数は
       一切実行できないことに注意しましょう。関数をコールすると、
       'out of sync' エラーが発生します */
    if (!$mysqli->query("SET @a:='this will not work'")) {
        printf("Error: %s¥n", $mysqli->error);
    }
    $result->close();
}

$mysqli->close();
```

```
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* テーブルを作成します。これは結果セットを返しません */
if (mysqli_query($link, "CREATE TEMPORARY TABLE myCity LIKE City") === TRUE) {
    printf("Table myCity successfully created.¥n");
}

/* Select クエリを実行します。これは結果セットを返します */
if ($result = mysqli_query($link, "SELECT Name FROM City LIMIT 10")) {
    printf("Select returned %d rows.¥n", mysqli_num_rows($result));

    /* 結果セットを開放します */
    mysqli_free_result($result);
}

/* 大量のデータを取得する必要がある場合は MYSQLI_USE_RESULT を使用します */
if ($result = mysqli_query($link, "SELECT * FROM City", MYSQLI_USE_RESULT)) {

    /* この結果セットが閉じられるまで、サーバとやりとりする関数は
       一切実行できないことに注意しましょう。関数をコールすると、
       'out of sync' エラーが発生します */
    if (!mysqli_query($link, "SET @a:='this will not work'")) {
        printf("Error: %s¥n", mysqli_error($link));
    }
    mysqli_free_result($result);
}

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Table myCity successfully created.
Select returned 10 rows.
Error: Commands out of sync;  You can't run this command now
```

参考

- mysqli_real_query()
- mysqli_multi_query()
- mysqli_free_result()

---

mysqli_real_connect

mysqli->real_connect()

(PHP 5)

mysqli->real_connect() — mysql サーバとの接続をオープンする

説明

手続き型

```
bool mysqli_real_connect ( mysqli $link [, string $host [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket [, int $flags ]]]]]]] )
```

オブジェクト指向型（メソッド）

```
mysqli
bool real_connect ([ string $host [, string $username [, string $passwd [, string $dbname [, int $port [, string $socket [, int $flags ]]]]]]] )
```

MySQL データベースエンジンとの接続を確立します。

この関数は、以下の点で mysqli_connect() とは異なります。

- **mysqli_real_connect()** は、 mysqli_init() が作成した接続オブジェクトを必要とします。
- mysqli_options() 関数を使用して、 さまざまな接続オプションを設定することが可能です。
- flags パラメータが使用できます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

host

ホスト名あるいは IP アドレス。NULL 値あるいは文字列 "localhost" をこのパラメータに指定すると、ローカルホストを使用します。 使用可能な場合は、TCP/IP プロトコルよりもパイプを優先して使用します。

username

MySQL ユーザ名。

passwd

NULL を指定した場合は、MySQL サーバは パスワードを持たないユーザレコードについてのみ認証を試みます。 これにより、同一のユーザ名に対して (パスワードが指定されたか 否かによって) 違う権限を与えることができます。

dbname

指定した場合は、 クエリが行われるデフォルトのデータベースとなります。

port

MySQL サーバに接続する際のポート番号を指定します。

socket

使用するソケットあるいは名前つきパイプを指定します。

> **注意**: socket 引数を指定しても、MySQL サーバへの 接続時の型を明示的に定義することにはなりません。MySQL サーバへの 接続方法については host 引数で定義されます。

flags

パラメータ flags で、接続時のさまざまなオプションを設定します。

サポートされるフラグ

名前	説明
MYSQLI_CLIENT_COMPRESS	圧縮プロトコルを使用します。
MYSQLI_CLIENT_FOUND_ROWS	変更された行数ではなく、マッチした行数を返します。
MYSQLI_CLIENT_IGNORE_SPACE	関数名に続く空白文字を許可します。すべての関数名を予約語とします。
MYSQLI_CLIENT_INTERACTIVE	接続を閉じるまでのタイムアウト時間として、 (wait_timeout のかわりに) interactive_timeout の使用を許可します。
MYSQLI_CLIENT_SSL	SSL (暗号化) を使用します。

> **注意**: セキュリティの観点から、PHP では MULTI_STATEMENT フラグはサポートされていません。複数のクエリを実行したい場合は、mysqli_multi_query() 関数を使用してください。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 オブジェクト指向型

```
<?php

/* 接続オブジェクトを作成します。まだ接続はしていません */
$mysqli = mysqli_init();

/* 接続オプションを設定します */
$mysqli->options(MYSQLI_INIT_COMMAND, "SET AUTOCOMMIT=0");
$mysqli->options(MYSQLI_OPT_CONNECT_TIMEOUT, 5);

/* サーバに接続します */
$mysqli->real_connect('localhost', 'my_user', 'my_password', 'world');

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

printf ("Connection: %s¥n.", $mysqli->host_info);

$mysqli->close();
?>
```

Example#2 手続き型

```
<?php

/* 接続オブジェクトを作成します。まだ接続はしていません */
$link = mysqli_init();

/* 接続オプションを設定します */
mysqli_options($link, MYSQLI_INIT_COMMAND, "SET AUTOCOMMIT=0");
mysqli_options($link, MYSQLI_OPT_CONNECT_TIMEOUT, 5);

/* サーバに接続します */
mysqli_real_connect($link, 'localhost', 'my_user', 'my_password', 'world');

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
```

```
    exit();
}

printf ("Connection:  %s¥n.", mysqli_get_host_info($link));

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Connection:  Localhost  via  UNIX  socket
```

参考

- mysqli_connect()
- mysqli_init()
- mysqli_options()
- mysqli_ssl_set()
- mysqli_close()

---

mysqli_real_escape_string

mysqli->real_escape_string()

(PHP 5)

mysqli->real_escape_string() — 接続の現在の文字セットを考慮して、SQL 文で使用する文字列の 特殊文字をエスケープする

説明

手続き型:

```
string mysqli_real_escape_string ( mysqli $link , string $escapestr )
```

オブジェクト指向型（どちらのメソッドも等価です）:

```
mysqli
string escape_string ( string $escapestr )
string real_escape_string ( string $escapestr )
```

この関数を使用して、SQL 文中で使用できる正当な形式の SQL 文字列を作成します。 文字列 escapestr が、エスケープされた SQL に変換されます。その際、接続で使用している現在の文字セットが考慮されます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

escapestr

エスケープする文字列。

エンコードされる文字は NUL (ASCII 0), \n, \r, \, ', ", および Control-Z です。

返り値

エスケープ済みの文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TEMPORARY TABLE myCity LIKE City");

$city = "'s Hertogenbosch";

/* このクエリは失敗します。なぜなら $city をエスケープしていないからです */
if (!$mysqli->query("INSERT into myCity (Name) VALUES ('$city')")) {
    printf("Error: %s¥n", $mysqli->sqlstate);
}

$city = $mysqli->real_escape_string($city);

/* $city をエスケープしたので、このクエリは正しく動作します */
```

```
if ($mysqli->query("INSERT into myCity (Name) VALUES ('$city')")) {
    printf("%d Row inserted.¥n", $mysqli->affected_rows);
}

$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TEMPORARY TABLE myCity LIKE City");

$city = "'s Hertogenbosch";

/* このクエリは失敗します。なぜなら $city をエスケープしていないからです */
if (!mysqli_query($link, "INSERT into myCity (Name) VALUES ('$city')")) {
    printf("Error: %s¥n", mysqli_sqlstate($link));
}

$city = mysqli_real_escape_string($link, $city);

/* $city をエスケープしたので、このクエリは正しく動作します */
if (mysqli_query($link, "INSERT into myCity (Name) VALUES ('$city')")) {
    printf("%d Row inserted.¥n", mysqli_affected_rows($link));
}

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Error: 42000
1 Row inserted.
```

参考

- mysqli_character_set_name()

---

mysqli_real_query

mysqli->real_query()

(PHP 5)

mysqli->real_query() — SQL クエリを実行する

説明

手続き型

```
bool mysqli_real_query ( mysqli $link , string $query )
```

オブジェクト指向型（メソッド）:

```
mysqli
bool real_query ( string $query )
```

データベースに対して単一のクエリを実行します。 その結果を取得したり保存したりするには、関数 mysqli_store_result() あるいは mysqli_use_result() を使用します。

指定したクエリが結果を返すかどうかを調べるには、 mysqli_field_count() を参照ください。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

query

クエリを表す文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mysqli_query()

- mysqli_store_result()
- mysqli_use_result()

---

mysqli_report

(PHP 5)

mysqli_report — 内部のレポート関数を有効あるいは無効にする

説明

bool **mysqli_report** (int $flags)

mysqli_report() は、開発やテストのフェーズにおいて クエリの機能を向上させる強力な関数です。フラグの設定により、この関数は インデックスを使用しない（あるいは間違ったインデックスを使用している） mysqli 関数コールやクエリに対してエラーを報告します。

パラメータ

flags

使用できるフラグ

名前	説明
MYSQLI_REPORT_OFF	レポート機能をオフにします。
MYSQLI_REPORT_ERROR	mysqli 関数コールのエラーを報告します。
MYSQLI_REPORT_STRICT	mysqli 関数コールの警告を報告します。
MYSQLI_REPORT_INDEX	クエリでインデックスを使用しない（あるいは 間違ったインデックスを使用している）場合に報告します。
MYSQLI_REPORT_ALL	すべてのオプションを設定します（report all）。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* レポート機能を有効にします */
mysqli_report(MYSQLI_REPORT_ALL);

$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* このクエリはエラーを報告します */
$result = $mysqli->query("SELECT Name FROM Nonexistingtable WHERE population > 50000");

/* このクエリは警告を発生させます */
$result = $mysqli->query("SELECT Name FROM City WHERE population > 50000");
$result->close();

$mysqli->close();
?>
```

参考

- mysqli_debug()
- mysqli_dump_debug_info()

---

mysqli_rollback

mysqli->rollback()

(PHP 5)

mysqli->rollback() — 現在のトランザクションをロールバックする

説明

bool **mysqli_rollback** (mysqli $link)
mysqli
bool **rollback** (void)

データベースの現在のトランザクションをロールバックします。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 自動コミットを無効にします */
$mysqli->autocommit(FALSE);

$mysqli->query("CREATE TABLE myCity LIKE City");
$mysqli->query("ALTER TABLE myCity Type=InnoDB");
$mysqli->query("INSERT INTO myCity SELECT * FROM City LIMIT 50");

/* insert をコミットします */
$mysqli->commit();

/* すべての行を削除します */
$mysqli->query("DELETE FROM myCity");

if ($result = $mysqli->query("SELECT COUNT(*) FROM myCity")) {
    $row = $result->fetch_row();
    printf("%d rows in table myCity.¥n", $row[0]);
    /* 結果を開放します */
    $result->close();
}

/* ロールバックします */
$mysqli->rollback();

if ($result = $mysqli->query("SELECT COUNT(*) FROM myCity")) {
    $row = $result->fetch_row();
    printf("%d rows in table myCity (after rollback).¥n", $row[0]);
    /* 結果を開放します */
    $result->close();
}

/* myCity テーブルを削除します */
$mysqli->query("DROP TABLE myCity");

$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 自動コミットを無効にします */
mysqli_autocommit($link, FALSE);

mysqli_query($link, "CREATE TABLE myCity LIKE City");
mysqli_query($link, "ALTER TABLE myCity Type=InnoDB");
mysqli_query($link, "INSERT INTO myCity SELECT * FROM City LIMIT 50");

/* insert をコミットします */
mysqli_commit($link);

/* すべての行を削除します */
mysqli_query($link, "DELETE FROM myCity");

if ($result = mysqli_query($link, "SELECT COUNT(*) FROM myCity")) {
    $row = mysqli_fetch_row($result);
    printf("%d rows in table myCity.¥n", $row[0]);
    /* 結果を開放します */
    mysqli_free_result($result);
}

/* ロールバックします */
mysqli_rollback($link);

if ($result = mysqli_query($link, "SELECT COUNT(*) FROM myCity")) {
    $row = mysqli_fetch_row($result);
    printf("%d rows in table myCity (after rollback).¥n", $row[0]);
    /* 結果を開放します */
    mysqli_free_result($result);
}

/* myCity テーブルを削除します */
```

```
mysqli_query($link, "DROP TABLE myCity");

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
0 rows in table myCity.
50 rows in table myCity (after rollback).
```

参考

- mysqli_commit()
- mysqli_autocommit()

---

mysqli_rpl_parse_enabled

(PHP 5)

mysqli_rpl_parse_enabled — RPL のパースが有効かどうかを確認する

説明

int **mysqli_rpl_parse_enabled** (mysqli $link)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_rpl_probe

(PHP 5)

mysqli_rpl_probe — RPL の調査

説明

bool **mysqli_rpl_probe** (mysqli $link)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_rpl_query_type

mysqli->rpl_query_type()

(PHP 5)

mysqli->rpl_query_type() — RPL クエリの型を返す

説明

手続き型:

int **mysqli_rpl_query_type** (mysqli $link , string $query)

オブジェクト指向型（メソッド）

mysqli
int **rpl_query_type** (string $query)

クエリの型により、 **MYSQLI_RPL_MASTER**、 **MYSQLI_RPL_SLAVE** あるいは **MYSQLI_RPL_ADMIN** のいずれかを返します。 INSERT・UPDATE およびそれに類する ものは master クエリで、SELECT は slave、そして FLUSH・ REPAIR およびそれに類するものは admin です。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_select_db

mysqli->select_db()

(PHP 5)

mysqli->select_db() — クエリを実行するためのデフォルトのデータベースを選択する

説明

手続き型:

bool **mysqli_select_db** (mysqli $link , string $dbname)

オブジェクト指向型 (メソッド):

mysqli
bool **select_db** (string $dbname)

データベース接続に対してクエリを実行する際に使用する、 デフォルトのデータベースを設定します。

> **注意**: この関数は、接続のデフォルトデータベースを変更する際にのみ使用します。 デフォルトデータベースは、mysqli_connect() の 4 番目の引数でも指定可能です。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

dbname

データベース名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 現在のデフォルトデータベース名を返します */
if ($result = $mysqli->query("SELECT DATABASE()")) {
    $row = $result->fetch_row();
    printf("Default database is %s.¥n", $row[0]);
    $result->close();
}

/* データベースを world に変更します */
$mysqli->select_db("world");

/* 現在のデフォルトデータベース名を返します */
if ($result = $mysqli->query("SELECT DATABASE()")) {
    $row = $result->fetch_row();
    printf("Default database is %s.¥n", $row[0]);
    $result->close();
}

$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 現在のデフォルトデータベース名を返します */
if ($result = mysqli_query($link, "SELECT DATABASE()")) {
    $row = mysqli_fetch_row($result);
    printf("Default database is %s.¥n", $row[0]);
    mysqli_free_result($result);
}

/* データベースを world に変更します */
mysqli_select_db($link, "world");

/* 現在のデフォルトデータベース名を返します */
if ($result = mysqli_query($link, "SELECT DATABASE()")) {
    $row = mysqli_fetch_row($result);
    printf("Default database is %s.¥n", $row[0]);
    mysqli_free_result($result);
}

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Default database is test.
Default database is world.
```

参考

- mysqli_connect()
- mysqli_real_connect()

---

mysqli_send_long_data

(PHP 5)

mysqli_send_long_data — mysqli_stmt_send_long_data() のエイリアス

説明

この関数は mysqli_stmt_send_long_data() のエイリアスです。

注意

注意: **mysqli_send_long_data()** は非推奨で、廃止予定です。

参考

- mysqli_stmt_send_long_data()

---

mysqli_send_query

mysqli->send_query()

(PHP 5)

mysqli->send_query() — クエリを送信する

説明

手続き型:

```
bool mysqli_send_query ( mysqli $link , string $query )
```

オブジェクト指向型（メソッド）

```
mysqli
bool send_query ( string $query )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_server_end

(PHP 5 <= 5.0.5)

mysqli_server_end — 組み込みのサーバをシャットダウンする

説明

```
void mysqli_server_end ( void )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_server_init

(PHP 5 <= 5.0.5)

mysqli_server_init — 組み込みのサーバを初期化する

説明

```
bool mysqli_server_init ([ array $server [, array $groups ]] )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_set_charset

mysqli->set_charset

(PHP 5 >= 5.0.5)

mysqli->set_charset — クライアントのデフォルト文字セットを設定する

説明

手続き型:

bool **mysqli_set_charset** (mysqli $link , string $charset)

オブジェクト指向型 (メソッド):

mysqli
bool **set_charset** (string $charset)

データベースサーバとのデータの送受信に使用する、 デフォルトの文字セットを設定します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

charset

デフォルトとして設定する文字セット。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: Windows プラットフォームでこの関数を使用するには、MySQL クライアント ライブラリのバージョン 4.1.11 以降 (MySQL 5.0 の場合は 5.0.6 以降) が必要です。これは、» http://dev.mysql.com/downloads/connector/php/ でダウンロードできます。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 文字セットを utf8 に変更します */
if (!$mysqli->set_charset("utf8")) {
    printf("Error loading character set utf8: %s¥n", $mysqli->error);
} else {
    printf("Current character set: %s¥n", $mysqli->character_set_name());
}

$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect('localhost', 'my_user', 'my_password', 'test');

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 文字セットを utf8 に変更します */
if (!mysqli_set_charset($link, "utf8")) {
    printf("Error loading character set utf8: %s¥n", mysqli_error($link));
} else {
    printf("Current character set: %s¥n", mysqli_character_set_name($link));
}

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Current character set: utf8
```

参考

- mysqli_character_set_name()
- mysqli_real_escape_string()

---

mysqli_set_local_infile_default

(PHP 5)

mysqli_set_local_infile_default — load local infile コマンド用のユーザ定義ハンドラを削除する

説明

void **mysqli_set_local_infile_default** (mysqli $link)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_set_local_infile_handler

(PHP 5)

mysqli_set_local_infile_handler — LOAD DATA LOCAL INFILE コマンド用のコールバック関数を設定する

説明

bool **mysqli_set_local_infile_handler** (mysqli $link , callback $read_func)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_set_opt

(PHP 5)

mysqli_set_opt — mysqli_options() のエイリアス

説明

この関数は mysqli_options() のエイリアスです。

---

mysqli_slave_query

(PHP 5)

mysqli_slave_query — マスタ/スレーブ設定で、スレーブ側のクエリを実行する

説明

bool **mysqli_slave_query** (mysqli $link , string $query)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_sqlstate

mysqli->sqlstate

(PHP 5)

mysqli->sqlstate — 直前の MySQL の操作での SQLSTATE エラーを返す

説明

手続き型:

string **mysqli_sqlstate** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
string$sqlstate;

直近のエラーについて、SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 つの文字で構成されています。'00000' はエラーが発

生しなかったことを意味します。この値は、ANSI SQL および ODBC で定義されています。とりうる値の一覧は » http://dev.mysql.com/doc/mysql/en/error-handling.html を参照ください。

注意: すべての MySQL エラーが SQLSTATE に対応しているわけではないことに 注意してください。そのようなエラーが発生した場合は、 HY000（一般的なエラー）が返されます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

直前のエラーに関する SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 文字で構成され、'00000' はエラーが発生しなかったことを意味します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* City テーブルはすでに存在します。そのためエラーとなります */
if (!$mysqli->query("CREATE TABLE City (ID INT, Name VARCHAR(30))")) {
    printf("Error - SQLSTATE %s.¥n", $mysqli->sqlstate);
}

$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* City テーブルはすでに存在します。そのためエラーとなります */
if (!mysqli_query($link, "CREATE TABLE City (ID INT, Name VARCHAR(30))")) {
    printf("Error - SQLSTATE %s.¥n", mysqli_sqlstate($link));
}

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Error - SQLSTATE 42S01.
```

参考

- mysqli_errno()
- mysqli_error()

---

mysqli_ssl_set

mysqli->ssl_set()

(PHP 5)

mysqli->ssl_set() — SSL を使用したセキュアな接続を確立する

説明

手続き型:

bool **mysqli_ssl_set** (mysqli $link , string $key , string $cert , string $ca , string $capath , string $cipher)

オブジェクト指向型（メソッド）:

mysqli
bool **ssl_set** (string $key , string $cert , string $ca , string $capath , string $cipher)

SSL を使用してセキュアな接続を確立します。 before mysqli_real_connect() より前にコールする必要があります。この関数は、 OpenSSL サポートが有効になっていない場合は何もしません。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

key

鍵ファイルへのパス。

cert

証明書ファイルへのパス。

ca

CA ファイルへのパス。

capath

信頼された SSL CA 証明書が PEM フォーマットで格納されているディレクトリへのパス。

cipher

SSL の暗号化に使用可能な暗号形式の一覧。

未使用の SSL パラメータには **NULL** を渡します。

返り値

この関数は、常に **TRUE** を返します。もし SSL が正しく設定できていない場合、mysqli_real_connect() は接続時にエラーを返します。

参考

- mysqli_options()
- mysqli_real_connect()

---

mysqli_stat

mysqli->stat()

(PHP 5)

mysqli->stat() — 現在のシステム状態を取得する

説明

手続き型:

```
string mysqli_stat ( mysqli $link )
```

オブジェクト指向型（メソッド）:

```
mysqli
string stat ( void )
```

mysqli_stat() は、'mysqladmin status' コマンドが 返すのと同じ情報を返します。この中には、起動からの秒数・起動中の スレッドの数・リロード数および開かれているテーブルなどが含まれます。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

サーバの状態を示す文字列を返します。エラー時には **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

printf ("System status: %s¥n", $mysqli->stat());

$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

printf("System status: %s¥n", mysqli_stat($link));

mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
System status: Uptime: 272  Threads: 1  Questions: 5340  Slow queries: 0
Opens: 13  Flush tables: 1  Open tables: 0  Queries per second avg: 19.632
Memory in use: 8496K  Max memory used: 8560K
```

参考

- mysqli_get_server_info()

---

mysqli_stmt_affected_rows

mysqli_stmt->affected_rows

(PHP 5)

mysqli_stmt->affected_rows — 直近に実行されたステートメントで変更・削除・あるいは追加された行の総数を返す

説明

手続き型 :

int **mysqli_stmt_affected_rows** (mysqli_stmt $stmt)

オブジェクト指向型（プロパティ）:

mysqli_stmt
int$affected_rows;

INSERT、 UPDATE あるいは DELETE クエリによって変更された行の数を返します。

この関数は、テーブルを更新するクエリに対してのみ働きます。 SELECT クエリが返す行の数を知るには、 mysqli_stmt_num_rows() 関数を代わりに使用します。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

ゼロより大きい整数の場合、変更した行の数を示します。ゼロの場合は、 UPDATE/DELETE で 1 行も更新されなかった・WHERE 句にマッチする行がなかった・ あるいはまだクエリが実行されていないのいずれかであることを示します。 -1 は、クエリがエラーを返したことを示します。

> **注意**: 変更された行の数が PHP の int 型の最大値をこえる場合は、変更された 行の数は文字列として返されます。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 一時テーブルを作成します */
$mysqli->query("CREATE TEMPORARY TABLE myCountry LIKE Country");

$query = "INSERT INTO myCountry SELECT * FROM Country WHERE Code LIKE ?";

/* ステートメントを準備します */
if ($stmt = $mysqli->prepare($query)) {

    /* プレースホルダに変数をバインドします */
    $code = 'A%';
    $stmt->bind_param("s", $code);

    /* ステートメントを実行します */
    $stmt->execute();
```

```
    printf("rows inserted: %d¥n", $stmt->affected_rows);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* 一時テーブルを作成します */
mysqli_query($link, "CREATE TEMPORARY TABLE myCountry LIKE Country");

$query = "INSERT INTO myCountry SELECT * FROM Country WHERE Code LIKE ?";

/* ステートメントを準備します */
if ($stmt = mysqli_prepare($link, $query)) {

    /* プレースホルダに変数をバインドします */
    $code = 'A%';
    mysqli_stmt_bind_param($stmt, "s", $code);

    /* ステートメントを実行します */
    mysqli_stmt_execute($stmt);

    printf("rows inserted: %d¥n", mysqli_stmt_affected_rows($stmt));

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
rows inserted: 17
```

参考

- mysqli_stmt_num_rows()
- mysqli_prepare()

---

mysqli_stmt_attr_get

(PHP 5)

mysqli_stmt_attr_get —

説明

```
int mysqli_stmt_attr_get ( mysqli_stmt $stmt , int $attr )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_stmt_attr_set

(PHP 5)

mysqli_stmt_attr_set —

説明

```
bool mysqli_stmt_attr_set ( mysqli_stmt $stmt , int $attr , int $mode )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_stmt_bind_param

stmt->bind_param()

(No version information available, might be only in CVS)

stmt->bind_param() — プリペアドステートメントのパラメータに変数をバインドする

説明

手続き型:

bool **mysqli_stmt_bind_param** (mysqli_stmt $stmt , string $types , mixed &$var1 [, mixed &$...])

オブジェクト指向型（メソッド）:

mysqli_stmt
bool **bind_param** (string $types , mixed &$var1 [, mixed &$...])

変数を、SQL ステートメントのパラメータマーカにバインドします。 この変数は、mysqli_prepare() に渡されたものです。

> **注意**: データのサイズがパケットサイズの最大値（max_allowed_packet）を こえた場合、types に b を 指定して mysqli_stmt_send_long_data() を使用し、 データをパケットに分割して送信する必要があります。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

types

ひとつあるいは複数の文字で、対応するバインド変数の型を表します。

型指定文字

文字	説明
i	対応する変数の型は integer です。
d	対応する変数の型は double です。
s	対応する変数の型は string です。
b	対応する変数の型は blob で、複数のパケットに分割して送信されます。

var1

変数の数。文字列 types の長さは、ステートメント中のパラメータの数と一致する必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli('localhost', 'my_user', 'my_password', 'world');

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$stmt = $mysqli->prepare("INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");
$stmt->bind_param('sssd', $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;

/* プリペアドステートメントを実行します */
$stmt->execute();

printf("%d Row inserted.¥n", $stmt->affected_rows);

/* ステートメントと接続を閉じます */
$stmt->close();

/* CountryLanguage テーブルをクリアします */
$mysqli->query("DELETE FROM CountryLanguage WHERE Language='Bavarian'");
printf("%d Row deleted.¥n", $mysqli->affected_rows);

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect('localhost', 'my_user', 'my_password', 'world');

/* 接続状況をチェックします */
if (!$link) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}
```

```
$stmt = mysqli_prepare($link, "INSERT INTO CountryLanguage VALUES (?, ?, ?, ?)");
mysqli_stmt_bind_param($stmt, 'sssd', $code, $language, $official, $percent);

$code = 'DEU';
$language = 'Bavarian';
$official = "F";
$percent = 11.2;

/* プリペアドステートメントを実行します */
mysqli_stmt_execute($stmt);

printf("%d Row inserted.¥n", mysqli_stmt_affected_rows($stmt));

/* ステートメントと接続を閉じます */
mysqli_stmt_close($stmt);

/* CountryLanguage テーブルをクリアします */
mysqli_query($link, "DELETE FROM CountryLanguage WHERE Language='Bavarian'");
printf("%d Row deleted.¥n", mysqli_affected_rows($link));

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
1 Row inserted.
1 Row deleted.
```

参考

- mysqli_stmt_bind_result()
- mysqli_stmt_execute()
- mysqli_stmt_fetch()
- mysqli_prepare()
- mysqli_stmt_send_long_data()
- mysqli_stmt_errno()
- mysqli_stmt_error()

---

mysqli_stmt_bind_result

stmt->bind_result()

(No version information available, might be only in CVS)

stmt->bind_result() — 結果を保存するため、プリペアドステートメントに変数をバインドする

説明

手続き型:

bool **mysqli_stmt_bind_result** (mysqli_stmt $stmt , mixed &$var1 [, mixed &$...])

オブジェクト指向型（メソッド）:

mysqli_stmt
bool **bind_result** (mixed &$var1 [, mixed &$...])

結果セットのカラムを変数にバインドします。

データを取得するために mysqli_stmt_fetch() がコールされた場合、MySQL クライアント/ サーバ プロトコルはバインドされたカラムのデータを var1, ... に格納します。

> **注意**: すべてのカラムを、mysqli_stmt_execute() をコールしてから mysqli_stmt_fetch() をコールするまでの間に バインドしておく必要があることに注意しましょう。カラムの型に 応じて、バインド変数の型も対応する PHP の型に自動的に変換されます。カラムのバインドや再バインドはいつでも可能で、たとえ結果セットを途中まで 取得した後であっても可能です。新しくバインドした内容が効力を発揮するのは、 次に mysqli_stmt_fetch() がコールされたときからです。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

var1

バインドする変数。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* ステートメントを準備します */
if ($stmt = $mysqli->prepare("SELECT Code, Name FROM Country ORDER BY Name LIMIT 5")) {
    $stmt->execute();

    /* プリペアドステートメントに変数をバインドします */
    $stmt->bind_result($col1, $col2);

    /* 値を取得します */
    while ($stmt->fetch()) {
        printf("%s %s¥n", $col1, $col2);
    }

    /* ステートメントを閉じます */
    $stmt->close();
}
/* 接続を閉じます */
$mysqli->close();

?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (!$link) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* ステートメントを準備します */
if ($stmt = mysqli_prepare($link, "SELECT Code, Name FROM Country ORDER BY Name LIMIT 5")) {
    mysqli_stmt_execute($stmt);

    /* プリペアドステートメントに変数をバインドします */
    mysqli_stmt_bind_result($stmt, $col1, $col2);

    /* 値を取得します */
    while (mysqli_stmt_fetch($stmt)) {
        printf("%s %s¥n", $col1, $col2);
    }

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
AFG  Afghanistan
ALB  Albania
DZA  Algeria
ASM  American Samoa
AND  Andorra
```

参考

- mysqli_stmt_bind_param()
- mysqli_stmt_execute()
- mysqli_stmt_fetch()
- mysqli_prepare()
- mysqli_stmt_prepare()
- mysqli_stmt_init()
- mysqli_stmt_errno()
- mysqli_stmt_error()

---

mysqli_stmt_close

mysqli_stmt->close()

(PHP 5)

mysqli_stmt->close() — プリペアドステートメントを閉じる

説明

手続き型:

```
bool mysqli_stmt_close ( mysqli_stmt $stmt )
```

オブジェクト指向型（メソッド）:

```
mysqli_stmt
bool close ( void )
```

プリペアドステートメントを閉じます。また、**mysqli_stmt_close()** は stmt が指すステートメントハンドルを開放します。 現在のステートメントが実行中あるいはまだ結果を取得していない場合、 この関数はキャンセルされ、次のクエリが実行されます。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mysqli_prepare()

---

mysqli_stmt_data_seek

stmt->data_seek()

(No version information available, might be only in CVS)

stmt->data_seek() — ステートメントの結果セットの任意の行に移動する

説明

手続き型:

```
void mysqli_stmt_data_seek ( mysqli_stmt $stmt , int $offset )
```

オブジェクト指向型（メソッド）:

```
mysqli_stmt
void data_seek ( int $offset )
```

ステートメントの結果セット内で、 任意の位置に結果ポインタを移動します。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

offset

ゼロから行の総数 - 1 (0..mysqli_stmt_num_rows() - 1) までの間である必要があります。

返り値

値を返しません。

例

Example#1 オブジェクト指向型

```php
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* クエリを実行します */
    $stmt->execute();

    /* 結果変数をバインドします */
    $stmt->bind_result($name, $code);

    /* 結果を取得します */
    $stmt->store_result();
```

```
    /* 行番号 400 に移動します */
    $stmt->data_seek(399);

    /* 値を取得します */
    $stmt->fetch();

    printf ("City: %s  Countrycode: %s¥n", $name, $code);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    /* 結果変数をバインドします */
    mysqli_stmt_bind_result($stmt, $name, $code);

    /* 結果を取得します */
    mysqli_stmt_store_result($stmt);

    /* 行番号 400 に移動します */
    mysqli_stmt_data_seek($stmt, 399);

    /* 値を取得します */
    mysqli_stmt_fetch($stmt);

    printf ("City: %s  Countrycode: %s¥n", $name, $code);

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
City: Benin City  Countrycode: NGA
```

参考

- *mysqli_prepare()*

---

mysqli_stmt_errno

mysqli_stmt->errno

(PHP 5)

mysqli_stmt->errno — 直近のステートメントのコールに関するエラーコードを返す

説明

手続き型 :

```
int mysqli_stmt_errno ( mysqli_stmt $stmt )
```

オブジェクト指向型（プロパティ）:

```
mysqli_stmt
int$errno;
```

直近に実行されたステートメントが 成功あるいは失敗した際のエラーコードを返します。

クライアントのエラーメッセージ番号は MySQL の errmsg.h ヘッダファイルで、そしてサーバのエラーメッセージ番号は mysqld_error.h で定義されています。MySQL のソース配布の中には、エラーメッセージの 完全なリストが Docs/mysqld_error.txt に含まれています。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

エラーコードの値を返します。ゼロはエラーが発生しなかったことを示します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* テーブルを削除します */
    $mysqli->query("DROP TABLE myCountry");

    /* クエリを実行します */
    $stmt->execute();

    printf("Error: %d.¥n", $stmt->errno);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* テーブルを削除します */
    mysqli_query($link, "DROP TABLE myCountry");

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    printf("Error: %d.¥n", mysqli_stmt_errno($stmt));

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Error: 1146.
```

参考

- mysqli_stmt_error()
- mysqli_stmt_sqlstate()

---

mysqli_stmt_error

mysqli_stmt->error

(PHP 5)

mysqli_stmt->error — 直近のステートメントのエラー内容を文字列で返す

説明

手続き型:

string **mysqli_stmt_error** (mysqli_stmt $stmt)

オブジェクト指向型（プロパティ）:

mysqli_stmt
string$error;

直近に実行されたステートメントが 成功あるいは失敗した際のエラーメッセージを返します。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

エラーの内容を文字列で返します。エラーが発生しなかった場合は空文字列を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* テーブルを削除します */
    $mysqli->query("DROP TABLE myCountry");

    /* クエリを実行します */
    $stmt->execute();

    printf("Error: %s.¥n", $stmt->error);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* テーブルを削除します */
    mysqli_query($link, "DROP TABLE myCountry");

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    printf("Error: %s.¥n", mysqli_stmt_error($stmt));

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}
```

```
/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Error: Table 'world.myCountry' doesn't exist.
```

参考

- mysqli_stmt_errno()
- mysqli_stmt_sqlstate()

---

mysqli_stmt_execute

stmt->execute()

(No version information available, might be only in CVS)

stmt->execute() — プリペアドクエリを実行する

説明

手続き型:

```
bool mysqli_stmt_execute ( mysqli_stmt $stmt )
```

オブジェクト指向型（メソッド）:

```
mysqli_stmt
bool execute ( void )
```

事前に mysqli_prepare() 関数で用意されたクエリを実行します。 パラメータマーカが存在する場合、その内容は自動的に適切なデータに置き換えられます。

ステートメントが UPDATE、DELETE あるいは INSERT であった場合、 変更された行の総数は mysqli_stmt_affected_rows() 関数を使用することで取得可能です。同様に、クエリが結果セットを返す場合は mysqli_stmt_fetch() 関数を使用できます。

> **注意**: **mysqli_stmt_execute()** を使用した際には、 他のクエリを実行する前に mysqli_stmt_fetch() 関数を使用する必要があります。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

/* insert ステートメントを準備します */
$query = "INSERT INTO myCity (Name, CountryCode, District) VALUES (?,?,?)";
$stmt = $mysqli->prepare($query);

$stmt->bind_param("sss", $val1, $val2, $val3);

$val1 = 'Stuttgart';
$val2 = 'DEU';
$val3 = 'Baden-Wuerttemberg';

/* ステートメントを実行します */
$stmt->execute();

$val1 = 'Bordeaux';
$val2 = 'FRA';
$val3 = 'Aquitaine';

/* ステートメントを実行します */
$stmt->execute();
```

```
/* ステートメントを閉じます */
$stmt->close();

/* myCity からすべての行を取得します */
$query = "SELECT Name, CountryCode, District FROM myCity";
if ($result = $mysqli->query($query)) {
    while ($row = $result->fetch_row()) {
        printf("%s (%s,%s)¥n", $row[0], $row[1], $row[2]);
    }
    /* 結果セットを開放します */
    $result->close();
}

/* テーブルを削除します */
$mysqli->query("DROP TABLE myCity");

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

/* insert ステートメントを準備します */
$query = "INSERT INTO myCity (Name, CountryCode, District) VALUES (?,?,?)";
$stmt = mysqli_prepare($link, $query);

mysqli_stmt_bind_param($stmt, "sss", $val1, $val2, $val3);

$val1 = 'Stuttgart';
$val2 = 'DEU';
$val3 = 'Baden-Wuerttemberg';

/* ステートメントを実行します */
mysqli_stmt_execute($stmt);

$val1 = 'Bordeaux';
$val2 = 'FRA';
$val3 = 'Aquitaine';

/* ステートメントを実行します */
mysqli_stmt_execute($stmt);

/* ステートメントを閉じます */
mysqli_stmt_close($stmt);

/* myCity からすべての行を取得します */
$query = "SELECT Name, CountryCode, District FROM myCity";
if ($result = mysqli_query($link, $query)) {
    while ($row = mysqli_fetch_row($result)) {
        printf("%s (%s,%s)¥n", $row[0], $row[1], $row[2]);
    }
    /* 結果セットを開放します */
    mysqli_free_result($result);
}

/* テーブルを削除します */
mysqli_query($link, "DROP TABLE myCity");

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Stuttgart (DEU,Baden-Wuerttemberg)
Bordeaux (FRA,Aquitaine)
```

参考

- mysqli_prepare()
- mysqli_stmt_bind_param()

---

mysqli_stmt_fetch

stmt->fetch()

(No version information available, might be only in CVS)

stmt->fetch() — プリペアドステートメントから結果を取得し、バインド変数に格納する

説明

手続き型:

bool **mysqli_stmt_fetch** (mysqli_stmt $stmt)

オブジェクト指向型（メソッド）:

mysqli_stmt
bool **fetch** (void)

プリペアドステートメントから結果を読み込み、 mysqli_stmt_bind_result() でバインドした変数に格納します。

注意: **mysqli_stmt_fetch()** をコールする前に、すべての カラムがバインド済みである必要があることに注意しましょう。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

返り値

値	説明
TRUE	成功。データが取得されました。
FALSE	エラーが発生しました。
NULL	行/データがもうありません。あるいは、データの切り詰めが発生しました。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 150,5";

if ($stmt = $mysqli->prepare($query)) {

    /* ステートメントを実行します */
    $stmt->execute();

    /* 結果変数をバインドします */
    $stmt->bind_result($name, $code);

    /* 値を取得します */
    while ($stmt->fetch()) {
        printf ("%s (%s)¥n", $name, $code);
    }

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER by ID DESC LIMIT 150,5";

if ($stmt = mysqli_prepare($link, $query)) {

    /* ステートメントを実行します */
    mysqli_stmt_execute($stmt);

    /* 結果変数をバインドします */
    mysqli_stmt_bind_result($stmt, $name, $code);

    /* 値を取得します */
    while (mysqli_stmt_fetch($stmt)) {
        printf ("%s (%s)¥n", $name, $code);
    }

    /* ステートメントを閉じます */
```

```
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Rockford  (USA)
Tallahassee  (USA)
Salinas  (USA)
Santa  Clarita  (USA)
Springfield  (USA)
```

参考

- mysqli_prepare()
- mysqli_stmt_errno()
- mysqli_stmt_error()
- mysqli_stmt_bind_result()

---

mysqli_stmt_field_count

(PHP 5)

mysqli_stmt_field_count — 指定したステートメントのフィールド数を返す

説明

int **mysqli_stmt_field_count** (mysqli_stmt $stmt)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_stmt_free_result

stmt->free_result()

(No version information available, might be only in CVS)

stmt->free_result() — 指定したステートメントハンドルの結果を格納したメモリを開放する

説明

手続き型:

void **mysqli_stmt_free_result** (mysqli_stmt $stmt)

オブジェクト指向型（メソッド）:

mysqli_stmt
void **free_result** (void)

ステートメントに関連する結果のメモリを開放します。このメモリは mysqli_stmt_store_result() によって割り当てられたものです。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

値を返しません。

参考

- mysqli_stmt_store_result()

---

mysqli_stmt_get_warnings

(PHP 5 >= 5.1.0)

mysqli_stmt_get_warnings —

説明

object ***mysqli_stmt_get_warnings*** *(mysqli_stmt $stmt)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_stmt_init

mysqli->stmt_init()

(PHP 5)

mysqli->stmt_init() — ステートメントを初期化し、mysqli_stmt_prepare で使用するオブジェクトを返す

説明

手続き型:

mysqli_stmt ***mysqli_stmt_init*** *(mysqli $link)*

オブジェクト指向型（プロパティ）:

mysqli
mysqli_stmt ***stmt_init*** *(void)*

mysqli_stmt_prepare() で使用可能な ステートメントオブジェクトを割り当て、初期化します。

***注意**: mysqli_stmt_prepare() がコールされるまで、 これ以降のあらゆる mysqli_stmt 関数のコールは失敗します。*

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

オブジェクトを返します。

参考

- *mysqli_stmt_prepare()*

---

mysqli_stmt_insert_id

(PHP 5)

mysqli_stmt_insert_id — 直近の INSERT 操作で生成した ID を取得する

説明

mixed ***mysqli_stmt_insert_id*** *(mysqli_stmt $stmt)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

mysqli_stmt_num_rows

stmt->num_rows

(No version information available, might be only in CVS)

stmt->num_rows — ステートメントの結果セットの行数を返す

説明

手続き型 :

int ***mysqli_stmt_num_rows*** *(mysqli_stmt $stmt)*

オブジェクト指向型（プロパティ）:

mysqli_stmt
int$num_rows;

*結果セットの行の数を返します。**mysqli_stmt_num_rows()** が使用できるかどうかは、mysqli_stmt_store_result() を用いて結果をステートメントハンドルにバッファリングしているかどうかに 依存します。*

*mysqli_stmt_store_result() を使用した場合は、 すぐに **mysqli_stmt_num_rows()** をコールできます。*

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

結果セットの行数を表す整数値を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = $mysqli->prepare($query)) {

    /* クエリを実行します */
    $stmt->execute();

    /* 結果を格納します */
    $stmt->store_result();

    printf("Number of rows: %d.¥n", $stmt->num_rows);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = mysqli_prepare($link, $query)) {

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    /* 結果を格納します */
    mysqli_stmt_store_result($stmt);

    printf("Number of rows: %d.¥n", mysqli_stmt_num_rows($stmt));

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Number of rows: 20.
```

参考

- mysqli_stmt_affected_rows()
- mysqli_prepare()
- mysqli_stmt_store_result()

---

mysqli_stmt_param_count

stmt->param_count

(No version information available, might be only in CVS)

stmt->param_count — 指定したステートメントのパラメータ数を返す

説明

手続き型:

int **mysqli_stmt_param_count** (mysqli_stmt $stmt)

オブジェクト指向型（プロパティ）:

mysqli_stmt
int$param_count;

プリペアドステートメント内のパラメータマーカの数を返します。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

パラメータの数を整数で返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if ($stmt = $mysqli->prepare("SELECT Name FROM Country WHERE Name=? OR Code=?")) {

    $marker = $stmt->param_count;
    printf("Statement has %d markers.¥n", $marker);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

if ($stmt = mysqli_prepare($link, "SELECT Name FROM Country WHERE Name=? OR Code=?")) {

    $marker = mysqli_stmt_param_count($stmt);
    printf("Statement has %d markers.¥n", $marker);

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Statement has 2 markers.
```

参考

- mysqli_prepare()

---

mysqli_stmt_prepare

stmt->prepare()

(No version information available, might be only in CVS)

stmt->prepare() — SQL ステートメントを実行するために準備する

説明

手続き型:

bool **mysqli_stmt_prepare** (mysqli_stmt $stmt , string $query)

オブジェクト指向型（メソッド）

mysqli_stmt
mixed **prepare** (string $query)

null で終わる文字列で指定した SQL クエリを準備します。

パラメータマーカは、ステートメントの実行や行の取得の前に mysqli_stmt_bind_param() や mysqli_stmt_bind_result() を使用して アプリケーション変数にバインドする必要があります。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

query

クエリを表す文字列。単一の SQL 文で構成されている必要があります。

ひとつまたは複数のパラメータを SQL 文に含めることができます。 そのためには、適切な位置にクエスチョンマーク (?) を埋め込みます。

> **注意**: ステートメントの最後にセミコロンや ¥g を追加してはいけません。
>
> **注意**: パラメータのマーカは、それが SQL 文の適切な位置にある場合のみ 有効です。例えば INSERT 文の VALUES() リストの中 (行に登録するカラム値を指定する) や WHERE 句で列のデータと比較する値などが適切な位置の例です。
> しかし、識別子 (テーブルやカラムの名前) や SELECT 文で選択する 項目の名前に指定したり、(等号 = のような) 二項演算子の両側にパラメータを指定したりすることはできません。 後者の制限は、パラメータの型が判断できなくなることによるものです。 また、パラメータのマーカを NULL と比較して ? IS NULL のようにすることもできません。 一般に、パラメータが使用可能なのはデータ操作言語 (DML) ステートメントであり、データ定義言語 (DDL) ステートメントでは 使用できません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```php
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";

/* プリペアドステートメントを作成します */
$stmt = $mysqli->stmt_init();
if ($stmt->prepare("SELECT District FROM City WHERE Name=?")) {

    /* マーカにパラメータをバインドします */
    $stmt->bind_param("s", $city);

    /* クエリを実行します */
    $stmt->execute();

    /* 結果変数をバインドします */
    $stmt->bind_result($district);

    /* 値を取得します */
    $stmt->fetch();

    printf("%s is in district %s¥n", $city, $district);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$city = "Amersfoort";
```

```
/* プリペアドステートメントを作成します */
$stmt = mysqli_stmt_init($link);
if (mysqli_stmt_prepare($stmt, 'SELECT District FROM City WHERE Name=?')) {

    /* マーカにパラメータをバインドします */
    mysqli_stmt_bind_param($stmt, "s", $city);

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    /* 結果変数をバインドします */
    mysqli_stmt_bind_result($stmt, $district);

    /* 値を取得します */
    mysqli_stmt_fetch($stmt);

    printf("%s is in district %s¥n", $city, $district);

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Amersfoort is in district Utrecht
```

参考

mysqli_stmt_init(), mysqli_stmt_execute(), mysqli_stmt_fetch(), mysqli_stmt_bind_param(), mysqli_stmt_bind_result()
mysqli_stmt_close().

---

mysqli_stmt_reset

stmt->reset()

(No version information available, might be only in CVS)

stmt->reset() — プリペアドステートメントをリセットする

説明

手続き型:

bool **mysqli_stmt_reset** (mysqli_stmt $stmt)

オブジェクト指向型（メソッド）:

mysqli_stmt
bool **reset** (void)

クライアントおよびサーバで、実行後のプリペアドステートメントをリセットします。

現在のところ、これは主に mysqli_stmt_send_long_data() で送信したデータをリセットするために使用されます。

別のクエリを元にプリペアドステートメントを用意する場合は、mysqli_stmt_prepare() 関数を使用します。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- mysqli_prepare()

---

mysqli_stmt_result_metadata

stmt->result_metadata()

(No version information available, might be only in CVS)

stmt->result_metadata() — プリペアドステートメントから結果セットのメタデータを返す

説明

手続き型:

mysqli_result **mysqli_stmt_result_metadata** (mysqli_stmt $stmt)

オブジェクト指向型（メソッド）:

mysqli_stmt
mysqli_result **result_metadata** (void)

mysqli_prepare() に渡したステートメントが 結果セットを返すものであった場合、**mysqli_stmt_result_metadata()** はオブジェクトを返します。このオブジェクトは、結果のフィールド数や 各フィールドの情報などのメタ情報を取得するために使用可能です。

> **注意**: メタデータを取得するには、この結果セットポインタを、以下のいずれかの（フィールドベースの）関数に渡します。
>
> - mysqli_num_fields()
> - mysqli_fetch_field()
> - mysqli_fetch_field_direct()
> - mysqli_fetch_fields()
> - mysqli_field_count()
> - mysqli_field_seek()
> - mysqli_field_tell()
> - mysqli_free_result()

結果セットは、使用終了後に開放すべきです。そのためには、結果セットを mysqli_free_result() に渡します。

> **注意**: **mysqli_stmt_result_metadata()** が返す結果セットには メタデータのみが含まれています。実際の行データは含まれません。行データを 取得するには、ステートメントハンドルを mysqli_stmt_fetch() に渡してください。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

結果のオブジェクトを返します。エラー時には **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "test");

$mysqli->query("DROP TABLE IF EXISTS friends");
$mysqli->query("CREATE TABLE friends (id int, name varchar(20))");

$mysqli->query("INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

$stmt = $mysqli->prepare("SELECT id, name FROM friends");
$stmt->execute();

/* メタデータの結果セットを取得します */
$result = $stmt->result_metadata();

/* メタデータ結果セットからフィールド情報を取得します */
$field = $result->fetch_field();

printf("Fieldname: %s¥n", $field->name);

/* 結果セットを閉じます */
$result->close();

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "test");

mysqli_query($link, "DROP TABLE IF EXISTS friends");
mysqli_query($link, "CREATE TABLE friends (id int, name varchar(20))");

mysqli_query($link, "INSERT INTO friends VALUES (1,'Hartmut'), (2, 'Ulf')");

$stmt = mysqli_prepare($link, "SELECT id, name FROM friends");
mysqli_stmt_execute($stmt);

/* メタデータの結果セットを取得します */
$result = mysqli_stmt_result_metadata($stmt);

/* メタデータ結果セットからフィールド情報を取得します */
$field = mysqli_fetch_field($result);
```

```
printf("Fieldname:  %s¥n", $field->name);

/* 結果セットを閉じます */
mysqli_free_result($result);

/* 接続を閉じます */
mysqli_close($link);
?>
```

参考

- mysqli_prepare()
- mysqli_free_result()

---

mysqli_stmt_send_long_data

stmt->send_long_data()

(No version information available, might be only in CVS)

stmt->send_long_data() — データをブロックで送信する

説明

手続き型:

bool **mysqli_stmt_send_long_data** (mysqli_stmt $stmt , int $param_nr , string $data)

オブジェクト指向型（メソッド）

mysqli_stmt
bool **send_long_data** (int $param_nr , string $data)

パラメータのデータを、サーバに分割して送信します。例えば blob のサイズが max_allowed_packet を越えてしまう場合などに使用します。この関数は、カラムに文字やバイナリのデータを送信するために複数回 コールすることが可能です。そのカラムの型は TEXT あるいは BLOB である 必要があります。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

param_nr

データに関連付けるパラメータを示します。 パラメータの番号は 0 から始まります。

data

送信するデータを含む文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向形式

```
<?php
$stmt = $mysqli->prepare("INSERT INTO messages (message) VALUES (?)");
$null = NULL;
$stmt->bind_param("b", $null);
$fp = fopen("messages.txt", "r");
while (!feof($fp)) {
    $stmt->send_long_data(0, fread($fp, 8192));
}
fclose($fp);
$stmt->execute();
?>
```

参考

- mysqli_prepare()
- mysqli_stmt_bind_param()

---

mysqli_stmt_sqlstate

mysqli_stmt->sqlstate()

(PHP 5)

mysqli_stmt->sqlstate() — 直前のステートメントの操作での SQLSTATE エラーを返す

説明

string **mysqli_stmt_sqlstate** (mysqli_stmt $stmt)

オブジェクト指向型（プロパティ）:

mysqli_stmt
string$sqlstate;

直近に実行されたプリペアドステートメントのエラーについて、 SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 つの文字で構成されています。'00000' はエラーが発生しなかったことを意味します。この値は、ANSI SQL および ODBC で定義されています。とりうる値の一覧は » http://dev.mysql.com/doc/mysql/en/error-handling.html を参照ください。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

直前のエラーに関する SQLSTATE エラーコードを含む文字列を返します。 エラーコードは 5 文字で構成され、'00000' はエラーが発生しなかったことを意味します。

注意

> **注意**: すべての MySQL エラーが SQLSTATE に対応しているわけではないことに 注意してください。そのようなエラーが発生した場合は、 HY000（一般的なエラー）が返されます。

例

Example#1 オブジェクト指向型

```php
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCountry LIKE Country");
$mysqli->query("INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = $mysqli->prepare($query)) {

    /* テーブルを削除します */
    $mysqli->query("DROP TABLE myCountry");

    /* クエリを実行します */
    $stmt->execute();

    printf("Error: %s.¥n", $stmt->sqlstate);

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```php
<?php
/* 接続をオープンします */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCountry LIKE Country");
mysqli_query($link, "INSERT INTO myCountry SELECT * FROM Country");


$query = "SELECT Name, Code FROM myCountry ORDER BY Name";
if ($stmt = mysqli_prepare($link, $query)) {

    /* テーブルを削除します */
    mysqli_query($link, "DROP TABLE myCountry");

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    printf("Error: %s.¥n", mysqli_stmt_sqlstate($stmt));

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
```

```
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Error:  42S02.
```

参考

- mysqli_stmt_errno()
- mysqli_stmt_error()

---

mysqli_stmt_store_result

mysqli_stmt->store_result()

(PHP 5)

mysqli_stmt->store_result() — プリペアドステートメントから結果を転送する

説明

手続き型:

```
bool mysqli_stmt_store_result ( mysqli_stmt $stmt )
```

オブジェクト指向型（メソッド）:

```
mysqli_stmt
bool store_result ( void )
```

クエリが結果セットを返す場合（SELECT, SHOW, DESCRIBE, EXPLAIN）、常に **mysqli_stmt_store_result()** をコールする必要があります。また、この関数は結果セットをクライアントの バッファに格納するだけであり、データを取得するには続けて mysqli_stmt_fetch() をコールします。

> **注意**: その他のクエリでは **mysqli_stmt_store_result()** をコールする必要はありません。しかし、もしコールしてしまったとしてもなんらかの悪影響を及ぼすことは一切ありません。クエリが結果セットを 返すかどうかは、mysqli_stmt_result_metadata() が NULL を返すかどうかで調べられます。

パラメータ

stmt

手続き型のみ: mysqli_stmt_init() が返すステートメント ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 オブジェクト指向型

```
<?php
/* 接続をオープンします */
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
     exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = $mysqli->prepare($query)) {

    /* クエリを実行します */
    $stmt->execute();

    /* 結果を保存します */
    $stmt->store_result();

    printf("Number of rows: %d.¥n", $stmt->num_rows);

    /* 結果を開放します */
    $stmt->free_result();

    /* ステートメントを閉じます */
    $stmt->close();
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
/* 接続をオープンします */
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query = "SELECT Name, CountryCode FROM City ORDER BY Name LIMIT 20";
if ($stmt = mysqli_prepare($link, $query)) {

    /* クエリを実行します */
    mysqli_stmt_execute($stmt);

    /* 結果を格納します */
    mysqli_stmt_store_result($stmt);

    printf("Number of rows: %d.¥n", mysqli_stmt_num_rows($stmt));

    /* 結果を開放します */
    mysqli_stmt_free_result($stmt);

    /* ステートメントを閉じます */
    mysqli_stmt_close($stmt);
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Number of rows: 20.
```

参考

- mysqli_prepare()
- mysqli_stmt_result_metadata()
- mysqli_stmt_fetch()

---

mysqli_store_result

mysqli->store_result()

(PHP 5)

mysqli->store_result() — 直近のクエリから結果セットを転送する

説明

手続き型:

```
mysqli_result mysqli_store_result ( mysqli $link )
```

オブジェクト指向型（メソッド）:

```
mysqli
mysqli_result store_result ( void )
```

mysqli_data_seek() で使用される、 link で表されたデータベース接続の直近のクエリ から結果セットを転送します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

バッファに格納した結果オブジェクトを返します。エラー時には FALSE を返します。

> **注意**: mysqli_store_result() は、クエリが結果セットを 返さなかった場合（例えば、クエリが INSERT 文であった場合）に FALSE を返します。また、結果セットの読み込みに失敗した場合にも FALSE を返します。エラーが発生したかどうかを調べるには、mysqli_error() が空文字列以外を返す・ mysqli_errno() がゼロ以外の値を返す・あるいは mysqli_field_count() がゼロ以外の値を返す のいずれかを確認します。それ以外にこの関数が FALSE を返す理由としては mysqli_query() のコールに成功して返された結果セットが大きすぎる（メモリに割り当てられない）場合がありえます。 もし mysqli_field_count() がゼロ以外の値を 返した場合、文は空でない結果セットを生成しています。

注意

> **注意**: クエリ結果が使用するメモリを mysqli_free_result() 関数で開放するのは、どんな場合でも大切です。しかし、大きい結果セットを mysqli_store_result() で転送した際は、特にこれが 重要となります。

例

See mysqli_multi_query().

参考

- mysqli_real_query()
- mysqli_use_result()

---

mysqli_thread_id

mysqli->thread_id

(PHP 5)

mysqli->thread_id — 現在の接続のスレッド ID を返す

説明

手続き型:

int **mysqli_thread_id** (mysqli $link)

オブジェクト指向型（プロパティ）:

mysqli
int$thread_id;

mysqli_thread_id() 関数は、現在の接続の スレッド ID を返します。この ID を使用すると、mysqli_kill() 関数でセッションを切断することが可能です。接続が切断し mysqli_ping() で再接続した場合は、スレッド ID は別のものになります。そのため、必要になったそのときにスレッド ID を取得するべきです。

> **注意**: スレッド ID は接続単位で割り当てられます。そのため、もし いったん切断した接続が再度確立された場合、新しいスレッド ID が割り当てられます。
> 実行中のクエリを停止するには、SQL コマンド KILL QUERY processid を使用します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

現在の接続のスレッド ID を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* スレッド ID を取得します */
$thread_id = $mysqli->thread_id;

/* 接続を切断します */
$mysqli->kill($thread_id);

/* これはエラーとなります */
if (!$mysqli->query("CREATE TABLE myCity LIKE City")) {
    printf("Error: %s¥n", $mysqli->error);
    exit;
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

/* スレッド ID を取得します */
$thread_id = mysqli_thread_id($link);

/* 接続を切断します */
```

```
mysqli_kill($link, $thread_id);

/* これはエラーとなります */
if (!mysqli_query($link, "CREATE TABLE myCity LIKE City")) {
    printf("Error: %s¥n", mysqli_error($link));
     exit;
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Error: MySQL server has gone away
```

参考

- mysqli_kill()

---

mysqli_thread_safe

(PHP 5)

mysqli_thread_safe — スレッドセーフであるかどうかを返す

説明

手続き型:

```
bool mysqli_thread_safe ( void )
```

クライアントライブラリがスレッドセーフでコンパイルされているかどうかを返します。

返り値

クライアントライブラリがスレッドセーフの場合に TRUE、 そうでない場合に FALSE を返します。

---

mysqli_use_result

mysqli->use_result()

(PHP 5)

mysqli->use_result() — 結果セットの取得を開始する

説明

手続き型:

```
mysqli_result mysqli_use_result ( mysqli $link )
```

オブジェクト指向型（メソッド）:

```
mysqli
mysqli_result use_result ( void )
```

データベース接続上で mysqli_real_query() 関数を使用して実行した直近のクエリから、結果セットの取得を開始します。

この関数あるいは mysqli_store_result() 関数は、 クエリ結果を取得する前にコールされる必要があります。また、どちらかの 関数をコールしなければ、データベース接続上の次のクエリは失敗します。

> **注意**: mysqli_use_result() は、データベースから結果セットの 全体を転送するわけではありません。そのため、セット内の行を移動するために mysqli_data_seek() を使用することはできません。 この機能を使用するには、mysqli_store_result() を使用して結果をバッファに取得する必要があります。クライアント側で 大量の処理を行う際は、mysqli_use_result() を 使用すべきではありません。なぜなら、この関数はサーバとの接続を保持 し続け、取得しているデータに関連するテーブルについて、他のスレッドから 更新ができなくなるからです。

返り値

バッファに取得しないで結果オブジェクトを返します。 エラー時には FALSE を返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
     exit();
```

```
}

$query  = "SELECT  CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* 複数のクエリを実行します */
if ($mysqli->multi_query($query)) {
    do {
        /* 最初の結果セットを取得します */
        if ($result = $mysqli->use_result()) {
            while ($row = $result->fetch_row()) {
                printf("%s¥n", $row[0]);
            }
            $result->close();
        }
        /* 区切り線を表示します */
        if ($mysqli->more_results()) {
            printf("-----------------¥n");
        }
    } while ($mysqli->next_result());
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$query  = "SELECT  CURRENT_USER();";
$query .= "SELECT Name FROM City ORDER BY ID LIMIT 20, 5";

/* 複数のクエリを実行します */
if (mysqli_multi_query($link, $query)) {
    do {
        /* 最初の結果セットを取得します */
        if ($result = mysqli_use_result($link)) {
            while ($row = mysqli_fetch_row($result)) {
                printf("%s¥n", $row[0]);
            }
            mysqli_free_result($result);
        }
        /* 区切り線を表示します */
        if (mysqli_more_results($link)) {
            printf("-----------------¥n");
        }
    } while (mysqli_next_result($link));
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
my_user@localhost
-----------------
Amersfoort
Maastricht
Dordrecht
Leiden
Haarlemmermeer
```

参考

- *mysqli_real_query()*
- *mysqli_store_result()*

---

mysqli_warning_count

mysqli->warning_count

(PHP 5)

mysqli->warning_count — 指定した接続の直近のクエリから発生した警告の数を返す

説明

手続き型:

```
int mysqli_warning_count ( mysqli $link )
```

オブジェクト指向型（プロパティ）：

mysqli
int$warning_count;

この接続の直近のクエリについて、発生した警告の数を返します。

注意: 警告の内容を取得するには、SQL コマンド SHOW WARNINGS [limit row_count] を使用します。

パラメータ

link

手続き型のみ: mysqli_connect() あるいは mysqli_init() が返すリンク ID。

返り値

警告の数を返します。警告がなかった場合はゼロを返します。

例

Example#1 オブジェクト指向型

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

$mysqli->query("CREATE TABLE myCity LIKE City");

/* ウェールズの珍しい地名です */
$query = "INSERT INTO myCity (CountryCode, Name) VALUES('GBR',
            'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

$mysqli->query($query);

if ($mysqli->warning_count) {
    if ($result = $mysqli->query("SHOW WARNINGS")) {
        $row = $result->fetch_row();
        printf("%s (%d): %s¥n", $row[0], $row[1], $row[2]);
        $result->close();
    }
}

/* 接続を閉じます */
$mysqli->close();
?>
```

Example#2 手続き型

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

/* 接続状況をチェックします */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s¥n", mysqli_connect_error());
    exit();
}

mysqli_query($link, "CREATE TABLE myCity LIKE City");

/* ウェールズの珍しい地名です */
$query = "INSERT INTO myCity (CountryCode, Name) VALUES('GBR',
            'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch')";

mysqli_query($link, $query);

if (mysqli_warning_count($link)) {
    if ($result = mysqli_query($link, "SHOW WARNINGS")) {
        $row = mysqli_fetch_row($result);
        printf("%s (%d): %s¥n", $row[0], $row[1], $row[2]);
        mysqli_free_result($result);
    }
}

/* 接続を閉じます */
mysqli_close($link);
?>
```

上の例の出力は以下となります。

```
Warning (1264): Data truncated for column 'Name' at row 1
```

参考

- mysqli_errno()
- mysqli_error()
- mysqli_sqlstate()

目次

- mysqli_real_connect — mysql サーバとの接続をオープンする
- mysqli_real_escape_string — 接続の現在の文字セットを考慮して、SQL 文で使用する文字列の 特殊文字をエスケープする
- mysqli_real_query — SQL クエリを実行する
- mysqli_report — 内部のレポート関数を有効あるいは無効にする
- mysqli_rollback — 現在のトランザクションをロールバックする
- mysqli_rpl_parse_enabled — RPL のパースが有効かどうかを確認する
- mysqli_rpl_probe — RPL の調査
- mysqli_rpl_query_type — RPL クエリの型を返す
- mysqli_select_db — クエリを実行するためのデフォルトのデータベースを選択する
- mysqli_send_long_data — mysqli_stmt_send_long_data のエイリアス
- mysqli_send_query — クエリを送信する
- mysqli_server_end — 組み込みのサーバをシャットダウンする
- mysqli_server_init — 組み込みのサーバを初期化する
- mysqli_set_charset — クライアントのデフォルト文字セットを設定する
- mysqli_set_local_infile_default — load local infile コマンド用のユーザ定義ハンドラを削除する
- mysqli_set_local_infile_handler — LOAD DATA LOCAL INFILE コマンド用のコールバック関数を設定する
- mysqli_set_opt — mysqli_options のエイリアス
- mysqli_slave_query — マスタ/スレーブ設定で、スレーブ側のクエリを実行する
- mysqli_sqlstate — 直前の MySQL の操作での SQLSTATE エラーを返す
- mysqli_ssl_set — SSL を使用したセキュアな接続を確立する
- mysqli_stat — 現在のシステム状態を取得する
- mysqli_stmt_affected_rows — 直近に実行されたステートメントで変更・削除・あるいは追加された行の総数を返す
- mysqli_stmt_attr_get — 説明
- mysqli_stmt_attr_set — 説明
- mysqli_stmt_bind_param — プリペアドステートメントのパラメータに変数をバインドする
- mysqli_stmt_bind_result — 結果を保存するため、プリペアドステートメントに変数をバインドする
- mysqli_stmt_close — プリペアドステートメントを閉じる
- mysqli_stmt_data_seek — ステートメントの結果セットの任意の行に移動する
- mysqli_stmt_errno — 直近のステートメントのコールに関するエラーコードを返す
- mysqli_stmt_error — 直近のステートメントのエラー内容を文字列で返す
- mysqli_stmt_execute — プリペアドクエリを実行する
- mysqli_stmt_fetch — プリペアドステートメントから結果を取得し、バインド変数に格納する
- mysqli_stmt_field_count — 指定したステートメントのフィールド数を返す
- mysqli_stmt_free_result — 指定したステートメントハンドルの結果を格納したメモリを開放する
- mysqli_stmt_get_warnings — 説明
- mysqli_stmt_init — ステートメントを初期化し、mysqli_stmt_prepare で使用するオブジェクトを返す
- mysqli_stmt_insert_id — 直近の INSERT 操作で生成した ID を取得する
- mysqli_stmt_num_rows — ステートメントの結果セットの行数を返す
- mysqli_stmt_param_count — 指定したステートメントのパラメータ数を返す
- mysqli_stmt_prepare — SQL ステートメントを実行するために準備する
- mysqli_stmt_reset — プリペアドステートメントをリセットする
- mysqli_stmt_result_metadata — プリペアドステートメントから結果セットのメタデータを返す
- mysqli_stmt_send_long_data — データをブロックで送信する
- mysqli_stmt_sqlstate — 直前のステートメントの操作での SQLSTATE エラーを返す
- mysqli_stmt_store_result — プリペアドステートメントから結果を転送する
- mysqli_store_result — 直近のクエリから結果セットを転送する
- mysqli_thread_id — 現在の接続のスレッド ID を返す
- mysqli_thread_safe — スレッドセーフであるかどうかを返す
- mysqli_use_result — 結果セットの取得を開始する
- mysqli_warning_count — 指定した接続の直近のクエリから発生した警告の数を返す

Ncurses 端末画面制御関数

導入

ncurses (new curses) は、System V Rel 4.0(及びそれ以前)の curses の フリーなソフトウエアエミュレーションです。ncurses は terminfo 型式を 使用し、パッド、カラー、複数のハイライト、フォーム文字、 ファンクションキーマッピングをサポートします。 このライブラリは対話的なものであるため、 Web アプリケーションを作成する際にはほとんど使用されませんが、 コマンドラインから PHP を使用 するスクリプトを書く際には有用です。

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

Ncurses は以下のプラットフォームで利用可能です。

- AIX

- BeOS
- Cygwin
- Digital Unix (aka OSF1)
- FreeBSD
- GNU/Linux
- HPUX
- IRIX
- OS/2
- SCO OpenServer
- Solaris
- SunOS

注意: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 6.0.0

要件

ncurses ライブラリおよびヘッダファイルが必要です。最新のバージョンを » ftp://ftp.gnu.org/pub/gnu/ncurses/ あるいは他の GNU ミラーサイトからダウンロードしてください。

インストール手順

これらの関数を動作させるには、--with-ncurses[=DIR]を指定して CGI または CLI バージョンの PHP をコンパイルする必要があります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールではウィンドウ、パネルおよびパッドリソースを定義しています。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

エラーコード

エラー時に、ncurese 関数は NCURSES_ERR を返します。

カラー

ncurses カラー定数

定数	意味
NCURSES_COLOR_BLACK	色なし(黒)
NCURSES_COLOR_WHITE	白
NCURSES_COLOR_RED	赤 - 端末がカラーモードの場合のみサポート
NCURSES_COLOR_GREEN	緑 - 端末がカラーモードの場合のみサポート
NCURSES_COLOR_YELLOW	黄 - 端末がカラーモードの場合のみサポート
NCURSES_COLOR_BLUE	青 - 端末がカラーモードの場合のみサポート
NCURSES_COLOR_CYAN	シアン - 端末がカラーモードの場合のみサポート
NCURSES_COLOR_MAGENTA	マゼンタ - 端末がカラーモードの場合のみサポート

キー

ncurses キー定数

定数	意味
NCURSES_KEY_F0 - NCURSES_KEY_F64	ファンクションキー F1 - F64
NCURSES_KEY_DOWN	下矢印
NCURSES_KEY_UP	上矢印
NCURSES_KEY_LEFT	左矢印
NCURSES_KEY_RIGHT	右矢印

定数	意味
NCURSES_KEY_HOME	ホームキー(upward+左矢印)
NCURSES_KEY_BACKSPACE	バックスペース
NCURSES_KEY_DL	行削除
NCURSES_KEY_IL	行挿入
NCURSES_KEY_DC	文字削除
NCURSES_KEY_IC	文字挿入あるいは挿入モード移行
NCURSES_KEY_EIC	文字挿入モード終了
NCURSES_KEY_CLEAR	画面消去
NCURSES_KEY_EOS	画面最下部までを消去
NCURSES_KEY_EOL	行末までを消去
NCURSES_KEY_SF	1 行スクロール
NCURSES_KEY_SR	1 行逆スクロール
NCURSES_KEY_NPAGE	次ページ
NCURSES_KEY_PPAGE	前ページ
NCURSES_KEY_STAB	タブ
NCURSES_KEY_CTAB	タブ消去
NCURSES_KEY_CATAB	全タブ消去
NCURSES_KEY_SRESET	ソフト(部分)リセット
NCURSES_KEY_RESET	リセットあるいはハードリセット
NCURSES_KEY_PRINT	印刷
NCURSES_KEY_LL	左下
NCURSES_KEY_A1	キーパッドの左上
NCURSES_KEY_A3	キーパッドの右上
NCURSES_KEY_B2	キーパッドの中央
NCURSES_KEY_C1	キーパッドの左下
NCURSES_KEY_C3	キーパッドの右下
NCURSES_KEY_BTAB	バックタブ
NCURSES_KEY_BEG	先頭
NCURSES_KEY_CANCEL	キャンセル
NCURSES_KEY_CLOSE	閉じる
NCURSES_KEY_COMMAND	cmd (コマンド)
NCURSES_KEY_COPY	コピー
NCURSES_KEY_CREATE	作成
NCURSES_KEY_END	行末
NCURSES_KEY_EXIT	終了
NCURSES_KEY_FIND	検索
NCURSES_KEY_HELP	ヘルプ
NCURSES_KEY_MARK	マーク
NCURSES_KEY_MESSAGE	メッセージ
NCURSES_KEY_MOVE	移動
NCURSES_KEY_NEXT	次
NCURSES_KEY_OPEN	オープン
NCURSES_KEY_OPTIONS	オプション
NCURSES_KEY_PREVIOUS	前
NCURSES_KEY_REDO	やり直し
NCURSES_KEY_REFERENCE	ref (参照)
NCURSES_KEY_REFRESH	リフレッシュ
NCURSES_KEY_REPLACE	置換
NCURSES_KEY_RESTART	再起動

定数	意味
NCURSES_KEY_RESUME	再開
NCURSES_KEY_SAVE	保存
NCURSES_KEY_SBEG	shiftet beg (beginning)
NCURSES_KEY_SCANCEL	shift + キャンセル
NCURSES_KEY_SCOMMAND	shift + command
NCURSES_KEY_SCOPY	shift + コピー
NCURSES_KEY_SCREATE	shift + create
NCURSES_KEY_SDC	shift + 文字削除
NCURSES_KEY_SDL	shift + 行削除
NCURSES_KEY_SELECT	選択
NCURSES_KEY_SEND	shift + end
NCURSES_KEY_SEOL	shift + 行末
NCURSES_KEY_SEXIT	shift + exit
NCURSES_KEY_SFIND	shift + 検索
NCURSES_KEY_SHELP	shift + ヘルプ
NCURSES_KEY_SHOME	shift + ホーム
NCURSES_KEY_SIC	shift + input
NCURSES_KEY_SLEFT	shift + 左矢印
NCURSES_KEY_SMESSAGE	shift + メッセージ
NCURSES_KEY_SMOVE	shift + 移動
NCURSES_KEY_SNEXT	shift + 次
NCURSES_KEY_SOPTIONS	shift + オプション
NCURSES_KEY_SPREVIOUS	shift + 前
NCURSES_KEY_SPRINT	shift + 印刷
NCURSES_KEY_SREDO	shift + やり直し
NCURSES_KEY_SREPLACE	shift + 置換
NCURSES_KEY_SRIGHT	shift + 右矢印
NCURSES_KEY_SRSUME	shift + 再開
NCURSES_KEY_SSAVE	shift + 保存
NCURSES_KEY_SSUSPEND	shift + サスペンド
NCURSES_KEY_UNDO	元に戻す
NCURSES_KEY_MOUSE	マウスイベントが発生
NCURSES_KEY_MAX	最大のキーの値

マウス

マウス定数

定数	意味
NCURSES_BUTTON1_RELEASED - NCURSES_BUTTON4_RELEASED	ボタン (1-4) が離された
NCURSES_BUTTON1_PRESSED - NCURSES_BUTTON4_PRESSED	ボタン (1-4) が押された
NCURSES_BUTTON1_CLICKED - NCURSES_BUTTON4_CLICKED	ボタン (1-4) がクリックされた
NCURSES_BUTTON1_DOUBLE_CLICKED - NCURSES_BUTTON4_DOUBLE_CLICKED	ボタン (1-4) がダブルクリックされた
NCURSES_BUTTON1_TRIPLE_CLICKED - NCURSES_BUTTON4_TRIPLE_CLICKED	ボタン (1-4) がトリプルクリックされた
NCURSES_BUTTON_CTRL	クリック中に ctrl が押された
NCURSES_BUTTON_SHIFT	クリック中に shift が押された
NCURSES_BUTTON_ALT	クリック中に alt が押された
NCURSES_ALL_MOUSE_EVENTS	すべてのマウスイベントを報告する
NCURSES_REPORT_MOUSE_POSITION	マウスの位置を報告する

ncurses_addch

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_addch — 現在の位置に文字を追加し、カーソルを進める

説明

int **ncurses_addch** (int $ch)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

ch

---

ncurses_addchnstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_addchnstr — 現在の位置に指定した長さの属性付き文字列を追加する

説明

int **ncurses_addchnstr** (string $s , int $n)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

s

n

---

ncurses_addchstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_addchstr — 現在の位置に属性付き文字列を追加する

説明

int **ncurses_addchstr** (string $s)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

s

---

ncurses_addnstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_addnstr — 現在の位置に、指定した長さの文字列を追加する

説明

int **ncurses_addnstr** (string $s , int $n)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

s

n

---

ncurses_addstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_addstr — 現在の位置にテキストを出力する

説明

```
int ncurses_addstr ( string $text )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

text

---

ncurses_assume_default_colors

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_assume_default_colors — カラー 0 のデフォルト色を定義する

説明

```
int ncurses_assume_default_colors ( int $fg , int $bg )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

fg

bg

---

ncurses_attroff

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_attroff — 指定した属性を無効とする

説明

```
int ncurses_attroff ( int $attributes )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

attributes

---

ncurses_attron

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_attron — 指定した属性を有効にする

説明

int **ncurses_attron** (int $attributes)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

attributes

---

ncurses_attrset

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_attrset — 指定した属性を設定する

説明

int **ncurses_attrset** (int $attributes)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

attributes

---

ncurses_baudrate

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_baudrate — 端末のボーレートを返す

説明

int **ncurses_baudrate** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_beep

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_beep — 端末のビープを鳴らす

説明

int **ncurses_beep** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses_beep() は、耳に聞こえる警告(ベル) を送信ます。送信できなかった場合は画面をフラッシュします。

参考

- ncurses_flash()

---

ncurses_bkgd

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_bkgd — 端末画面の背景属性を設定する

説明

int **ncurses_bkgd** (int $attrchar)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

attrchar

---

ncurses_bkgdset

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_bkgdset — 画面背景を制御する

説明

void **ncurses_bkgdset** (int $attrchar)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

attrchar

---

ncurses_border

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_border — 属性付きの文字で画面周囲に境界を描画する

説明

int **ncurses_border** (int $left , int $right , int $top , int $bottom , int $tl_corner , int $tr_corner , int $bl_corner , int $br_corner)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した線と角を、メインウィンドウの周りに描画します。

サブウィンドウの周りに境界を描画するには ncurses_wborder() を使用してください!

パラメータ

各パラメータに 0 を指定すると描画を行い、1 を指定すると描画しません。

left

right

top

bottom

tl_corner

左上隅。

tr_corner

右上隅。

bl_corner

左下隅。

br_corner

右下隅。

参考

- *ncurses_wborder()*

---

ncurses_bottom_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_bottom_panel — パネルをスタックの最下部に移動する

説明

int ***ncurses_bottom_panel*** *(resource $panel)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

---

ncurses_can_change_color

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_can_change_color — 端末の色を変更可能かどうか確認する

説明

bool ***ncurses_can_change_color*** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末が色を扱えるかどうか、 そしてプログラムで色を変更できるかどうかを返します。

返り値

端末が色を扱え、かつプログラマがそれを変更できる場合に ***TRUE****、それ以外の場合に* ***FALSE*** *を返します。*

---

ncurses_cbreak

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_cbreak — 入力のバッファリングを変更する

説明

bool ***ncurses_cbreak*** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses_cbreak() *は、行のバッファリングと 文字の処理(文字が影響を受けない割り込みやフロー制御)を無効にし、 ユーザが入力した文字をすぐにプログラムに渡します。*

返り値

TRUE *を返します。エラーが発生した場合は NCURSES_ERR を返します。*

参考

- *ncurses_nocbreak()*

---

ncurses_clear

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_clear — スクリーンをクリアする

説明

bool ***ncurses_clear*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

空白を設定せず 完全に画面を消去します。

*注意: **ncurses_clear()** は空白を設定せずに 画面を消去します。つまり、現在の背景の状態が残されるということです。 空白文字で画面を消去するには ncurses_erase() を使用します。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ncurses_erase()*

---

ncurses_clrtobot

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_clrtobot — 現在位置から最下部までスクリーンをクリアする

説明

bool ***ncurses_clrtobot*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

***ncurses_clrtobot()** は、カーソル位置から 画面の最下部までのすべての行を消去し、空白で埋めます。 **ncurses_clrtobot()** で作成される空白には 現在の背景設定が使用されます。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ncurses_clear()*
- *ncurses_clrtoeol()*

---

ncurses_clrtoeol

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_clrtoeol — 現在位置から行末までスクリーンをクリアする

説明

bool ***ncurses_clrtoeol*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

***ncurses_clrtoeol()** は、カーソル位置から 行末までを消去し、空白で埋めます。 **ncurses_clrtoeol()** で作成される空白には 現在の背景設定が使用されます。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ncurses_clear()*
- *ncurses_clrtobot()*

---

ncurses_color_content

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_color_content — 色の RGB 値を取得する

説明

int ***ncurses_color_content*** *(int $color , int &$r , int &$g , int &$b)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

color

r

g

b

---

ncurses_color_set

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_color_set — *前景/背景色を設定する*

説明

int **ncurses_color_set** *(int $pair)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

pair

---

ncurses_curs_set

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_curs_set — *カーソル状態を設定する*

説明

int **ncurses_curs_set** *(int $visibility)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

visibility

---

ncurses_def_prog_mode

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_def_prog_mode — *端末(プログラム)モードを保存する*

説明

bool **ncurses_def_prog_mode** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses_reset_prog_mode() で使用するために (curses 内の)プログラムの現在の端末モードを保存します。

返り値

成功した場合に **FALSE**、*それ以外の場合に* **TRUE** *を返します。*

参考

- *ncurses_reset_prog_mode()*

---

ncurses_def_shell_mode

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_def_shell_mode — 端末(シェル)モードを保存する

説明

bool **ncurses_def_shell_mode** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses_reset_shell_mode() で使用するために (curses 内でない)シェルの現在の端末モードを保存します。

返り値

成功した場合に **FALSE**、それ以外の場合に **TRUE** を返します。

参考

- ncurses_reset_shell_mode()

---

ncurses_define_key

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_define_key — キーコードを定義する

説明

int **ncurses_define_key** (string $definition , int $keycode)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

definition

keycode

---

ncurses_del_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_del_panel — パネルをスタックから取り除き、削除する (しかし、関連付けられているウィンドウは削除しない)

説明

bool **ncurses_del_panel** (resource $panel)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

---

ncurses_delay_output

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_delay_output — パディング文字を用いて端末出力を遅延させる

説明

int **ncurses_delay_output** (int $milliseconds)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

milliseconds

---

ncurses_delch

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_delch — 現在位置の文字を削除し、残った部分を左に移動する

説明

bool **ncurses_delch** (*void*)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

カーソルがある位置の文字を削除します。 同じ行でカーソルの右側にある文字が左側にひとつづつ移動し、 その行の最後の文字があった場所は空白で埋められます。 カーソルの位置は変わりません。

返り値

成功した場合に **FALSE**、 それ以外の場合に **TRUE** を返します。

参考

- ncurses_deleteln()

---

ncurses_deleteln

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_deleteln — 現在位置の行を削除し、残りの部分を上に上げる

説明

bool **ncurses_deleteln** (*void*)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

カーソル位置の行を削除します。 現在の行より下の行は 1 行ずつ上に移動します。 ウィンドウの最下行はクリアされます。カーソルの位置は変わりません。

返り値

成功した場合に **FALSE**、それ以外の場合に **TRUE** を返します。

参考

- ncurses_delch()

---

ncurses_delwin

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_delwin — ncurses ウインドウを削除する

説明

bool **ncurses_delwin** (*resource $window*)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_doupdate

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_doupdate — 準備中の全ての出力を書き込み、端末をリフレッシュする

説明

bool **ncurses_doupdate** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

仮想スクリーンと物理スクリーンを比較し、 物理スクリーンを更新します。リフレッシュを何度もコールするよりも、 この方法のほうが効率的です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ncurses_echo

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_echo — キーボード入力のエコーを有効とする

説明

bool **ncurses_echo** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

エコーモードを有効にします。 ユーザがタイプした文字がすべて ncurses_getch() によってエコーされます。

返り値

成功した場合に **FALSE**、何らかのエラーが発生した場合に **TRUE** を返します。

参考

- エコーモードを無効にするには ncurses_noecho() を使用します。

---

ncurses_echochar

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_echochar — リフレッシュを行いつつ 1 文字出力する

説明

int **ncurses_echochar** (int $character)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

character

---

ncurses_end

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_end — ncurses を終了し、画面を消去する

説明

int **ncurses_end** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_erase

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_erase — 端末画面を消去する

説明

bool **ncurses_erase** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末の画面を空白で埋めます。

作成された空白の背景処理は、ncurses_bkgd() で設定したものとなります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ncurses_bkgd()
- ncurses_clear()

---

ncurses_erasechar

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_erasechar — 現在の erase 文字を返す

説明

string **ncurses_erasechar** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の erase 文字を返します。

返り値

現在の erase 文字を文字列で返します。

参考

- ncurses_killchar()

---

ncurses_filter

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_filter — iniscr() および newterm() の LINES を 1 に設定する

説明

void **ncurses_filter** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_flash

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_flash — 端末画面をフラッシュする(ビジュアルベル)

説明

bool **ncurses_flash** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

画面をフラッシュします。 もしそれができなかった場合は、音声の警告(ベル)を送信します。

返り値

成功した場合に **FALSE**、そうでない場合に **TRUE** を返します。

参考

- ncurses_beep()

---

ncurses_flushinp

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_flushinp — キーボード入力バッファをフラッシュする

説明

bool **ncurses_flushinp** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

タイプされた内容のうち、 まだプログラムで読み込まれていないものををすべて捨てます。

返り値

成功した場合に **FALSE**、それ以外の場合に **TRUE** を返します。

---

ncurses_getch

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_getch — キーボードから 1 文字読み込む

説明

int **ncurses_getch** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_getmaxyx

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_getmaxyx — ウィンドウの大きさを返す

説明

void **ncurses_getmaxyx** (resource $window , int &$y , int &$x)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したウィンドウ window の水平方向・垂直方向の大きさを取得します。

変数は参照渡しとする必要があり、 ユーザが端末の大きさを変更した際にこの変数が更新されます。

パラメータ

window

調べるウィンドウ。

x

ウィンドウの幅が設定されます。

y

ウィンドウの高さが設定されます。

返り値

値を返しません。

---

ncurses_getmouse

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_getmouse — マウスイベントを読みこむ

説明

bool **ncurses_getmouse** (array &$mevent)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses_getmouse() は、 キューからマウスイベントを読み込みます。

パラメータ

mevent

イベントのオプションを指定します。 これは配列への参照として渡されます(以下の例を参照ください)。

成功した場合、以下のキーを持つ連想配列が返されます。

- "id" : 複数デバイスを識別する ID。
- "x" : 画面上の相対的な x 位置(文字単位)。
- "y" : 画面上の相対的な y 位置(文字単位)。
- "z" : 現在はサポートされていません。
- "mmask" : マウスアクション。

返り値

指定したウィンドウでマウスイベントが実際に見える場合に **FALSE**、そうでない場合に **TRUE** を返します。

例

Example#1 ncurses_getmouse() の例

```
<?php
switch (ncurses_getch()){
  case NCURSES_KEY_MOUSE:
    if (!ncurses_getmouse(&$mevent)){
      if ($mevent["mmask"] & NCURSES_MOUSE_BUTTON1_PRESSED){
        $mouse_x = $mevent["x"]; //  マウスの位置を保存します
        $mouse_y = $mevent["y"];
      }
    }
  break;

  default:
    /* .... */
}
?>
```

参考

- ncurses_ungetmouse()

---

ncurses_getyx

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_getyx — ウィンドウ内の現在のカーソル位置を返す

説明

void **ncurses_getyx** (resource $window , int &$y , int &$x)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

y

x

---

ncurses_halfdelay

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_halfdelay — 端末をハーフディレイモードにする

説明

int **ncurses_halfdelay** (int $tenth)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

tenth

---

ncurses_has_colors

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_has_colors — カラー端末かどうか確認する

説明

bool **ncurses_has_colors** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末がカラー表示機能を持つかどうかを調べます。

返り値

端末がカラー表示機能を持っている場合に **TRUE**、持っていない場合に **FALSE** を返します。

参考

- ncurses_can_change_color()

---

ncurses_has_ic

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_has_ic — 挿入/削除機能の有無を確認する

説明

bool **ncurses_has_ic** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末が挿入/削除機能を持つかどうかを調べます。

返り値

端末が挿入/削除機能を持つ場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- ncurses_has_il()

---

ncurses_has_il

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_has_il — 行挿入/削除機能の有無を確認する

説明

bool **ncurses_has_il** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末が行挿入/削除機能を持つかどうかを調べます。

返り値

端末が行挿入/削除機能を持つ場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- ncurses_has_ic()

---

ncurses_has_key

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_has_key — 端末キーボードにおいてファンクションキーの有無を調べる

説明

int **ncurses_has_key** (int $keycode)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

keycode

---

ncurses_hide_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_hide_panel — パネルをスタックから取り除き、見えなくする

説明

int **ncurses_hide_panel** (resource $panel)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

---

ncurses_hline

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_hline — 現在位置に属性付きの文字を用いて最大 n 文字長の線を水平に描画する

説明

int **ncurses_hline** (int $charattr , int $n)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

charattr

n

---

ncurses_inch

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_inch — 現在位置の文字と属性を取得する

説明

string **ncurses_inch** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在位置の文字を返します。

返り値

文字を返します。

---

ncurses_init_color

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_init_color — 新規に RGB 値を設定する

説明

int **ncurses_init_color** (int $color , int $r , int $g , int $b)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

color

r

g

b

---

ncurses_init_pair

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_init_pair — 色の組を確保する

説明

int **ncurses_init_pair** (int $pair , int $fg , int $bg)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

pair

fg

bg

---

ncurses_init

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_init — ncurses を初期化する

説明

void ***ncurses_init*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses インターフェイスを初期化します。 必ず、その他の ncurses 関数の前に使用する必要があります。

返り値

値を返しません。

---

ncurses_insch

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_insch — 文字を挿入し、現在位置にある文字を含む残りの行を移動する

説明

int ***ncurses_insch*** *(int $character)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

character

---

ncurses_insdelln

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_insdelln — 現在の行の後に複数の行を挿入し、スクロールダウンする（負の数を指定すると削除し、スクロールアップする）

説明

int ***ncurses_insdelln*** *(int $count)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

count

---

ncurses_insertln

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_insertln — 行を挿入し、残りの部分をスクロールダウンする

説明

int ***ncurses_insertln*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の行の上に新しい行を挿入します。いちばん下の行は失われます。

---

ncurses_insstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_insstr — 現在位置に文字列を挿入し、残りの行を右に移動する

説明

int ***ncurses_insstr*** *(string $text)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

text

---

ncurses_instr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_instr — 端末画面から文字列を読み込む

説明

int ***ncurses_instr*** *(string &$buffer)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の位置から行末までを読み込み、その文字数を返します。

パラメータ

buffer

読み込んだ文字列。文字の属性は削除されます。

返り値

文字の数を返します。

---

ncurses_isendwin

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_isendwin — Ncurses が endwin モードの場合、通常の画面出力が実行可能

説明

bool ***ncurses_isendwin*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses が endwin モードであるかどうかを調べます。

返り値

ncurses_endwin() *がコールされた後に 続けて ncurses_wrefresh() がコールされていない場合に* ***TRUE****、それ以外の場合に* ***FALSE*** *を返します。*

参考

- ***ncurses_endwin()***
- *ncurses_wrefresh()*

---

ncurses_keyok

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_keyok — キーコードを有効または無効にする

説明

int ***ncurses_keyok*** *(int $keycode , bool $enable)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

keycode

enable

---

ncurses_keypad

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_keypad — キーパッドを on あるいは off にする

説明

int **ncurses_keypad** (resource $window , bool $bf)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

bf

---

ncurses_killchar

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_killchar — 現在の行削除文字を返す

説明

string **ncurses_killchar** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の行削除文字を返します。

返り値

現在の行削除文字を返します。

参考

- ncurses_erasechar()

---

ncurses_longname

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_longname — 端末の説明を返す

説明

string **ncurses_longname** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末の詳細な説明を返します。

返り値

端末の詳細な説明を返します。 説明は、最大 128 文字までで切り詰められます。 エラー時には **NULL** を返します。

参考

- ncurses_termname()

---

ncurses_meta

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_meta — 8 ビットのメタキー情報を有効/無効にする

説明

int **ncurses_meta** (resource $window , bool $8bit)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

8bit

---

ncurses_mouse_trafo

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mouse_trafo — 座標を変換する

説明

bool **ncurses_mouse_trafo** (int &$y , int &$x , bool $toscreen)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

toscreen

---

ncurses_mouseinterval

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_mouseinterval — マウスボタンクリックのタイムアウトを設定する

説明

int **ncurses_mouseinterval** (int $milliseconds)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

milliseconds

---

ncurses_mousemask

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mousemask — マウスオプションを設定する

説明

int **ncurses_mousemask** (int $newmask , int &$oldmask)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

報告されるマウスイベントを設定します。デフォルトでは、 どのマウスイベントについても報告されません。

マウスイベントは、 ncurses_wgetch() 入力ストリーム内では NCURSES_KEY_MOUSE で表されます。 イベントデータを読み込んでキューからイベントを取り出すには、 ncurses_getmouse() をコールします。

パラメータ

newmask

マウスマスクオプションには、以下の定義済み定数が指定可能です。

- NCURSES_BUTTON1_PRESSED
- NCURSES_BUTTON1_RELEASED
- NCURSES_BUTTON1_CLICKED
- NCURSES_BUTTON1_DOUBLE_CLICKED
- NCURSES_BUTTON1_TRIPLE_CLICKED
- NCURSES_BUTTON2_PRESSED
- NCURSES_BUTTON2_RELEASED
- NCURSES_BUTTON2_CLICKED
- NCURSES_BUTTON2_DOUBLE_CLICKED
- NCURSES_BUTTON2_TRIPLE_CLICKED
- NCURSES_BUTTON3_PRESSED
- NCURSES_BUTTON3_RELEASED
- NCURSES_BUTTON3_CLICKED
- NCURSES_BUTTON3_DOUBLE_CLICKED
- NCURSES_BUTTON3_TRIPLE_CLICKED
- NCURSES_BUTTON4_PRESSED
- NCURSES_BUTTON4_RELEASED
- NCURSES_BUTTON4_CLICKED
- NCURSES_BUTTON4_DOUBLE_CLICKED
- NCURSES_BUTTON4_TRIPLE_CLICKED
- NCURSES_BUTTON_SHIFT>
- NCURSES_BUTTON_CTRL
- NCURSES_BUTTON_ALT
- NCURSES_ALL_MOUSE_EVENTS
- NCURSES_REPORT_MOUSE_POSITION

副作用として、newmask にゼロを設定すると マウスポインタを消去します。ゼロ以外の値を設定すると マウスポインタが表示されます。

oldmask

以前のマウスイベントマスクの値が設定されます。

返り値

newmask が指定するイベントを報告することが できるマスクを返します。失敗した場合は 0 を返します。

例

Example#1 ncurses_mousemask() の例

```
<?php
$newmask = NCURSES_BUTTON1_CLICKED + NCURSES_BUTTON1_RELEASED;
$mask = ncurses_mousemask($newmask, &$oldmask);
if ($mask & $newmask){
    printf("All specified mouse options will be supported¥n");
}
?>
```

参考

- ncurses_getmouse()
- ncurses_ungetmouse()
- **ncurese_getch()**

---

ncurses_move_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_move_panel — 左上が [startx, starty] となるようにパネルを移動する

説明

int **ncurses_move_panel** (resource $panel , int $startx , int $starty)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

startx

starty

---

ncurses_move

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_move — 出力位置を移動する

説明

int **ncurses_move** (int $y , int $x)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

---

ncurses_mvaddch

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvaddch — 現在位置を移動し、文字を追加する

説明

int **ncurses_mvaddch** (int $y , int $x , int $c)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

c

---

ncurses_mvaddchnstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvaddchnstr — 位置を移動し、指定長の属性付きの文字列を追加する

説明

int **ncurses_mvaddchnstr** (int $y , int $x , string $s , int $n)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

s

n

---

ncurses_mvaddchstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvaddchstr — 位置を移動し、属性付きの文字列を追加する

説明

int ***ncurses_mvaddchstr*** *(int $y , int $x , string $s)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

s

---

ncurses_mvaddnstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvaddnstr — 位置を移動し、指定長の文字列を追加する

説明

int ***ncurses_mvaddnstr*** *(int $y , int $x , string $s , int $n)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

s

n

---

ncurses_mvaddstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvaddstr — 位置を移動し、文字列を追加する

説明

int ***ncurses_mvaddstr*** *(int $y , int $x , string $s)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

s

---

ncurses_mvcur

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvcur — 直ちにカーソルを移動する

説明

int **ncurses_mvcur** (int $old_y , int $old_x , int $new_y , int $new_x)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

old_y

old_x

new_y

new_x

---

ncurses_mvdelch

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvdelch — 位置を移動し、文字を削除、行の残りを左シフトする

説明

int **ncurses_mvdelch** (int $y , int $x)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

---

ncurses_mvgetch

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvgetch — 位置を移動し、新しい位置で文字を得る

説明

int **ncurses_mvgetch** (int $y , int $x)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

---

ncurses_mvhline

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvhline — 位置を新しく設定し、属性付きの文字を用いて最大n文字の水平線を描画

説明

int **ncurses_mvhline** (int $y , int $x , int $attrchar , int $n)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

attrchar

n

---

ncurses_mvinch

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvinch — 位置を移動し、新しい位置の属性付きの文字を取得する

説明

int **ncurses_mvinch** (int $y , int $x)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

---

ncurses_mvvline

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvvline — 位置を新しく設定し、属性付きの文字を用いて最大 n 文字の垂直線を描画する

説明

int **ncurses_mvvline** (int $y , int $x , int $attrchar , int $n)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

y

x

attrchar

n

---

ncurses_mvwaddstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_mvwaddstr — ウインドウの新規位置に文字列を追加する

説明

int **ncurses_mvwaddstr** (resource $window , int $y , int $x , string $text)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

y

x

text

---

ncurses_napms

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_napms — スリープ

説明

int ***ncurses_napms*** *(int $milliseconds)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

milliseconds

---

ncurses_new_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_new_panel — 新しいパネルを作成し、それをウィンドウに関連づける

説明

resource ***ncurses_new_panel*** *(resource $window)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_newpad

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_newpad — 新しいパッド (window) を作成する

説明

resource ***ncurses_newpad*** *(int $rows , int $cols)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

rows

cols

---

ncurses_newwin

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_newwin — 新規ウインドウを作成する

説明

resource **ncurses_newwin** (*int $rows* , *int $cols* , *int $y* , *int $x*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

描画要素を入れるための新しいウィンドウを作成します。

端末の大きさはそれぞれ異なりいろいろな値をとる可能性があるので、 ウィンドウを追加する際には ncurses_getmaxyx() を使用して使用可能な領域を調べることを忘れないようにしましょう。

パラメータ

rows

行の数。

cols

列の数。

y

原点の y 座標。

x

原点の x 座標。

返り値

新しいウィンドウのリソース ID を返します。

---

ncurses_nl

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_nl — 改行と復改/ラインフィードを変換する

説明

bool **ncurses_nl** (*void*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_nocbreak

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_nocbreak — 端末を cooked モードに変更する

説明

bool **ncurses_nocbreak** (*void*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

*端末を通常モード (cooked モード) に戻します。モードが継承されている場合、端末の 書記モードは cbreak モードであったりそうでなかったりするかもしれません。 そのため、プログラムでは ncurses_cbreak() および **ncurses_nocbreak()** を明示的にコールすべきです。*

返り値

*何らかのエラーが発生した場合に **TRUE**、 それ以外の場合に **FALSE** を返します。*

参考

- *ncurses_cbreak()*

---

ncurses_noecho

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_noecho — キーボード入力エコーを無効にする

説明

bool **ncurses_noecho** (*void*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ユーザが入力した文字のエコーを無効にします。

返り値

*何らかのエラーが発生した場合に **TRUE**、 それ以外の場合に **FALSE** を返します。*

参考

- *ncurses_echo()*
- *ncurses_getch()*

---

ncurses_nonl

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_nonl — 改行と復改/ラインフィードを変換しない

説明

bool **ncurses_nonl** (*void*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_noqiflush

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_noqiflush — シグナル文字のフラッシュを無効とする

説明

void **ncurses_noqiflush** (*void*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_noraw

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_noraw — 端末を raw モード以外に変更する

説明

bool **ncurses_noraw** (*void*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末を raw モード以外に変更します。raw モードは cbreak モードと似ており、 タイプされた文字はすぐにプログラムへ渡されます。違う点は、raw モードの場合は 中断 (interrupt)、終了 (quit)、停止 (suspend) およびフロー制御文字もそのまま渡され、シグナルは発生しないということです。

返り値

*何らかのエラーが発生した場合に **TRUE**、 それ以外の場合に **FALSE** を返します。*

参考

- *ncurses_raw()*
- *ncurses_cbreak()*

- ncurses_nocbreak()

---

ncurses_pair_content

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_pair_content — 色の RGB 値を取得する

説明

int **ncurses_pair_content** (int $pair , int &$f , int &$b)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

pair

f

b

---

ncurses_panel_above

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_panel_above — パネルの上のパネルを返す

説明

resource **ncurses_panel_above** (resource $panel)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

返り値

パネルが null の場合は、スタックの最下部のパネルを返します。

---

ncurses_panel_below

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_panel_below — パネルの下のパネルを返す

説明

resource **ncurses_panel_below** (resource $panel)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

パラメータ

パネルが null の場合は、スタックの最上部のパネルを返します。

---

ncurses_panel_window

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_panel_window — パネルに関連付けられたウィンドウを返す

説明

resource **ncurses_panel_window** (resource $panel)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

---

ncurses_pnoutrefresh

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_pnoutrefresh — パッドから仮想画面にリージョンをコピーする

説明

int ***ncurses_pnoutrefresh*** *(resource $pad , int $pminrow , int $pmincol , int $sminrow , int $smincol , int $smaxrow , int $smaxcol)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

pad

pminrow

pmincol

sminrow

smincol

smaxrow

smaxcol

---

ncurses_prefresh

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_prefresh — パッドから仮想画面にリージョンをコピーする

説明

int ***ncurses_prefresh*** *(resource $pad , int $pminrow , int $pmincol , int $sminrow , int $smincol , int $smaxrow , int $smaxcol)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

pad

pminrow

pmincol

sminrow

smincol

smaxrow

smaxcol

---

ncurses_putp

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_putp — パディング情報を文字列に適用し、それを出力する

説明

int ***ncurses_putp*** *(string $text)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

text

---

ncurses_qiflush

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_qiflush — シグナル文字のフラッシュを有効とする

説明

void **ncurses_qiflush** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_raw

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_raw — 端末を raw モードに変更する

説明

bool **ncurses_raw** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

端末を raw モードに変更します。row モードは cbreak モードと似ており、 タイプされた文字はすぐにプログラムへ渡されます。違う点は、raw モードの場合は 中断 (interrupt)、終了 (quit)、停止 (suspend) およびフロー制御文字もそのまま渡され、シグナルは発生しないということです。

返り値

何らかのエラーが発生した場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

参考

- ncurses_noraw()
- ncurses_cbreak()
- ncurses_nocbreak()

---

ncurses_refresh

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_refresh — 画面をリフレッシュする

説明

int **ncurses_refresh** (int $ch)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

ch

---

ncurses_replace_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_replace_panel — パネルに関連付けられたウィンドウを置き換える

説明

int **ncurses_replace_panel** (resource $panel , resource $window)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

window

---

ncurses_reset_prog_mode

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_reset_prog_mode — def_prog_mode で保存したプログラムモードをリセットする

説明

int **ncurses_reset_prog_mode** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_reset_shell_mode

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_reset_shell_mode — def_shell_mode で保存したシェルモードをリセットする

説明

int **ncurses_reset_shell_mode** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_resetty

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_resetty — 保存した端末モードに復帰する

説明

bool **ncurses_resetty** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

事前に ncurses_savetty() をコールすることによって保存しておいた端末モードを復元します。

返り値

常に **FALSE** を返します。

参考

- ncurses_savetty()

---

ncurses_savetty

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_savetty — 端末の状態を保存する

説明

bool **ncurses_savetty** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在の端末の状態を保存します。 保存された状態は、関数 ncurses_resetty() によって復元することが可能です。

返り値

常に **FALSE** を返します。

参考

- ncurses_resetty()

---

ncurses_scr_dump

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_scr_dump — 画面の内容をファイルにダンプする

説明

int **ncurses_scr_dump** (string $filename)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

filename

---

ncurses_scr_init

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_scr_init — ファイルダンプから画面を初期化する

説明

int **ncurses_scr_init** (string $filename)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

filename

---

ncurses_scr_restore

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_scr_restore — ファイルダンプから画面を復帰する

説明

int **ncurses_scr_restore** (string $filename)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

filename

---

ncurses_scr_set

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_scr_set — ファイルダンプから画面を継承する

説明

int ***ncurses_scr_set*** *(string $filename)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

filename

---

ncurses_scrl

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_scrl — 現在位置を変更せずに画面の内容をスクロールアップまたはダウンする

説明

int ***ncurses_scrl*** *(int $count)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

count

---

ncurses_show_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_show_panel — 不可視のパネルをスタックの最上部に置き、見えるようにする

説明

int ***ncurses_show_panel*** *(resource $panel)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

---

ncurses_slk_attr

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_attr — 現在のソフトラベルキー属性を返す

説明

int ***ncurses_slk_attr*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

現在のソフトラベルキー属性を返します。

返り値

属性を整数値で返します。

---

ncurses_slk_attroff

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_attroff — ソフトファンクションキーラベルの指定した属性を無効にする

説明

int **ncurses_slk_attroff** (*int $intarg*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

intarg

---

ncurses_slk_attron

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_attron — *ソフトファンクションキーラベルの指定した属性を有効にする*

説明

int **ncurses_slk_attron** (*int $intarg*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

intarg

---

ncurses_slk_attrset

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_attrset — *ソフトファンクションキーラベルに、指定した属性を設定する*

説明

int **ncurses_slk_attrset** (*int $intarg*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

intarg

---

ncurses_slk_clear

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_clear — *画面からソフトラベルをクリアする*

説明

bool **ncurses_slk_clear** (*void*)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

*関数 **ncurses_slk_clear()** は、 画面からソフトラベルキーを消去します。*

返り値

*エラー時に **TRUE**、 それ以外の場合に **FALSE** を返します。*

---

ncurses_slk_color

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_color — ソフトラベルキーの色を設定する

説明

int **ncurses_slk_color** (int $intarg)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

intarg

---

ncurses_slk_init

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_init — ソフトラベルキー関数を初期化する

説明

bool **ncurses_slk_init** (int $format)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ソフトラベルキー関数を初期化します。

この関数は、必ず **ncurses_initscr()** あるいは **ncurses_newterm()** がコールされる前にコールする 必要があります。

パラメータ

format

ncurses_initscr() が、 最終的にソフトラベルをエミュレートするために stdscr からの線を使用する場合、 このパラメータで画面上でのラベルの配置方法を指定します。

0 にするとラベルを 3-2-3 形式に配置し、1 にすると 4-4 形式に配置します。 また 2 にすると PC 風の 4-4-4 形式に配置しますが、それに加えて インデックスラインが作成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ncurses_slk_noutrefresh

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_noutrefresh — 仮想画面にソフトラベルキーをコピーする

説明

bool **ncurses_slk_noutrefresh** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_slk_refresh

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_refresh — ソフトラベルキーを画面にコピーする

説明

int **ncurses_slk_refresh** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ソフトラベルキーを仮想画面から実際の画面にコピーします。

---

ncurses_slk_restore

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_restore — ソフトラベルキーを復帰する

説明

int **ncurses_slk_restore** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

ncurses_slk_clear() が実行された後に ソフトラベルキーを復元します。

---

ncurses_slk_set

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_set — ファンクションキーラベルを設定する

説明

bool **ncurses_slk_set** (int $labelnr , string $label , int $format)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

labelnr

label

format

---

ncurses_slk_touch

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_slk_touch — ncurses_slk_noutrefresh を実行する際に強制的に出力する

説明

int **ncurses_slk_touch** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

次に ncurses_slk_noutrefresh() が実行された際に すべてのソフトラベルを強制的に出力するようにします。

---

ncurses_standend

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_standend — 'standout' 属性の使用を停止する

説明

int **ncurses_standend** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_standout

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_standout — 'standout' 属性の使用を開始する

説明

int **ncurses_standout** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_start_color

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_start_color — 色の使用を開始する

説明

int **ncurses_start_color** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_termattrs

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_termattrs — 端末でサポートされる全ての属性フラグの論理和を返す

説明

bool **ncurses_termattrs** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_termname

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_termname — 端末の(簡略)名を返す

説明

string **ncurses_termname** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

Returns terminals shortname.

返り値

端末の簡略名を返します。 簡略名は最大 14 文字で切り詰められます。エラー時には **NULL** を返します。

参考

- ncurses_longname()

---

ncurses_timeout

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_timeout — 特別なキーシーケンスのタイムアウトを設定する

説明

void **ncurses_timeout** (int $millisec)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

millisec

---

ncurses_top_panel

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_top_panel — 可視パネルをスタックの最上部に移動する

説明

int **ncurses_top_panel** (resource $panel)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

panel

---

ncurses_typeahead

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_typeahead — typeahead 確認用に別のファイル記述子を指定する

説明

int **ncurses_typeahead** (int $fd)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

fd

---

ncurses_ungetch

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_ungetch — 入力ストリームに 1 文字戻す

説明

int **ncurses_ungetch** (int $keycode)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

keycode

---

ncurses_ungetmouse

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_ungetmouse — マウスイベントをキューにプッシュする

説明

bool **ncurses_ungetmouse** (array $mevent)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

キューに KEY_MOUSE イベントをプッシュし、このイベントを、mevent で指定した状態・画面上の位置座標と関連付けます。

パラメータ

mevent

イベントのオプションを指定する連想配列。

- "id" : 複数デバイスを識別する ID。
- "x" : 画面上の相対的な x 位置(文字単位)。
- "y" : 画面上の相対的な y 位置(文字単位)。
- "z" : 現在はサポートされていません。
- "mmask" : マウスアクション

返り値

成功した場合に **FALSE**、それ以外の場合に **TRUE** を返します。

参考

- ncurses_getmouse()

---

ncurses_update_panels

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_update_panels — 仮想画面を再描画し、スタック内のパネルとの関係を反映させる

説明

void **ncurses_update_panels** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_use_default_colors

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_use_default_colors — 端末のデフォルト色をカラー ID -1 に割り付ける

説明

bool **ncurses_use_default_colors** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

ncurses_use_env

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_use_env — 端末の大きさに関する環境情報の使用を制御する

説明

void **ncurses_use_env** (bool $flag)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

flag

---

ncurses_use_extended_names

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_use_extended_names — terminfo 記述において拡張名の使用を制御する

説明

int **ncurses_use_extended_names** *(bool $flag)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

flag

---

ncurses_vidattr

(PHP 4 >= 4.0.7, PHP 5, PECL ncurses:1.0.0)

ncurses_vidattr — video attribute モードで、端末上に文字列を表示する

説明

int **ncurses_vidattr** *(int $intarg)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

intarg

---

ncurses_vline

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_vline — 現在位置に最大 n 文字の属性付きの文字を用いて垂直線を描画する

説明

int **ncurses_vline** *(int $charattr , int $n)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

charattr

n

---

ncurses_waddch

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_waddch — ウィンドウ内の現在位置に文字を追加し、カーソルを進める

説明

int **ncurses_waddch** (resource $window , int $ch)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

ch

---

ncurses_waddstr

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_waddstr — ウィンドウ内の現在位置にテキストを出力する

説明

int **ncurses_waddstr** (resource $window , string $str [, int $n])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

str

n

---

ncurses_wattroff

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wattroff — ウィンドウの属性をオフにする

説明

int **ncurses_wattroff** (resource $window , int $attrs)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

attrs

---

ncurses_wattron

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wattron — ウィンドウの属性をオンにする

説明

int **ncurses_wattron** (resource $window , int $attrs)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

attrs

---

ncurses_wattrset

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wattrset — ウィンドウの属性を設定する

説明

```
int ncurses_wattrset ( resource $window , int $attrs )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

attrs

---

ncurses_wborder

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wborder — 属性文字を使用してウィンドウの周囲に線を描画する

説明

```
int ncurses_wborder ( resource $window , int $left , int $right , int $top , int $bottom , int $tl_corner , int $tr_corner , int $bl_corner , int $br_corner )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

window で指定したウィンドウの周囲を指定した線と角で囲みます。

メインウィンドウの周囲を囲むには、ncurses_border() を使用します。

パラメータ

各パラメータに 0 を渡すとその部分を描画し、1 を渡すと描画しません。

window

操作するウィンドウ。

left

right

top

bottom

tl_corner

左上隅。

tr_corner

右上隅。

bl_corner

左下隅。

br_corner

右下隅。

参考

- ncurses_border()

---

ncurses_wclear

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wclear — ウィンドウをクリアする

説明

```
int ncurses_wclear ( resource $window )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_wcolor_set

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wcolor_set — ウィンドウの色の組み合わせを設定する

説明

int **ncurses_wcolor_set** (resource $window , int $color_pair)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

color_pair

---

ncurses_werase

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_werase — ウィンドウを消去する

説明

int **ncurses_werase** (resource $window)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_wgetch

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wgetch — キーボート（ウィンドウ）から文字を読み込む

説明

int **ncurses_wgetch** (resource $window)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_whline

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_whline — 指定した属性文字を用いて、最大 n 文字分の長さの水平線を ウィンドウに描画する

説明

int **ncurses_whline** (resource $window , int $charattr , int $n)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

charattr

n

---

ncurses_wmouse_trafo

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wmouse_trafo — ウィンドウ/標準画面の座標系を変換する

説明

bool **ncurses_wmouse_trafo** (resource $window , int &$y , int &$x , bool $toscreen)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

x

y

toscreen

---

ncurses_wmove

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wmove — ウィンドウの出力位置を移動する

説明

int **ncurses_wmove** (resource $window , int $y , int $x)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

y

x

---

ncurses_wnoutrefresh

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wnoutrefresh — ウィンドウを仮想画面にコピーする

説明

int **ncurses_wnoutrefresh** (resource $window)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_wrefresh

(PHP 4 >= 4.2.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wrefresh — 端末画面のウインドウをリフレッシュする

説明

int **ncurses_wrefresh** (resource $window)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_wstandend

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wstandend — ウィンドウの standout モードを終了する

説明

int **ncurses_wstandend** (resource $window)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_wstandout

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wstandout — ウィンドウの standout モードに入る

説明

int **ncurses_wstandout** (resource $window)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

---

ncurses_wvline

(PHP 4 >= 4.3.0, PHP 5, PECL ncurses:1.0.0)

ncurses_wvline — 指定した属性文字を用いて、最大 n 文字分の長さの垂直線を ウィンドウに描画する

説明

int **ncurses_wvline** (resource $window , int $charattr , int $n)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

window

charattr

n

目次

- ncurses_clear — スクリーンをクリアする
- ncurses_clrtobot — 現在位置から最下部までスクリーンをクリアする
- ncurses_clrtoeol — 現在位置から行末までスクリーンをクリアする
- ncurses_color_content — 色の RGB 値を取得する
- ncurses_color_set — 前景/背景色を設定する
- ncurses_curs_set — カーソル状態を設定する
- ncurses_def_prog_mode — 端末(プログラム)モードを保存する
- ncurses_def_shell_mode — 端末(シェル)モードを保存する
- ncurses_define_key — キーコードを定義する
- ncurses_del_panel — パネルをスタックから取り除き、削除する (しかし、関連付けられているウィンドウは削除しない)
- ncurses_delay_output — パディング文字を用いて端末出力を遅延させる
- ncurses_delch — 現在位置の文字を削除し、残った部分を左に移動する
- ncurses_deleteln — 現在位置の行を削除し、残りの部分を上に上げる
- ncurses_delwin — ncurses ウインドウを削除する
- ncurses_doupdate — 準備中の全ての出力を書き込み、端末をリフレッシュする
- ncurses_echo — キーボード入力のエコーを有効とする
- ncurses_echochar — リフレッシュを行いつつ 1 文字出力する
- ncurses_end — ncurses を終了し、画面を消去する
- ncurses_erase — 端末画面を消去する
- ncurses_erasechar — 現在の erase 文字を返す
- ncurses_filter — iniscr() および newterm() の LINES を 1 に設定する
- ncurses_flash — 端末画面をフラッシュする(ビジュアルベル)
- ncurses_flushinp — キーボード入力バッファをフラッシュする
- ncurses_getch — キーボードから 1 文字読み込む
- ncurses_getmaxyx — ウィンドウの大きさを返す
- ncurses_getmouse — マウスイベントを読みこむ
- ncurses_getyx — ウィンドウ内の現在のカーソル位置を返す
- ncurses_halfdelay — 端末をハーフディレイモードにする
- ncurses_has_colors — カラー端末かどうか確認する
- ncurses_has_ic — 挿入/削除機能の有無を確認する
- ncurses_has_il — 行挿入/削除機能の有無を確認する
- ncurses_has_key — 端末キーボードにおいてファンクションキーの有無を調べる
- ncurses_hide_panel — パネルをスタックから取り除き、見えなくする
- ncurses_hline — 現在位置に属性付きの文字を用いて最大 n 文字長の線を水平に描画する
- ncurses_inch — 現在位置の文字と属性を取得する
- ncurses_init_color — 新規に RGB 値を設定する
- ncurses_init_pair — 色の組を確保する
- ncurses_init — ncurses を初期化する
- ncurses_insch — 文字を挿入し、現在位置にある文字を含む残りの行を移動する
- ncurses_insdelln — 現在の行の後に複数の行を挿入し、スクロールダウンする (負の数を指定すると削除し、スクロールアップする)
- ncurses_insertln — 行を挿入し、残りの部分をスクロールダウンする
- ncurses_insstr — 現在位置に文字列を挿入し、残りの行を右に移動する
- ncurses_instr — 端末画面から文字列を読み込む
- ncurses_isendwin — Ncurses が endwin モードの場合、通常の画面出力が実行可能
- ncurses_keyok — キーコードを有効または無効にする
- ncurses_keypad — キーパッドを on あるいは off にする
- ncurses_killchar — 現在の行削除文字を返す
- ncurses_longname — 端末の説明を返す
- ncurses_meta — 8 ビットのメタキー情報を有効/無効にする
- ncurses_mouse_trafo — 座標を変換する
- ncurses_mouseinterval — マウスボタンクリックのタイムアウトを設定する
- ncurses_mousemask — マウスオプションを設定する
- ncurses_move_panel — 左上が [startx, starty] となるようにパネルを移動する
- ncurses_move — 出力位置を移動する
- ncurses_mvaddch — 現在位置を移動し、文字を追加する
- ncurses_mvaddchnstr — 位置を移動し、指定長の属性付きの文字列を追加する
- ncurses_mvaddchstr — 位置を移動し、属性付きの文字列を追加する
- ncurses_mvaddnstr — 位置を移動し、指定長の文字列を追加する
- ncurses_mvaddstr — 位置を移動し、文字列を追加する
- ncurses_mvcur — 直ちにカーソルを移動する
- ncurses_mvdelch — 位置を移動し、文字を削除、行の残りを左シフトする
- ncurses_mvgetch — 位置を移動し、新しい位置で文字を得る
- ncurses_mvhline — 位置を新しく設定し、属性付きの文字を用いて最大n文字の水平線を描画
- ncurses_mvinch — 位置を移動し、新しい位置の属性付きの文字を取得する

- ncurses_mvvline — 位置を新しく設定し、属性付きの文字を用いて最大 n 文字の垂直線を描画する
- ncurses_mvwaddstr — ウインドウの新規位置に文字列を追加する
- ncurses_napms — スリープ
- ncurses_new_panel — 新しいパネルを作成し、それをウィンドウに関連づける
- ncurses_newpad — 新しいパッド (window) を作成する
- ncurses_newwin — 新規ウインドウを作成する
- ncurses_nl — 改行と復改/ラインフィードを変換する
- ncurses_nocbreak — 端末を cooked モードに変更する
- ncurses_noecho — キーボード入力エコーを無効にする
- ncurses_nonl — 改行と復改/ラインフィードを変換しない
- ncurses_noqiflush — シグナル文字のフラッシュを無効とする
- ncurses_noraw — 端末を raw モード以外に変更する
- ncurses_pair_content — 色の RGB 値を取得する
- ncurses_panel_above — パネルの上のパネルを返す
- ncurses_panel_below — パネルの下のパネルを返す
- ncurses_panel_window — パネルに関連付けられたウィンドウを返す
- ncurses_pnoutrefresh — パッドから仮想画面にリージョンをコピーする
- ncurses_prefresh — パッドから仮想画面にリージョンをコピーする
- ncurses_putp — パディング情報を文字列に適用し、それを出力する
- ncurses_qiflush — シグナル文字のフラッシュを有効とする
- ncurses_raw — 端末を raw モードに変更する
- ncurses_refresh — 画面をリフレッシュする
- ncurses_replace_panel — パネルに関連付けられたウィンドウを置き換える
- ncurses_reset_prog_mode — def_prog_mode で保存したプログラムモードをリセットする
- ncurses_reset_shell_mode — def_shell_mode で保存したシェルモードをリセットする
- ncurses_resetty — 保存した端末モードに復帰する
- ncurses_savetty — 端末の状態を保存する
- ncurses_scr_dump — 画面の内容をファイルにダンプする
- ncurses_scr_init — ファイルダンプから画面を初期化する
- ncurses_scr_restore — ファイルダンプから画面を復帰する
- ncurses_scr_set — ファイルダンプから画面を継承する
- ncurses_scrl — 現在位置を変更せずに画面の内容をスクロールアップまたはダウンする
- ncurses_show_panel — 不可視のパネルをスタックの最上部に置き、見えるようにする
- ncurses_slk_attr — 現在のソフトラベルキー属性を返す
- ncurses_slk_attroff — ソフトファンクションキーラベルの指定した属性を無効にする
- ncurses_slk_attron — ソフトファンクションキーラベルの指定した属性を有効にする
- ncurses_slk_attrset — ソフトファンクションキーラベルに、指定した属性を設定する
- ncurses_slk_clear — 画面からソフトラベルをクリアする
- ncurses_slk_color — ソフトラベルキーの色を設定する
- ncurses_slk_init — ソフトラベルキー関数を初期化する
- ncurses_slk_noutrefresh — 仮想画面にソフトラベルキーをコピーする
- ncurses_slk_refresh — ソフトラベルキーを画面にコピーする
- ncurses_slk_restore — ソフトラベルキーを復帰する
- ncurses_slk_set — ファンクションキーラベルを設定する
- ncurses_slk_touch — ncurses_slk_noutrefresh を実行する際に強制的に出力する
- ncurses_standend — 'standout' 属性の使用を停止する
- ncurses_standout — 'standout' 属性の使用を開始する
- ncurses_start_color — 色の使用を開始する
- ncurses_termattrs — 端末でサポートされる全ての属性フラグの論理和を返す
- ncurses_termname — 端末の(簡略)名を返す
- ncurses_timeout — 特別なキーシーケンスのタイムアウトを設定する
- ncurses_top_panel — 可視パネルをスタックの最上部に移動する
- ncurses_typeahead — typeahead 確認用に別のファイル記述子を指定する
- ncurses_ungetch — 入力ストリームに 1 文字戻す
- ncurses_ungetmouse — マウスイベントをキューにプッシュする
- ncurses_update_panels — 仮想画面を再描画し、スタック内のパネルとの関係を反映させる
- ncurses_use_default_colors — 端末のデフォルト色をカラー ID -1 に割り付ける
- ncurses_use_env — 端末の大きさに関する環境情報の使用を制御する
- ncurses_use_extended_names — terminfo 記述において拡張名の使用を制御する
- ncurses_vidattr — video attribute モードで、端末上に文字列を表示する
- ncurses_vline — 現在位置に最大 n 文字の属性付きの文字を用いて垂直線を描画する
- ncurses_waddch — ウィンドウ内の現在位置に文字を追加し、カーソルを進める
- ncurses_waddstr — ウィンドウ内の現在位置にテキストを出力する
- ncurses_wattroff — ウィンドウの属性をオフにする

- ncurses_wattron — ウィンドウの属性をオンにする
- ncurses_wattrset — ウィンドウの属性を設定する
- ncurses_wborder — 属性文字を使用してウィンドウの周囲に線を描画する
- ncurses_wclear — ウィンドウをクリアする
- ncurses_wcolor_set — ウィンドウの色の組み合わせを設定する
- ncurses_werase — ウィンドウを消去する
- ncurses_wgetch — キーボート（ウィンドウ）から文字を読み込む
- ncurses_whline — 指定した属性文字を用いて、最大 n 文字分の長さの水平線を ウィンドウに描画する
- ncurses_wmouse_trafo — ウィンドウ/標準画面の座標系を変換する
- ncurses_wmove — ウィンドウの出力位置を移動する
- ncurses_wnoutrefresh — ウィンドウを仮想画面にコピーする
- ncurses_wrefresh — 端末画面のウインドウをリフレッシュする
- ncurses_wstandend — ウィンドウの standout モードを終了する
- ncurses_wstandout — ウィンドウの standout モードに入る
- ncurses_wvline — 指定した属性文字を用いて、最大 n 文字分の長さの垂直線を ウィンドウに描画する

ネットワーク関数

導入

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

ネットワーク設定オプション(Network)

名前	デフォルト	変更の可否	変更の履歴
define_syslog_variables	"0"	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

define_syslog_variables boolean

様々なsyslog変数(例: $LOG_PID、$LOG_CRON、等)を定義するかどうか。 これをoffにするのは、性能面では良い考えです。 実行時に define_syslog_variables()をコールする ことにより、これらを定義することができます。

リソース型

この拡張モジュールでは、ファイルポインタリソースを定義しています。これは fsockopen() および pfsockopen() で使用するものです。

定義済み定数

以下の定数は、PHP コアに含まれており、常に利用可能です。

openlog()オプション

定数	説明
LOG_CONS	システムロガーにデータを送信する際にエラーが発生した場合、 システムコンソールに直接書き込む。
LOG_NDELAY	ロガーにただちに接続をオープンする。
LOG_ODELAY	(デフォルト) 最初のメッセージが記録されるまで接続のオープンを 遅延させる。
LOG_NOWAIT	
LOG_PERROR	標準エラー出力にもログメッセージを出力する
LOG_PID	各メッセージにプロセスIDを含める

openlog() ファシリティ

定数	説明
LOG_AUTH	セキュリティ/認証メッセージ（LOG_AUTHPRIVが定義されているシステムであれば 代わりにそれを使用してください）
LOG_AUTHPRIV	セキュリティ/認証メッセージ (private)

定数	説明
LOG_CRON	時刻デーモン (cron and at)
LOG_DAEMON	その他のシステムデーモン
LOG_KERN	カーネルメッセージ
LOG_LOCAL0 ... LOG_LOCAL7	ローカルで使用できるようリザーブされている。Windowsでは使用できない。
LOG_LPR	ラインプリンタサブシステム
LOG_MAIL	メールサブシステム
LOG_NEWS	USENETニュースサブシステム
LOG_SYSLOG	syslogdによって内部で生成されたメッセージ
LOG_USER	一般ユーザーレベルのメッセージ
LOG_UUCP	UUCPサブシステム

syslog()プロパティ(降順)

定数	説明
LOG_EMERG	システムは使用不可
LOG_ALERT	アクションを直ちに起こすことが必要
LOG_CRIT	危機的な条件
LOG_ERR	エラー条件
LOG_WARNING	警告条件
LOG_NOTICE	正常、しかし、注意すべき条件
LOG_INFO	情報メッセージ
LOG_DEBUG	デバッグレベルメッセージ

dns_get_record()オプション

定数	説明
DNS_A	IPv4アドレスリソース
DNS_MX	Mail Exchangerリソース
DNS_CNAME	エイリアス(Canonical Name)リソース
DNS_NS	Authoritative Name Serverリソース
DNS_PTR	ポインタリソース
DNS_HINFO	ホスト情報リソース（これらの値の意味については、IANA の » Operating System Names を参照ください）
DNS_SOA	認証リソースの開始
DNS_TXT	テキストリソース
DNS_ANY	全てのリソースレコード。多くのシステムでは、 これは、全てのリソースレコードを返します。 しかし、危機的な状況には対応できません。代わりに DNS_ALLを試してください。
DNS_AAAA	IPv6アドレスリソース
DNS_ALL	利用可能なレコード型毎のネームサーバへの反復クエリ。

checkdnsrr

(PHP 4, PHP 5)

checkdnsrr — 指定したインターネットホスト名もしくは IP アドレスに対応する DNS レコードを検索する

説明

int ***checkdnsrr*** *(string $host [, string $type])*

host に対応する type 型のレコードを DNS から探します。

パラメータ

host

host は、ドットで 4 つに分けられた形式の IP アドレスか、あるいはホスト名です。

type

type は、A, MX, NS, SOA, PTR, CNAME, AAAA, A6, SRV, NAPTR, ANY のどれか一つです。デフォルトは MX です。

返り値

レコードが見つかった場合に **TRUE**、 何も見つからないかエラーが発生した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.2.4	TXT *type* が追加されました。
5.0.0	AAAA *type* が追加されました。

注意

> **注意**: この関数は Windows プラットフォームでは実装されていません。 » PEAR の » Net_DNS クラスをお試しください。

参考

- dns_get_record()
- getmxrr()
- gethostbyaddr()
- gethostbyname()
- gethostbynamel()
- the named(8) manual page

---

closelog

(PHP 4, PHP 5)

closelog — システムログへの接続を閉じる

説明

bool **closelog** (void)

closelog() はシステムログへの書きこみに使用されているデスクプリタを閉じます。 **closelog()** の使用はオプションです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- define_syslog_variables()
- syslog()
- openlog()

---

debugger_off

(No version information available, might be only in CVS)

debugger_off — PHP の内部デバッガを無効にする (PHP 3)

説明

int **debugger_off** (void)

PHP の内部デバッガを無効にします。

返り値

成功した場合に 1、エラー時に 0 を返します。

注意

この関数は PHP 3 でのみ有効です。

参考

- PHP のデバッグ

---

debugger_on

(No version information available, might be only in CVS)

debugger_on — PHP の内部デバッガを有効にする (PHP 3)

説明

*int **debugger_on** (string $address)*

PHP の内部デバッガを有効にし、 address に接続します。

パラメータ

address

デバッガのホスト名。

返り値

成功した場合に 1、エラー時に 0 を返します。

注意

この関数は PHP 3 でのみ有効です。

参考

- *PHP のデバッグ*

---

define_syslog_variables

(PHP 4, PHP 5)

define_syslog_variables — syslog に関係する全ての定数を初期化する

説明

*void **define_syslog_variables** (void)*

syslog 関数で使用される全ての定数を初期化します。

返り値

値を返しません。

参考

- *openlog()*
- *syslog()*
- *closelog()*

---

dns_check_record

(PHP 5)

dns_check_record — checkdnsrr() のエイリアス

説明

この関数は次の関数のエイリアスです。 checkdnsrr().

---

dns_get_mx

(PHP 5)

dns_get_mx — getmxrr() のエイリアス

説明

この関数は次の関数のエイリアスです。 getmxrr().

---

dns_get_record

(PHP 5)

dns_get_record — ホスト名に関連する DNS リソースレコードを取得する

説明

array ***dns_get_record*** *(string $hostname [, int $type [, array &$authns]], array &$addtl)*

指定した hostname に関連づけられた DNS リソースレコードを取得します。

パラメータ

hostname

hostname は、正しい DNS ホスト名、すなわち "www.example.com" のようなものでなければなりません。 in-addr.arpa 形式の表記を用いた逆引き検索は可能ですが、 たいていは gethostbyaddr() を用いるほうが適当です。

> ***注意**: DNS の標準規格により、メールアドレスは user.host 形式で渡されます （たとえば、hostmaster.example.com が hostmaster@example.com の代わりとなります）。 この値をしっかり確認し、mail() のような関数で 利用する前には必要なら変更を加えることを忘れないようにしてください。*

type

*デフォルトでは、**dns_get_record()** は hostname に関連するすべてのリソースレコードを検索します。 これを制限するには、オプションの type パラメータを指定してください。 type は以下のうちのいずれかです。 **DNS_A**, **DNS_CNAME**, **DNS_HINFO**, **DNS_MX**, **DNS_NS**, **DNS_PTR**, **DNS_SOA**, **DNS_TXT**, **DNS_AAAA**, **DNS_SRV**, **DNS_NAPTR**, **DNS_A6**, **DNS_ALL** or **DNS_ANY**。 デフォルトは DNS_ANY です。*

> ***注意**: プラットフォーム依存の libresolv のおかしな挙動のせいで、 **DNS_ANY** が常にすべてのレコードを返すとは 限りません。速度は遅くなりますが、**DNS_ALL** のほうがより確実にすべてのレコードを取得できます。*

authns

参照で渡し、 Authoritative Name Servers のリソースレコードが格納されます。

addtl

参照で渡し、 Additional Records が格納されます。

返り値

この関数は、連想配列を要素にもつ配列を返します。それぞれの連想配列には 少なくとも 以下のキーが含まれています:

基本 DNS 属性

属性	意味
host	これ以降の関連するデータが参照する DNS 名。
class	**dns_get_record()** は Internet クラスのレコードのみを返すので、このパラメータは常に *IN* を返します。
type	レコード型を表す文字列。type の値に応じて、 結果の配列には追加の属性が含まれます。以下の表を参照ください。
ttl	このレコードの有効期限。レコードの本来の ttl と一致するとは *限りません*。むしろ、 ネームサーバへのクエリにかかった時間をそこから引いたものに一致します。

'type' に応じて連想配列に追加される項目

レコード型	追加項目
A	*ip*: ドット区切り 10 進数形式の IPv4 アドレス。
MX	*pri*: メールエクスチェンジャの優先度。 数字が小さいほど優先度が高い。 *target*: メールエクスチェンジャの FQDN 。 dns_get_mx() も参照ください。
CNAME	*target*: レコードのエイリアスの対象となっている場所の FQDN 。
NS	*target*: このホスト名に対する権威をもっているネームサーバの FQDN 。
PTR	*target*: レコードが指している、DNS 名前空間内の場所
TXT	*txt*: このレコードに関連付けられている任意の文字列。
HINFO	*cpu*: このレコードが参照しているマシンの CPU を識別する IANA 番号。 *os*: このレコードが参照しているマシン上の OS を識別する IANA 番号。 これらの値の意味については、IANA の » Operating System Names を参照ください。
SOA	*mname*: リソースレコードの元となるマシンの FQDN 。 *rname*: このドメインの管理責任者の Email アドレス。 *serial*: ドメインのシリアル番号。 *refresh*: セカンダリネームサーバがこのドメインのコピーを更新する際に参照するリフレッシュ間隔（秒単位）。 *retry*: リフレッシュが失敗した際に 2 度目のリフレッシュを試みるまでの間隔（秒単位） *expire*: セカンダリネームサーバが、ゾーンデータの リフレッシュに失敗した場合にコピーのデータを破棄せず持ち続ける期間 （秒単位）。 *minimum-ttl*: クライアントが、 一度取得したデータを再リクエストすることなしに利用できる最小期間（秒単位）。 個々のリソースレコードによって上書きが可能。
AAAA	*ipv6*: IPv6 アドレス。
A6(PHP >= 5.1.0)	*masklen*: *chain* で指定された対象から引き継ぐビット長。 *ipv6*: *chain* とマージするためのこのレコードのアドレス。 *chain*: *ipv6* データとマージするための親レコード。
SRV	*pri*: (Priority) 値が小さいものが優先されます。 *weight*: 同じ優先順位の *targets* からランダムに選択する際の重み。 *target* および *port*: リクエストされたサービスが存在するホスト名とポート。 詳細は » RFC 2782 を参照ください。
NAPTR	*order* および *pref*: 上の *pri* および *weight* と同じ。 *flags*, *services*, *regex*, および *replacement*: » RFC 2915 で定義されるパラメータ。

例

Example#1 dns_get_record() の使用

```php
<?php
$result = dns_get_record("php.net");
print_r($result);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => MX
            [pri] => 5
            [target] => pair2.php.net
            [class] => IN
            [ttl] => 6765
        )

    [1] => Array
        (
            [host] => php.net
            [type] => A
            [ip] => 64.246.30.37
            [class] => IN
            [ttl] => 8125
        )

)
```

Example#2 dns_get_record() と DNS_ANY の使用

MX レコードが解決されれば、たいていはメールサーバの IP アドレスを 取得したくなるものです。そのため、**dns_get_record()** は addtl に関連するレコードを含めて返します。 また、authns には 権威のあるネームサーバのリストを含めて返します。

```php
<?php
/* php.net の "ANY" レコードを要求し、
   それに付随する情報を格納した配列を
   作成する。
   $authns にはネームサーバの一覧が、
   また $addtl には追加レコードが
   それぞれ格納される。 */
$result = dns_get_record("php.net", DNS_ANY, $authns, $addtl);
echo "Result = ";
print_r($result);
echo "Auth NS = ";
print_r($authns);
echo "Additional = ";
print_r($addtl);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Result = Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => MX
            [pri] => 5
            [target] => pair2.php.net
            [class] => IN
            [ttl] => 6765
        )

    [1] => Array
        (
            [host] => php.net
            [type] => A
            [ip] => 64.246.30.37
            [class] => IN
            [ttl] => 8125
        )

)
Auth NS = Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => remote1.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [1] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => remote2.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [2] => Array
        (
            [host] => php.net
```

```
            [type] => NS
             [target] => ns1.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [3] => Array
        (
             [host] => php.net
            [type] => NS
             [target] => ns2.easydns.com
            [class] => IN
            [ttl] => 10722
        )

)
Additional  =  Array
(
    [0] => Array
        (
             [host] => pair2.php.net
            [type] => A
             [ip] => 216.92.131.5
            [class] => IN
            [ttl] => 6766
        )

    [1] => Array
        (
             [host] => remote1.easydns.com
            [type] => A
             [ip] => 64.39.29.212
            [class] => IN
            [ttl] => 100384
        )

    [2] => Array
        (
             [host] => remote2.easydns.com
            [type] => A
             [ip] => 212.100.224.80
            [class] => IN
            [ttl] => 81241
        )

    [3] => Array
        (
             [host] => ns1.easydns.com
            [type] => A
             [ip] => 216.220.40.243
            [class] => IN
            [ttl] => 81241
        )

    [4] => Array
        (
             [host] => ns2.easydns.com
            [type] => A
             [ip] => 216.220.40.244
            [class] => IN
            [ttl] => 81241
        )

)
```

注意

注意: この関数は Windows プラットフォームでは実装されていません。 また、(Mac を含む) *BSD システムでも (現時点では) 動作しません。 » PEAR の » Net_DNS クラスを試してみてください。

参考

- dns_get_mx()
- dns_check_record()

---

fsockopen

(PHP 4, PHP 5)

fsockopen — インターネット接続もしくはUnix ドメインソケット接続をオープンする

説明

resource **fsockopen** (string $hostname [, int $port [, int &$errno [, string &$errstr [, float $timeout]]]])

hostname で指定したリソースへのソケット接続を開始します。

サポートされるソケットトランスポートのリスト に記述されているように、PHP は Internet ドメインまたは Unix ドメインをサポートします。 サポートされるトランスポートのリストは、stream_get_transports() を使って取得することもできます。

ソケットはデフォルトでブロックモードで開かれます。 socket_set_blocking() を使用して、 非ブロックモードに切換えることができます。

パラメータ

hostname

OpenSSL サポートを有効にしてコンパイルした場合、 hostname の前に 'ssl://' または 'tls://' を付加することにより、TCP/IP 経由でリモートホストに接続する際に SSL または TLS クライアント接続を使用することができます。

port

ポート番号。

errno

指定した場合は、システムコール connect() で発生したエラーのエラー番号が格納されます。

errno は0 なのに関数が **FALSE** を返す場合、connect() をコールする前にエラーを発生したことを示します。 この場合、おそらくはソケットの初期化に原因があります。

errstr

エラーメッセージを表す文字列。

timeout

接続タイムアウト秒数。

> **注意**: ソケット経由でデータを読み書きする際のタイムアウトを設定する必要がある場合、 **fsockopen()** の timeout パラメータは、ソケットに接続する間にだけ適用されるため、 socket_set_timeout()を使用してください。

返り値

fsockopen()は、ファイルポインタを返します。 このファイルポインタは、 (fgets(), fgetss(), fputs(), fclose(), feof() のような) 他のファイル関数で使用可能です。 失敗した場合は **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	timeout パラメータが win32 もサポートするようになりました。
4.3.0	TCP/IP 上での SSL および TLS のサポートが追加されました。
4.0.0	UDP のサポートが追加されました。
3.0.9	timeout パラメータが追加されました。

例

Example#1 fsockopen() の例

```
<?php
$fp = fsockopen("www.example.com", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br />¥n";
} else {
    $out = "GET / HTTP/1.1¥r¥n";
    $out .= "Host: www.example.com¥r¥n";
    $out .= "Connection: Close¥r¥n¥r¥n";

    fwrite($fp, $out);
    while (!feof($fp)) {
        echo fgets($fp, 128);
    }
    fclose($fp);
}
?>
```

Example#2 UDP 接続の使用法

以下の例は、自分のマシンの UDP サービス "daytime" (ポート13) から日付と時間を取得する方法を示すものです。

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, $errno, $errstr);
if (!$fp) {
    echo "ERROR: $errno - $errstr<br />¥n";
} else {
    fwrite($fp, "¥n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

注意

> **注意**: 環境によっては Unix ドメインまたは オプションの接続タイムアウトが利用できないこともあります。

警告

UDPソケットは、リモートホストとの接続が確立されていない場合でも、 エラーを発生せずにオープンされたように見えることが時々あります。 このエラーは、そのソケットでデータを読み書きした際にのみ明らかになります。 この原因は、UDPが"コネクションレス"のプロトコルであり、 実際にデータを送受信する必要が生じるまで、 オペレーションシステムがソケット用のリンクを確立しようとしないためです。

> **注意**: 数値で IPv6 アドレスを指定するときは、 (例 fe80::1) アドレスを角カッコでくくらなくてはなりません。たとえば、tcp://[fe80::1]:80.

参考

- pfsockopen()
- stream_set_blocking()
- stream_set_timeout()
- fgets()
- fgetss()
- fwrite()
- fclose()
- feof()
- Curl 拡張モジュール

---

gethostbyaddr

(PHP 4, PHP 5)

gethostbyaddr — 指定した IP アドレスに対応するインターネットホスト名を取得する

説明

string **gethostbyaddr** (string $ip_address)

ip_address で指定したインターネットホストのホスト名を返します。

パラメータ

ip_address

ホストの IP アドレス。

返り値

ホスト名を返します。 失敗した場合は、そのままの形の ip_address を文字列で返します。

例

Example#1 単純な gethostbyaddr() の例

```
<?php
$hostname = gethostbyaddr($_SERVER['REMOTE_ADDR']);

echo $hostname;
?>
```

参考

- gethostbyname()
- gethostbynamel()

---

gethostbyname

(PHP 4, PHP 5)

gethostbyname — インターネットホスト名に対応するIPアドレスを取得する

説明

string **gethostbyname** (string $hostname)

hostname で指定したインターネットホストの IP アドレスを返します。

パラメータ

hostname

ホスト名。

返り値

IP アドレスを返します。失敗した場合は、そのままの形の hostname を文字列で返します。

例

Example#1 単純な gethostbyname() の例

```
<?php
$ip = gethostbyname('www.example.com');

echo $ip;
?>
```

参考

- gethostbyaddr()
- gethostbynamel()

---

gethostbynamel

(PHP 4, PHP 5)

gethostbynamel — 指定したインターネットホスト名に対応するIPアドレスのリストを取得する

説明

array **gethostbynamel** (string $hostname)

hostname で指定したインターネットホストを検索して得られた IP アドレスのリストを返します。

パラメータ

hostname

ホスト名。

返り値

IP アドレスの配列を返します。 もし hostname が解決できなかった場合は **FALSE** を返します。

例

Example#1 gethostbynamel() の例

```
<?php
$hosts = gethostbynamel('www.example.com');
print_r($hosts);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => 192.0.34.166
)
```

参考

- gethostbyname()
- gethostbyaddr()
- checkdnsrr()
- getmxrr()
- the named(8) manual page

---

getmxrr

(PHP 4, PHP 5)

getmxrr — 指定したインターネットホスト名に対応する MX レコードを取得する

説明

bool **getmxrr** (string $hostname , array &$mxhosts [, array &$weight])

hostname に対応する MX レコードを DNS から探します。

パラメータ

hostname

インターネットホスト名。

mxhosts

見つかった MX レコードのリストが、配列 mxhosts に格納されます。

weight

配列 weight を指定すると、 そこに重み情報が格納されます。

返り値

何かレコードが見つかった場合に **TRUE**、 何も見つからないかエラーが発生した場合に **FALSE** を返します。

注意

注意: この関数をメールアドレスの確認の目的で使用すべきではありません。 DNS が検出したメールエクスチェンジャーを返すだけです。 しかし、» RFC 2821 によれば、 メールエクスチェンジャーがひとつも見つからなければ、 hostname 自体が唯一のメールエクスチェンジャー であるとみなされ、その優先度は 0（最高）となります。

注意: この関数は Windows では実装されていません。 » PEAR の » Net_DNS クラスを試してみてください。

参考

- checkdnsrr()
- dns_get_record()
- gethostbyname()
- gethostbynamel()
- gethostbyaddr()
- named(8) のマニュアルページ

---

getprotobyname

(PHP 4, PHP 5)

getprotobyname — プロトコル名のプロトコル番号を得る

説明

int **getprotobyname** (string $name)

getprotobyname()は、プロトコル name のプロトコル番号を /etc/protocols から取得して返します。

パラメータ

name

プロトコル名。

返り値

プロトコル番号、あるいはプロトコルが見つからない場合に -1 を返します。

例

Example#1 getprotobyname() の例

```
<?php
$protocol = 'tcp';
$get_prot = getprotobyname($protocol);
if ($get_prot == -1) {
    echo '無効なプロトコル';
} else {
    echo 'プロトコル番号: ' . $get_prot;
}
?>
```

参考

- getprotobynumber()

---

getprotobynumber

(PHP 4, PHP 5)

getprotobynumber — プロトコル番号が指すプロトコル名を取得する

説明

string **getprotobynumber** (int $number)

getprotobynumber() は、 /etc/protocols に基づき プロトコル番号 number が指すプロトコル名を返します。

パラメータ

number

プロトコル番号。

返り値

プロトコル名を文字列で返します。

参考

- getprotobyname()

---

getservbyname

(PHP 4, PHP 5)

getservbyname — インターネットサービスおよびプロトコルが関連するポート番号を取得する

説明

int **getservbyname** (string $service , string $protocol)

getservbyname() は、 /etc/services に基づき 指定したプロトコル protocol に関して service に対応するインターネットポートを返します。

パラメータ

service

インターネットサービス名を表す文字列。

protocol

protocol は "tcp" あるいは "udp" (小文字) のいずれかです。

返り値

ポート番号を返します。 service あるいは protocol が見つからない場合は **FALSE** を返します。

例

Example#1 getservbyname() の例

```
<?php
$services = array('http', 'ftp', 'ssh', 'telnet', 'imap',
'smtp', 'nicname', 'gopher', 'finger', 'pop3', 'www');

foreach ($services as $service) {
    $port = getservbyname($service, 'tcp');
    echo $service . ": " . $port . "<br />¥n";
}
?>
```

参考

- getservbyport()
- ポート番号の完全なリストは » http://www.iana.org/assignments/port-numbers

---

getservbyport

(PHP 4, PHP 5)

getservbyport — ポートおよびプロトコルに対応するインターネットサービスを得る

説明

string **getservbyport** (int $port , string $protocol)

getservbyport() は、 /etc/services に基づき 指定したプロトコル protocol に関してポート port に関連するインターネットサービスを返します。

パラメータ

port

ポート番号。

protocol

protocol は "tcp" あるいは "udp" (小文字) のいずれかです。

返り値

インターネットサービス名を文字列で返します。

参考

- getservbyname()

---

header

(PHP 4, PHP 5)

header — 生の HTTP ヘッダを送信する

説明

void **header** (string $string [, bool $replace [, int $http_response_code]])

header() は、生の HTTP ヘッダを送信するために使用されます。 HTTP ヘッダについての詳細な情報は » HTTP/1.1 仕様 を参照ください。

覚えておいて頂きたいのは、**header()** 関数は、 通常の HTML タグまたは PHP からの出力にかかわらず、すべての実際の 出力の前にコールする必要があることです。 頻出するエラーとして、include() または require() 関数、他のファイルをアクセスする関数に 空白または空行があり、**header()** の前に出力が 行われてしまうというものがあります。同じ問題は、単一の PHP/HTML ファイルを使用している場合でも存在します。

```
<html>
<?php
/* これはエラーとなります。この上に出力があることに注目してください。
 * それはheader()のコールより前であるということになります */
header('Location: http://www.example.com/');
?>
```

パラメータ

string

ヘッダ文字列。

特殊な header コールが 2 種類あります。最初のものは、 文字列 "HTTP/" から始まる全てのヘッダ（大文字・小文字は区別されません）です。このヘッダは、送信する HTTP ステータスコードを示すために使用されます。 例えば、存在しないファイルへのリクエストを処理するためにある PHP スクリプトを使用するよう（ErrorDocument ディレクティブにより）Apache を設定する場合、 そのスクリプトが正しいステータスコードを返すようにする必要があります。

```
<?php
header("HTTP/1.0 404 Not Found");
?>
```

2 番目の特別なヘッダは、"Location:" ヘッダです。このヘッダがブラウザに返されるだけではなく、 ブラウザに REDIRECT (302) ステータスコードを返します（3xx ステータスコードが既に送信されていない場合にのみ）。

```
<?php
header("Location: http://www.example.com/"); /* ブラウザをリダイレクトします */

/* リダイレクトしたら、これ以降のコードは実行されません */
exit;
?>
```

replace

オプションのパラメータ replace は、ヘッダが 前に送信された類似のヘッダを置換するか、または、同じ形式の二番目の ヘッダを追加するかどうかを指定します。デフォルトでは、この関数は 置換を行ないますが、二番目の引数に **FALSE** を指定すると、同じ型の 複数のヘッダを強制的に生成します。例えば、

```
<?php
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM', false);
?>
```

http_response_code

HTTP レスポンスコードを強制的に指定の値にします。

返り値

値を返しません。

変更履歴

バージョン	説明
4.4.2 および 5.1.2	この関数は一度に複数のヘッダを送信できないようになりました。 これは、ヘッダインジェクション攻撃への対策です。
4.3.0	http_response_code パラメータが追加されました。
4.0.4	replace パラメータが追加されました。

例

Example#1 ダウンロードダイアログ

PDF ファイルを生成した場合など、 それをダウンロードするかの確認ダイアログを表示させたいことがあるでしょう。 そんな場合は、» Content-Disposition ヘッダを使用してファイル名を指定すると、ブラウザ側でダイアログを表示させることができます。

```
<?php
// PDFを出力します
header('Content-type: application/pdf');

// downloaded.pdf という名前で保存させます
header('Content-Disposition: attachment; filename="downloaded.pdf"');

// もとの PDF ソースは original.pdf です
readfile('original.pdf');
?>
```

Example#2 キャッシュディレクティブ

PHP スクリプトはしばしば動的に HTML を生成するため、クライアント ブラウザやサーバおよびクライアントブラウザの間でプロキシがキャッシュを行ったりするべきではありません。多くのプロキシとクライアントでは、 以下のコードにより強制的にキャッシュを無効にできます。

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // 過去の日付
?>
```

> **注意**: 上記のヘッダを全て出力しなかったとしてもページのキャッシュが 行われない場合があることに気付くかもしれません。デフォルトの ブラウザのキャッシュの動作をユーザが変更できる手段はいくつもあります。 上記のヘッダを送信することにより、スクリプトの出力がキャッシュされる 可能性がある設定を上書きするべきです。
> さらに、session_cache_limiter() および 設定 session.cache_limiter を用いると、 セッションが使用された際にキャッシュ関係の正しいヘッダを 自動的に生成させることもできます。

注意

> **注意**: PHP 4 では、出力のバッファリングを使用してこの問題を回避することができます。 この場合、ブラウザへの出力が送信するまでサーバに 全てバッファリングされるオーバーヘッドがあります。出力バッファリングは、 ob_start() と ob_end_flush() をスクリプトでコールするか php.ini またはサーバ設定ファイルの設定ディレクティブ output_buffering を設定することにより 行うことが可能です。
>
> **注意**: 実際に **header()** が最初にコールされたか どうかにかかわらず、HTTP ステータスヘッダ行は クライアントに対し常に最初に送信されます。 HTTP ヘッダが既に送信されていない限り、 **header()** をコールすることでステータスは 常に上書きされます。
>
> **注意**: Microsoft Internet Explorer 4.01 にはバグがあり、 この方法が動作しません。これを解決する方法はありません。 同様に Microsoft Internet Explorer 5.5 にもこれを妨げるバグがあります。 こちらは、サービスパック 2 以降にアップグレードすることで対応できます。
>
> **注意**: セーフモードが有効な場合は、 WWW-Authenticate ヘッダ (HTTP 認証に使用する) を設定した際に、スクリプトの uid が realm 部に追加されます。
>
> **注意**: HTTP/1.1 では、スキーム、ホスト名、絶対パスを含む絶対 URI が » Location: の引数として必要ですが、相対 URI を受け付けるクライアントもあります。 通常は、相対 URI から絶対 URI を作成するためには $_SERVER['HTTP_HOST']、$_SERVER['PHP_SELF'] および dirname() を使用できます。
>
> ```
> <?php
> /* カレントディレクトリの別のページにリダイレクトします */
> $host  = $_SERVER['HTTP_HOST'];
> $uri   = rtrim(dirname($_SERVER['PHP_SELF']), '/¥¥');
> $extra = 'mypage.php';
> header("Location: http://$host$uri/$extra");
> exit;
> ?>
> ```
>
> **注意**: session.use_trans_sid が有効であったとしても、セッション ID が Location ヘッダとともに 渡されることはありません。SID 定数を使用して 手動で渡す必要があります。

参考

- headers_sent()
- setcookie()
- HTTP 認証

---

headers_list

(PHP 5)

headers_list — 送信した (もしくは送信される予定の) レスポンスヘッダの一覧を返す

説明

array **headers_list** (void)

headers_list() はブラウザもしくはクライアントに送信されるヘッダの数値配列を返します。 これらのヘッダが送信されたかどうかを判断するためには headers_sent() を使用します。

返り値

ヘッダを、数値添字の配列で返します。

例

Example#1 headers_list() の使用例

```
<?php

/* setcookie() でレスポンスヘッダをそれ自身に追加します */
setcookie('foo', 'bar');

/* 独自のレスポンスヘッダを定義します。
   これはほとんどのクライアントで無視されます */
header("X-Sample-Test: foo");

/* レスポンスがプレーンテキストだと宣言します */
header('Content-type: text/plain');

/* 送信しようとしているヘッダは? */
var_dump(headers_list());
```

```
?>
```

上の例の出力は以下となります。

```
array(4) {
  [0]=>
  string(23) "X-Powered-By: PHP/5.1.3"
  [1]=>
  string(19) "Set-Cookie: foo=bar"
  [2]=>
  string(18) "X-Sample-Test: foo"
  [3]=>
  string(24) "Content-type: text/plain"
}
```

参考

- headers_sent()
- header()
- setcookie()

---

headers_sent

(PHP 4, PHP 5)

headers_sent — ヘッダが既に送信されているかどうかを調べる

説明

```
bool headers_sent ([ string &$file [, int &$line ]] )
```

ヘッダがすでに送信されているかどうかを調べます。

ヘッダブロックがいったん送信されてしまった後で header() 関数を使って新たなヘッダ行を送信することはできません。 この関数を使うには、少なくとも HTTP ヘッダ関連のエラーを予防する必要があります。 あるいは、出力バッファリング を使う方法もあります。

パラメータ

file

オプション引数の file と line がセットされている場合、 PHP のソースファイル名と出力が開始された行番号が、それぞれ file と line に格納されます。

line

出力を開始した行番号。

返り値

headers_sent() は、HTTP ヘッダがまだ送信されていない場合に **FALSE**、 そうでないでない場合に **TRUE** を返します。

変更履歴

バージョン	説明
4.3.0	オプションのパラメータ file と line が追加されました。

例

Example#1 headers_sent() 関数の使用例

```
<?php

// ヘッダがまだ何も送信されていない場合に、送信します
if (!headers_sent()) {
    header('Location: http://www.example.com/');
    exit;
}

// オプションのfileとlineパラメータの使用例（PHP4.3.0以降）
// $filename と $linenum が後で使用されていることに注目。
// これらの変数に事前に値を与えたりしてはいけません。
if (!headers_sent($filename, $linenum)) {
    header('Location: http://www.example.com/');
    exit;

// おそらく、ここでエラー処理を行うでしょう。
} else {

    echo "$filename の $linenum 行目でヘッダがすでに送信されています。\n" .
         "リダイレクトできません。代わりにこの <a " .
         "href=\"http://www.example.com\">リンク</a> をクリックしてください。\n";
    exit;
}
```

```
?>
```

参考

- ob_start()
- trigger_error()
- 関連する詳細な情報については header()

---

inet_ntop

(PHP 5 >= 5.1.0)

inet_ntop — パックされたインターネットアドレスを、人間が読める形式に変換する

説明

string **inet_ntop** (string $in_addr)

この関数は 32 ビット IPv4 形式あるいは 128 ビット IPv6 形式 (PHP が IPv6 サポートを有効にしてビルドされている場合) のアドレスを文字列表現のアドレスに変換します。

パラメータ

in_addr

32 ビット IPv4、あるいは 128b ビット IPv6 形式のアドレス。

返り値

アドレスを文字列で表したもの、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 inet_ntop() の例

```
<?php
$packed = chr(127) . chr(0) . chr(0) . chr(1);
$expanded = inet_ntop($packed);

/* 出力: 127.0.0.1 */
echo $expanded;

$packed = str_repeat(chr(0), 15) . chr(1);
$expanded = inet_ntop($packed);

/* 出力: ::1 */
echo $expanded;
?>
```

注意

> **注意**: この関数は Windows 環境にはまだ実装されていません。

参考

- long2ip()
- ip2long()
- inet_pton()

---

inet_pton

(PHP 5 >= 5.1.0)

inet_pton — 人間が読める形式の IP アドレスを、パックされた in_addr 形式に変換する

説明

string **inet_pton** (string $address)

この関数は、人間が読める形式の IPv4 あるいは IPv6 (PHP が IPv6 サポートを有効にしてビルドされている場合) のアドレスを 32 ビットあるいは 128 ビットのバイナリ形式に変換します。

パラメータ

address

可読形式の IPv4 アドレスあるいは IPv6 アドレス。

返り値

指定した address を in_addr 形式で表したものを返します。

例

Example#1 inet_pton() の例

```
<?php
 $in_addr = inet_pton('127.0.0.1');

 $in6_addr = inet_pton('::1');

 ?>
```

注意

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- ip2long()
- long2ip()
- inet_ntop()

---

ip2long

(PHP 4, PHP 5)

ip2long — (IPv4) インターネットプロトコルドット表記のアドレスを、適当なアドレスを有する文字列に変換する

説明

int **ip2long** (string $ip_address)

関数 **ip2long()** は、インターネット標準形式（ドット表記の文字列）表現から IPv4 インターネットネットアドレスを生成します。

ip2long() は不完全な形式の IP アドレスに対しても動作します。詳しい情報は » http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/libs/commtrf2/inet_addr.htm を参照ください。

パラメータ

ip_address

標準形式のアドレス。

返り値

IPv4 アドレス、あるいは ip_address が不正な形式の場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	このバージョンより前は、ip2long() が失敗したときの返り値は -1 でした。

例

Example#1 ip2long() の例

```
<?php
$ip = gethostbyname('www.example.com');
$out = "以下の URL は同じ意味です:<br />\n";
$out .= 'http://www.example.com/, http://' . $ip . '/, そして http://' . sprintf("%u", ip2long($ip)) . "/<br />¥n";
echo $out;
?>
```

Example#2 IP アドレスの表示

2 番目の例は、printf() 関数で変換されたアドレスを出力する方法を示すものです。 PHP 4 と PHP 5 のどちらでも使えます。

```
<?php
$ip    = gethostbyname('www.example.com');
$long  = ip2long($ip);

if ($long == -1 || $long === FALSE) {
    echo '無効な IP です。もういちど確認してください。';
} else {
    echo $ip    . "¥n";           // 192.0.34.166
    echo $long . "¥n";            // -1073732954
    printf("%u¥n", ip2long($ip)); // 3221234342
}
?>
```

Example#3 IP の検証

ip2long() を、それ単体で IP の検証に利用するべきではありません。long2ip() と組み合わせて利用します。

```
<?php
// IP が有効であることを確認します。また、不完全な形式の IP を
// 以下で示すような正しい形式（ドットで 4 つに区切られている）に変換します。
$ip = long2ip(ip2long("127.0.0.1")); // "127.0.0.1"
```

```
$ip = long2ip(ip2long("10.0.0"));  //  "10.0.0.0"
$ip = long2ip(ip2long("10.0.256")); //  "10.0.1.0"
?>
```

注意

注意: PHP の整数型は符号付き形式であり、多くの IP アドレスは負の整数値になります。 そのため、符号なし IP アドレスを文字列形式で取得するには sprintf() や printf() で "%u" を使用する必要があります。

注意: PHP 5 <= 5.0.2 では、IP 255.255.255.255 に対して **ip2long()** が **FALSE** を返します。 これは PHP 5.0.3 で -1 を返すように修正されています (PHP 4 と同じ動作です)。

参考

- long2ip()
- sprintf()

---

long2ip

(PHP 4, PHP 5)

long2ip — (IPv4) インターネットアドレスをインターネット標準ドット表記に変換する

説明

string **long2ip** (int $proper_address)

関数**long2ip()** は、適切なアドレス表現からドット表記 (例:aaa.bbb.ccc.ddd)のインターネットアドレスを生成します。

パラメータ

proper_address

正しい形式のアドレス。

返り値

インターネットの IP アドレスを表す文字列を返します。

参考

- ip2long()

---

openlog

(PHP 4, PHP 5)

openlog — システムのロガーへの接続をオープンする

説明

bool **openlog** (string $ident , int $option , int $facility)

openlog() は、プログラムによるシステムロガーへの接続をオープンします。

openlog() の使用は必須ではありません。この関数は、 必要に応じて syslog() により自動的に呼び出されます。 この場合、ident のデフォルト値は **FALSE** となります。

パラメータ

ident

文字列 ident が、各メッセージに追加されます。

option

option 引数は、 ログメッセージの生成時に使用されるロギング用オプションを指定します。

openlog() のオプション

定数	説明
LOG_CONS	システムロガーにデータが送信される間にエラーが発生した場合、 直接、システムコンソールに書き込まれます。
LOG_NDELAY	直ちにロガーへの接続をオープンします。
LOG_ODELAY	(デフォルト) 最初のメッセージがロギングされるまで接続のオープンを遅延します。
LOG_PERROR	標準エラー出力にもログメッセージを出力します。
LOG_PID	各メッセージに PID も含めます。

このオプションを一つまたは複数設定することが可能です。 複数のオプションを使用する場合、その論理和を指定します。 つまり、直ちに接続をオープンし、コンソールに書き込み、 各メッセージに PID を含めるには、 LOG_CONS | LOG_NDELAY | LOG_PID とします。

facility

引数 facility には、 ロギングを行う際のメッセージ型を指定します。これにより、 (使用するシステムの syslog の設定に関して) 異なった facility を有するメッセージをどのように処理するかを指定できるようになります。

openlog() の機能

定数	説明
LOG_AUTH	セキュリティ/認証用メッセージ（定数 LOG_AUTHPRIV が定義されているシステムでは、代わりにそれを使用してください）
LOG_AUTHPRIV	セキュリティ/認証 メッセージ（プライベート）
LOG_CRON	クロックデーモン（cron や at）
LOG_DAEMON	他のシステムデーモン
LOG_KERN	カーネルメッセージ
LOG_LOCAL0 ... LOG_LOCAL7	ローカルでの使用のために確保されているもので、Windows では使用できません
LOG_LPR	ラインプリンタサブシステム
LOG_MAIL	メールサブシステム
LOG_NEWS	USENET ニュース サブシステム
LOG_SYSLOG	syslogd で内部的に生成されたメッセージ
LOG_USER	一般的なユーザレベルのメッセージ
LOG_UUCP	UUCP サブシステム

注意: Windows 環境で使用できるのは LOG_USER だけです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- define_syslog_variables()
- syslog()
- closelog()

---

pfsockopen

(PHP 4, PHP 5)

pfsockopen — 持続的な Internet または Unix ドメインソケット接続をオープンする

説明

resource **pfsockopen** (string $hostname [, int $port [, int &$errno [, string &$errstr [, float $timeout]]]])

この関数は、fsockopen() と全く同様に動作します。 ただし、この関数による接続はリクエストが終了した後でも閉じられないという違いがあります。 この関数は、fsockopen() の持続的接続版です。

参考

- fsockopen()

---

setcookie

(PHP 4, PHP 5)

setcookie — クッキーを送信する

説明

bool **setcookie** (string $name [, string $value [, int $expire [, string $path [, string $domain [, bool $secure [, bool $httponly]]]]]])

setcookie() は、その他のヘッダ情報と共に 送信するクッキーを定義します。 ほかのヘッダ情報と同様に、 クッキーは、スクリプトによる他のあらゆる出力よりも前に 送信される必要があります（これはHTTPプロトコルの制約です）。 <html> や <head> タグはもちろん 空白も含め、あらゆる出力よりも前にこの関数をコールするようにしなければなりません。

一度クッキーが送信されると、次のページのロードからは $_COOKIE や $HTTP_COOKIE_VARS 配列によってクッキーにアクセスできます。 $_COOKIE のような スーパーグローバル は PHP 4.1.0 以降で有効となることに注意してください。 $HTTP_COOKIE_VARS は PHP 3 以降で使用できます。 クッキーの値は $_REQUEST 配列からもアクセスできます。

パラメータ

name 以外の全ての引数はオプションです。 全ての引数に関して引数の指定をスキップするために空文字列 ("") とすることが可能です。 expire および secure は数値なので、空文字列でスキップすることはできません。代わりにゼロ (0) を使用してください。

setcookie() 関数の各引数が どのように作用するかを知るには » Netscape cookie specification を参照ください。

name

クッキーの名前。

value

クッキーの値。この値はクライアントのコンピュータに保存されますので、 重要な情報は格納しないでください。 name が 'cookiename' だとする

と、 その値は $_COOKIE['cookiename'] で取得することができます。

expire

クッキーの有効期限。これは Unix タイムスタンプなので Epoch (1970 年 1 月 1 日) からの経過秒数となります。 time() または mktime() 関数により 返された現在のUNIX標準時に、期限としたい必要な秒数を加算したものを 利用することができるでしょう。 time()+60*60*24*30 は クッキーの有効期限を 30 日後にセットします。 0 を設定したり省略したりした場合は、クッキーはセッションの最後 (つまりブラウザを閉じるとき) が有効期限となります。

> **注意**: expire パラメータには、Wdy, DD-Mon-YYYY HH:MM:SS GMT といった形式ではなく Unix タイムスタンプを渡していることにお気づきでしょうか。 これは、PHP の内部で自動的に変換を行っているからです。
> expire は、クライアントの時刻と比較されます。 サーバの時刻との比較ではありません。

path

サーバ上での、クッキーを有効としたいパス '/' をセットすると、クッキーは domain 配下の全てで有効となります。 '/foo/' をセットすると、クッキーは /foo/ ディレクトリとそのサブディレクトリ配下 (例えば /foo/bar/) で有効となります。 デフォルト値は、クッキーがセットされたときのカレントディレクトリです。

domain

クッキーが有効なドメイン。 exapmle.com の全てのサブドメインでクッキーを有効とするには '.example.com' をセットします。 . は必須ではありませんが、 多くのブラウザにおいて互換性があります。 www.example.com にセットすると、 クッキーはwww サブドメインにおいてのみ有効となります。 末尾のマッチングについての詳細は » 仕様 をご覧ください。

secure

クライアントからのセキュアな HTTPS 接続の場合にのみクッキーが送信されるようにします。 **TRUE** を設定すると、セキュアな接続が存在する場合にのみクッキーを 設定します。デフォルトは **FALSE** です。 サーバ側では、このようにセキュアな接続の場合にのみクッキーを送信するという処理はプログラマの責任で行うことになります (たとえば $_SERVER["HTTPS"] の内容を使用します)。

httponly

TRUE を設定すると、HTTP を通してのみクッキーにアクセスできるようになります。 つまり、JavaScript のようなスクリプト言語からはアクセスできなくなるということです。 この設定を使用すると、XSS 攻撃によって ID を盗まれる危険性を減らせます (が、すべてのブラウザがこの設定をサポートしているというわけではありません)。 PHP 5.2.0 で追加されました。 **TRUE** あるいは **FALSE** で指定します。

返り値

もしもこの関数をコールする前に何らかの出力がある場合には、 **setcookie()** は失敗し **FALSE** を返します。 **setcookie()** が正常に実行されると、**TRUE** を返します。 この関数では、ユーザがクッキーを受け入れたかどうかを判断することはできません。

例

以下はクッキーを送信する例です。

Example#1 setcookie() での送信の例

```php
<?php
$value = 'something from somewhere';

setcookie("TestCookie", $value);
setcookie("TestCookie", $value, time()+3600);  /* 有効期限は一時間です */
setcookie("TestCookie", $value, time()+3600, "/~rasmus/", ".example.com", 1);
?>
```

クッキーの value の部分は、クッキーの送信を行う際に自動的に URL エンコードされ、またクッキーを受信した際は、自動的にデコード されてクッキー名と同じ名前の変数に格納されることに注意してください。 この挙動が気に入らない場合、もし PHP 5 を使用しているなら 代わりに setrawcookie() を使ってください。 スクリプト内部で TestCookie の内容を見たい場合は、 以下のいずれかの例を使用します。

```php
<?php
// 個々のクッキーを表示します
echo $_COOKIE["TestCookie"];
echo $HTTP_COOKIE_VARS["TestCookie"];

// デバッグ／テスト用には、全てのクッキーを見る方法があります。
print_r($_COOKIE);
?>
```

Example#2 setcookie() による削除の例

クッキーを削除する場合には、ブラウザの削除機構を起動するために 必ず有効期限を過去に設定する必要があります。 次に、先ほどの例で送信したクッキーを削除する例を示します。

```php
<?php
// 有効期限を一時間前に設定します
setcookie ("TestCookie", "", time() - 3600);
setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".example.com", 1);
?>
```

Example#3 setcookie() と配列

クッキー名で配列を記述することにより、クッキーの配列を設定することも可能です。 これにより配列要素と同数のクッキーを設定されますが、 クッキーがスクリプトに受信された際に、 値はクッキー名を有する配列に置きかえられます。

```php
<?php
// クッキーを設定します
setcookie("cookie[three]", "cookiethree");
setcookie("cookie[two]", "cookietwo");
setcookie("cookie[one]", "cookieone");

// ページを再読み込みした後に、表示します
if (isset($_COOKIE['cookie'])) {
    foreach ($_COOKIE['cookie'] as $name => $value) {
        echo "$name : $value <br />¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
three  :  cookiethree
two  :  cookietwo
one  :  cookieone
```

注意

> **注意**: PHP 4 では、この関数をコールする前でも出力できるように、 スクリプトの全ての出力をサーバ内にバッファリングさせることができます。 そのためには、ob_start() や ob_end_flush() を使用するか、あるいは php.ini やサーバ設定ファイルの output_buffering を使用します。
>
> **注意**: PHPの register_globals ディレクティブが on になっている場合、 クッキーは変数にも登録されています。 以下の例では、$TestCookie 変数が存在します。 $_COOKIE の使用することを推奨します。

陥りやすい失敗

- クッキーは、クッキーを有効にするために次にページをロードするまで アクセスすることができません。クッキーが正常にセットされたか テストするために、クッキーの有効期限が切れる前に次のページを ロードしてクッキーをチェックしてください。 有効期限は expire 引数でセットされます。 クッキーの利用についてデバッグするのに良い方法は print_r($_COOKIE); をコールすることです。
- クッキーは設定されたものと同じパラメータで削除する必要があります。 値が空文字列あるいは **FALSE** で、その他の全ての引数が前に setcookie をコールした時と同じである場合に、指定された名前のクッキーが リモートクライアント上から削除されます。
- クッキーの値として **FALSE** を設定すると、クッキーを削除しようとします。 そのため、boolean 値は使用できません。その代わりとして、**FALSE** ではなく 0、そして **TRUE** ではなく 1 を使用します。
- クッキー名で配列を記述することにより、 クッキーの配列を設定することも可能ですが、複数のクッキー がユーザーのシステム上に保存されることになります。 explode() を使用して ひとつのクッキー上に複数の名前と値をセットすることも 考慮してください。serialize() の使用はセキュリティーホールになり得るため、 この目的のために使用することはお勧めしません。

PHP 3 において同じスクリプトで **setcookie()** を複数回コールした場合、逆の順番で実行されます。 他のクッキーを挿入する前にあるクッキーを削除しようとする場合、 削除する前に挿入を行う必要があります。PHP 4 では、 **setcookie()** を複数回コールした場合でもコールした順番で実行されます。

> **注意**: サービスパック 1 を適用した Microsoft Internet Explorer 4 は、 パスに関するパラメータを設定したクッキーを正確に処理することができません。 Netscape Communicator 4.05 および Microsoft Internet Explorer 3.x は、path と time が設定されていない場合、 クッキーを正確に処理することができないようです。

参考

- header()
- setrawcookie()
- クッキー
- » RFC 2109
- » RFC 2965

---

setrawcookie

(PHP 5)

setrawcookie — 値を URL エンコードせずにクッキーを送信する

説明

```
bool setrawcookie ( string $name [, string $value [, int $expire [, string $path [, string $domain [, bool $secure [, bool $httponly ]]]]] ] )
```

setrawcookie() は、 ブラウザに送信される際クッキーの値が自動的に URL エンコードされないことを除き、 setcookie() と等価です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.2.0	httponly パラメータが追加されました。

参考

- setcookie()

---

socket_get_status

(PHP 4, PHP 5)

socket_get_status — stream_get_meta_data() のエイリアス

説明

この関数は次の関数のエイリアスです。 stream_get_meta_data().

---

socket_set_blocking

(PHP 4, PHP 5)

socket_set_blocking — stream_set_blocking() のエイリアス

説明

この関数は次の関数のエイリアスです。 stream_set_blocking().

---

socket_set_timeout

(PHP 4, PHP 5)

socket_set_timeout — stream_set_timeout() のエイリアス

説明

この関数は次の関数のエイリアスです。 stream_set_timeout().

---

syslog

(PHP 4, PHP 5)

syslog — システムログのメッセージを生成する

説明

bool **syslog** (int $priority , string $message)

syslog() はシステムログが出力するログメッセージを生成します。

ユーザ定義のログハンドラの設定に関する情報については、Unix マニュアルの syslog.conf (5) を参照ください。 syslog の facility と option に関するより詳細な情報は、 Unix マシンの syslog (3) にあります。

パラメータ

priority

priority は、容易さ (facility) とレベル (level) の組み合わせです。以下の値が使用できます。

syslog() の優先順位 (降順)

定数	説明
LOG_EMERG	システムは使用不可
LOG_ALERT	アクションを直ちにおこす必要がある
LOG_CRIT	致命的な条件
LOG_ERR	エラーを発生する条件
LOG_WARNING	警告を発生する条件
LOG_NOTICE	通常の動作だが、特徴的な条件
LOG_INFO	情報を与えるメッセージ
LOG_DEBUG	デバッグ用のメッセージ

message

残りの引数は送信するメッセージです。ただし、文字 %m は、errno の値に対応するエラーメッセージ文字列 (strerror) に置換されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 syslog() の使用例

```
<?php
define_syslog_variables();
// syslogをオープンし、プロセスIDをインクルードし、標準エラー出力にも
// ログを出力します。そして、ユーザ定義のログ記録機構を使用します。
openlog("myScriptLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);

// 何らかのコード

if (authorized_client()) {
    // 何かをする
} else {
    // クライアントは未認証!
```

```
    // ログを記録する
    $access = date("Y/m/d H:i:s");
    syslog(LOG_WARNING, "Unauthorized client: $access {$_SERVER['REMOTE_ADDR']} ({$_SERVER['HTTP_USER_AGENT']})");
}

closelog();
?>
```

注意

Windows NT では、syslog サービスはイベントログを使用してエミュレートされます。

> **注意**: Windows 環境では、openlog() の facility パラメータに LOG_LOCAL0 から LOG_LOCAL7 までを使用することはできません。

参考

- define_syslog_variables()
- openlog()
- closelog()

目次

Newt 関数

導入

これは RedHat Newt ライブラリのための PHP 拡張モジュールで、 ユーザに優しいアプリケーションを作成するための、ターミナルベースの ウィンドウやウィジェットのライブラリです。 PHP でこの拡張モジュールを有効にすることで Newt ウィジェットが使用可能に なります。この中には ウィンドウやボタン・チェックボックス・ラジオボタン・ ラベル・エディットボックス・スクロールバー・テキストエリア・スケールなどが 含まれます。この拡張モジュールの使用方法は、もとの C 言語用の API と非常によく似ています。

要件

このモジュールは RedHat Newt ライブラリの関数を使用します。 libnewt バージョン >= 0.51.0 が必要です。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/newt.

PHP 4 の場合、この PECL 拡張モジュールのソースは、PHP のソースの ext/ ディレクトリ、または上の PECL リンクで入手可能です。 これらの関数を使用するには、--with-newt[=DIR] オプションを使用し、newt サポートを有効にした上で CGI あるいは CLI 版の PHP を コンパイルする必要があります。

> **注意**: この拡張モジュールは、Windows プラットフォームでは使用できません。
> この拡張モジュールをコンパイルするには、curses ライブラリおよび slang ライブラリも必要です。 これらのライブラリの場所を指定するには、以下の設定オプションを 使用します。 --with-curses-dir=/path/to/libcurses --with-slang-dir=/path/to/libslang

リソース型

この拡張モジュールは 2 つのリソース型 "newt component" および "newt grid" を使用します。 リソース型 "newt component" は関数から返されるもので、一般的な newt ウィジェット (例: newt_button()) を作成します。 リソース型 "newt grid" はコンポーネントの特別なリンク ID で、これは newt グリッドファクトリ関数 (例: newt_create_grid()) によって返されます。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

Newt フォームの終了原因

Newt フォームの終了原因

定数	意味
NEWT_EXIT_HOTKEY	newt_form_add_hot_key() で定義したホットキーが押されました
NEWT_EXIT_COMPONENT	何らかのコンポーネントがフォームを終了させました
NEWT_EXIT_FDREADY	newt_form_watch_fd() で指定したファイル記述子の 読み込みあるいは書き込みの準備が完了しました
NEWT_EXIT_TIMER	newt_form_set_timer() で指定した時間が 経過しました

Newt 色セット

Newt 色セット

定数	意味
NEWT_COLORSET_ROOT	
NEWT_COLORSET_BORDER	
NEWT_COLORSET_WINDOW	
NEWT_COLORSET_SHADOW	
NEWT_COLORSET_TITLE	
NEWT_COLORSET_BUTTON	
NEWT_COLORSET_ACTBUTTON	
NEWT_COLORSET_CHECKBOX	
NEWT_COLORSET_ACTCHECKBOX	
NEWT_COLORSET_ENTRY	
NEWT_COLORSET_LABEL	
NEWT_COLORSET_LISTBOX	
NEWT_COLORSET_ACTLISTBOX	
NEWT_COLORSET_TEXTBOX	
NEWT_COLORSET_ACTTEXTBOX	
NEWT_COLORSET_HELPLINE	
NEWT_COLORSET_ROOTTEXT	
NEWT_COLORSET_ROOTTEXT	
NEWT_COLORSET_EMPTYSCALE	
NEWT_COLORSET_FULLSCALE	
NEWT_COLORSET_DISENTRY	
NEWT_COLORSET_COMPACTBUTTON	
NEWT_COLORSET_ACTSELLISTBOX	
NEWT_COLORSET_SELLISTBOX	

Newt 引数のフラグ

Newt 引数のフラグ

定数	意味
NEWT_ARG_LAST	
NEWT_ARG_APPEND	

Newt フラグの状態

Newt フラグの状態

定数	意味
NEWT_FLAGS_SET	
NEWT_FLAGS_RESET	
NEWT_FLAGS_TOGGLE	

Newt コンポーネントのフラグ

Newt コンポーネントのフラグ

定数	意味
NEWT_FLAG_RETURNEXIT	コンポーネントがアクティブになった際にフォームを終了します
NEWT_FLAG_HIDDEN	コンポーネントは非表示です
NEWT_FLAG_SCROLL	コンポーネントはスクロール可能です
NEWT_FLAG_DISABLED	コンポーネントは使用不能です
NEWT_FLAG_BORDER	
NEWT_FLAG_WRAP	テキストを折り返します
NEWT_FLAG_NOF12	F12 を押してもフォームを終了しません
NEWT_FLAG_MULTIPLE	
NEWT_FLAG_SELECTED	コンポーネントは選択されています
NEWT_FLAG_CHECKBOX	コンポーネントはチェックボックスです
NEWT_FLAG_PASSWORD	入力コンポーネントはパスワード入力欄です
NEWT_FLAG_SHOWCURSOR	カーソルを表示します

ファイル識別子のフラグ

ファイル識別子のフラグ

定数	意味
NEWT_FD_READ	
NEWT_FD_WRITE	
NEWT_FD_EXCEPT	

チェックボックスツリーのフラグ

チェックボックスツリーのフラグ

定数	意味
NEWT_CHECKBOXTREE_UNSELECTABLE	
NEWT_CHECKBOXTREE_HIDE_BOX	
NEWT_CHECKBOXTREE_COLLAPSED	
NEWT_CHECKBOXTREE_EXPANDED	
NEWT_CHECKBOXTREE_UNSELECTED	
NEWT_CHECKBOXTREE_SELECTED	

入力フラグ

入力フラグ

定数	意味
NEWT_ENTRY_SCROLL	
NEWT_ENTRY_HIDDEN	
NEWT_ENTRY_RETURNEXIT	
NEWT_ENTRY_DISABLED	

リストボックスのフラグ

リストボックスのフラグ

定数	意味
NEWT_LISTBOX_RETURNEXIT	

テキストボックスのフラグ

テキストボックスのフラグ

定数	意味
NEWT_TEXTBOX_WRAP	テキストボックスのテキストを折り返します
NEWT_TEXTBOX_SCROLL	テキストボックスのテキストをスクロールします

フォームのフラグ

フォームのフラグ

定数	意味
NEWT_FORM_NOF12	F12 を押しても終了しません

Newt キー

Newt キー

定数	意味
NEWT_KEY_TAB	
NEWT_KEY_ENTER	
NEWT_KEY_SUSPEND	
NEWT_KEY_ESCAPE	
NEWT_KEY_RETURN	
NEWT_KEY_EXTRA_BASE	
NEWT_KEY_UP	
NEWT_KEY_DOWN	
NEWT_KEY_LEFT	
NEWT_KEY_RIGHT	
NEWT_KEY_BKSPC	
NEWT_KEY_DELETE	
NEWT_KEY_HOME	
NEWT_KEY_END	
NEWT_KEY_UNTAB	
NEWT_KEY_PGUP	
NEWT_KEY_PGDN	
NEWT_KEY_INSERT	
NEWT_KEY_F1	
NEWT_KEY_F2	
NEWT_KEY_F3	
NEWT_KEY_F4	
NEWT_KEY_F5	

定数	意味
NEWT_KEY_F6	
NEWT_KEY_F7	
NEWT_KEY_F8	
NEWT_KEY_F9	
NEWT_KEY_F10	
NEWT_KEY_F11	
NEWT_KEY_F12	
NEWT_KEY_RESIZE	

Newt アンカー

Newt アンカー

定数	意味
NEWT_ANCHOR_LEFT	
NEWT_ANCHOR_RIGHT	
NEWT_ANCHOR_TOP	
NEWT_ANCHOR_BOTTOM	

グリッドのフラグ

グリッドのフラグ

定数	意味
NEWT_GRID_FLAG_GROWX	
NEWT_GRID_FLAG_GROWY	
NEWT_GRID_EMPTY	
NEWT_GRID_COMPONENT	
NEWT_GRID_SUBGRID	

例

この例は、RedHat の 'setup' ユーティリティダイアログを PHP に移植した もので、テキストモードで動作します。

Example#1 Newt の使用例

```
<?php
     newt_init ();
    newt_cls ();

    newt_draw_root_text (0, 0, "Test Mode Setup Utility 1.12");
    newt_push_help_line (null);
    newt_draw_root_text (-30, 0, "(c) 1999-2002 RedHat, Inc");

    newt_get_screen_size (&$rows, &$cols);

    newt_open_window ($rows/2-17, $cols/2-10, 34, 17, "Choose a Tool");

    $form = newt_form ();

    $list = newt_listbox (3, 2, 10);

     foreach (array (
        "Authentication configuration",
        "Firewall configuration",
        "Mouse configuration",
        "Network configuration",
        "Printer configuration",
        "System services") as $l_item)
    {
        newt_listbox_add_entry ($list, $l_item, $l_item);
    }

    $b1 = newt_button (5, 12, "Run Tool");
    $b2 = newt_button (21, 12, "Quit");

    newt_form_add_component ($form, $list);
    newt_form_add_components ($form, array($b1, $b2));

    newt_refresh ();
    newt_run_form ($form);

    newt_pop_window ();
    newt_pop_help_line ();
```

```
    newt_finished ();
    newt_form_destroy ($form);
?>
```

---

newt_bell

(PECL newt:0.1-1.1)

newt_bell — ビープ音を端末に送信する

説明

void **newt_bell** (void)

この関数は、端末にビープ音を送信します。

注意: 端末の設定により、この音が聞こえることも聞こえないこともあります。

返り値

値を返しません。

---

newt_button_bar

(PECL newt:0.1-1.1)

newt_button_bar — 作成したボタンを含むグリッドを返す

説明

resource **newt_button_bar** (array &$buttons)

この関数は、作成したボタンを含むグリッドを返します。

パラメータ

buttons

返り値

作成したボタンを含むグリッドを返します。

---

newt_button

(PECL newt:0.1-1.1)

newt_button — 新しいボタンを作成する

説明

resource **newt_button** (int $left , int $top , string $text)

新しいボタンを作成します。

パラメータ

left

ボタンの X 座標。

top

ボタンの Y 座標。

text

ボタンに表示するテキスト。

返り値

作成したボタンコンポーネントへのリソースリンク、あるいはエラー時に **FALSE** を返します。

例

Example#1 newt_button() の例

```
<?php
$form = newt_form();
$ok_button = newt_button(5, 12, "Run Tool");
newt_form_add_component($form, $ok_button);
```

```
?>
```

参考

- newt_button_bar()

---

newt_centered_window

(PECL newt:0.1-1.1)

newt_centered_window — 画面の中央に指定したサイズのウィンドウをオープンする

説明

int **newt_centered_window** (int $width , int $height [, string $title])

画面の中央に、指定したサイズのウィンドウをオープンします。

パラメータ

width

ウィンドウの幅。

height

ウィンドウの高さ。

title

ウィンドウのタイトル。

返り値

未定義です。

参考

- newt_pop_window()
- newt_open_window()

---

newt_checkbox_get_value

(PECL newt:0.1-1.1)

newt_checkbox_get_value — チェックボックスリソースの値を取得する

説明

string **newt_checkbox_get_value** (resource $checkbox)

この関数は、シーケンス内の文字を返します。 これは、チェックボックスの現在の値を表します。

パラメータ

checkbox

返り値

チェックボックスの値を表す文字を返します。

---

newt_checkbox_set_flags

(PECL newt:0.1-1.1)

newt_checkbox_set_flags — チェックボックスリソースを設定する

説明

void **newt_checkbox_set_flags** (resource $checkbox , int $flags , int $sense)

この関数は、チェックボックスリソースのさまざまなフラグを設定します。

パラメータ

checkbox

flags

sense

返り値

値を返しません。

---

newt_checkbox_set_value

(PECL newt:0.1-1.1)

newt_checkbox_set_value ― チェックボックスの値を設定する

説明

void ***newt_checkbox_set_value*** *(resource $checkbox , string $value)*

この関数は、チェックボックスリソースの現在の値を設定します。

パラメータ

checkbox

value

返り値

値を返しません。

---

newt_checkbox_tree_add_item

(PECL newt:0.1-1.1)

newt_checkbox_tree_add_item ― 新しいアイテムをチェックボックスツリーに追加する

説明

void ***newt_checkbox_tree_add_item*** *(resource $checkboxtree , string $text , mixed $data , int $flags , int $index [, int $...])*

この関数は、新しいアイテムをチェックボックスツリーに追加します。

パラメータ

checkboxtree

text

data

flags

index

返り値

値を返しません。

---

newt_checkbox_tree_find_item

(PECL newt:0.1-1.1)

newt_checkbox_tree_find_item ― チェックボックスツリーのアイテムを探す

説明

array ***newt_checkbox_tree_find_item*** *(resource $checkboxtree , mixed $data)*

データを指定して、チェックボックスツリーのアイテムを探します。

パラメータ

checkboxtree

data

返り値

チェックボックスツリーアイテムリソースを返します。 見つからなかった場合は ***NULL*** *を返します。*

---

newt_checkbox_tree_get_current

(PECL newt:0.1-1.1)

newt_checkbox_tree_get_current — チェックボックスツリーの選択されているアイテムを返す

説明

mixed **newt_checkbox_tree_get_current** (resource $checkboxtree)

このメソッドは、チェックボックスツリーの選択されているアイテムを返します。

パラメータ

checkboxtree

返り値

現在の（選択されている）チェックボックスツリーアイテムを返します。

---

newt_checkbox_tree_get_entry_value

(PECL newt:0.1-1.1)

newt_checkbox_tree_get_entry_value —

説明

string **newt_checkbox_tree_get_entry_value** (resource $checkboxtree , mixed $data)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

checkboxtree

data

返り値

---

newt_checkbox_tree_get_multi_selection

(PECL newt:0.1-1.1)

newt_checkbox_tree_get_multi_selection —

説明

array **newt_checkbox_tree_get_multi_selection** (resource $checkboxtree , string $seqnum)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

checkboxtree

seqnum

返り値

---

newt_checkbox_tree_get_selection

(PECL newt:0.1-1.1)

newt_checkbox_tree_get_selection —

説明

array **newt_checkbox_tree_get_selection** (resource $checkboxtree)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

checkboxtree

返り値

---

newt_checkbox_tree_multi

(PECL newt:0.1-1.1)

newt_checkbox_tree_multi —

説明

resource **newt_checkbox_tree_multi** (int $left , int $top , int $height , string $seq [, int $flags])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

height

seq

flags

返り値

---

newt_checkbox_tree_set_current

(PECL newt:0.1-1.1)

newt_checkbox_tree_set_current —

説明

void **newt_checkbox_tree_set_current** (resource $checkboxtree , mixed $data)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

checkboxtree

data

返り値

値を返しません。

---

newt_checkbox_tree_set_entry_value

(PECL newt:0.1-1.1)

newt_checkbox_tree_set_entry_value —

説明

void **newt_checkbox_tree_set_entry_value** (resource $checkboxtree , mixed $data , string $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

checkboxtree

data

value

返り値

値を返しません。

---

newt_checkbox_tree_set_entry

(PECL newt:0.1-1.1)

newt_checkbox_tree_set_entry —

説明

void **newt_checkbox_tree_set_entry** (resource $checkboxtree , mixed $data , string $text)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

checkboxtree

data

text

返り値

値を返しません。

---

newt_checkbox_tree_set_width

(PECL newt:0.1-1.1)

newt_checkbox_tree_set_width —

説明

void **newt_checkbox_tree_set_width** (resource $checkbox_tree , int $width)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

checkbox_tree

width

返り値

値を返しません。

---

newt_checkbox_tree

(PECL newt:0.1-1.1)

newt_checkbox_tree —

説明

resource **newt_checkbox_tree** (int $left , int $top , int $height [, int $flags])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

height

flags

返り値

---

newt_checkbox

(PECL newt:0.1-1.1)

newt_checkbox —

説明

resource **newt_checkbox** (int $left , int $top , string $text , string $def_value [, string $seq])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

text

def_value

seq

返り値

---

newt_clear_key_buffer

(PECL newt:0.1-1.1)

newt_clear_key_buffer — 追加の入力を待たずに、端末の入力バッファの内容をクリアする

説明

void ***newt_clear_key_buffer*** *(void)*

追加の入力を待たずに、端末の入力バッファの内容をクリアします。

返り値

値を返しません。

参考

- *newt_wait_for_key()*

---

newt_cls

(PECL newt:0.1-1.1)

newt_cls —

説明

void ***newt_cls*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

返り値

値を返しません。

---

newt_compact_button

(PECL newt:0.1-1.1)

newt_compact_button —

説明

resource ***newt_compact_button*** *(int $left , int $top , string $text)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

text

返り値

---

newt_component_add_callback

(PECL newt:0.1-1.1)

newt_component_add_callback —

説明

void **newt_component_add_callback** (resource $component , mixed $func_name , mixed $data)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

component

func_name

data

返り値

値を返しません。

---

newt_component_takes_focus

(PECL newt:0.1-1.1)

newt_component_takes_focus —

説明

void **newt_component_takes_focus** (resource $component , bool $takes_focus)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

component

takes_focus

返り値

値を返しません。

---

newt_create_grid

(PECL newt:0.1-1.1)

newt_create_grid —

説明

resource **newt_create_grid** (int $cols , int $rows)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

cols

rows

返り値

---

newt_cursor_off

(PECL newt:0.1-1.1)

newt_cursor_off —

説明

void **newt_cursor_off** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

返り値

値を返しません。

---

newt_cursor_on

(PECL newt:0.1-1.1)

*newt_cursor_on — *

説明

void ***newt_cursor_on*** *(void)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

---

newt_delay

(PECL newt:0.1-1.1)

*newt_delay — *

説明

void ***newt_delay*** *(int $microseconds)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

microseconds

返り値

値を返しません。

---

newt_draw_form

(PECL newt:0.1-1.1)

*newt_draw_form — *

説明

void ***newt_draw_form*** *(resource $form)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

返り値

値を返しません。

---

newt_draw_root_text

(PECL newt:0.1-1.1)

newt_draw_root_text — 指定した位置に文字列を表示する

説明

void ***newt_draw_root_text*** *(int $left , int $top , string $text)*

指定した位置に文字列を表示します。

パラメータ

left

カラム番号。

注意: left が負の数の場合、画面の反対側から位置が計算されます。

top

行番号。

注意: top が負の数の場合、画面の反対側から位置が計算されます。

text

表示する文字列。

返り値

値を返しません。

例

Example#1 newt_draw_root_text() の例

このコードは、画面の両側の隅にタイトルを表示します。

```
<?php
 newt_init();
 newt_cls();

 newt_draw_root_text (2, 0, "Some root text");
 newt_refresh();
 sleep(1);

 newt_draw_root_text (-30, 0, "Root text in the other corner");
 newt_refresh();
 sleep(1);

 newt_finished();
?>
```

参考

- newt_push_help_line()
- newt_pop_help_line()

---

newt_entry_get_value

(PECL newt:0.1-1.1)

newt_entry_get_value —

説明

string **newt_entry_get_value** (resource $entry)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

entry

返り値

---

newt_entry_set_filter

(PECL newt:0.1-1.1)

newt_entry_set_filter —

説明

void **newt_entry_set_filter** (resource $entry , callback $filter , mixed $data)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

entry

filter

data

返り値

値を返しません。

---

newt_entry_set_flags

(PECL newt:0.1-1.1)

newt_entry_set_flags —

説明

void ***newt_entry_set_flags*** *(resource $entry , int $flags , int $sense)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

entry

flags

sense

返り値

値を返しません。

---

newt_entry_set

(PECL newt:0.1-1.1)

newt_entry_set —

説明

void ***newt_entry_set*** *(resource $entry , string $value [, bool $cursor_at_end])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

entry

value

cursor_at_end

返り値

値を返しません。

---

newt_entry

(PECL newt:0.1-1.1)

newt_entry —

説明

resource ***newt_entry*** *(int $left , int $top , int $width [, string $init_value [, int $flags]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

width

init_value

flags

返り値

---

newt_finished

(PECL newt:0.1-1.1)

newt_finished — newt インターフェースを終了する

説明

int **newt_finished** (void)

newt インターフェースを終了します。プログラムが終了する際にこの関数を コールします。

返り値

成功した場合に 1、失敗した場合に 0 を返します。

参考

- newt_init()

---

newt_form_add_component

(PECL newt:0.1-1.1)

newt_form_add_component — フォームにコンポーネントを追加する

説明

void **newt_form_add_component** (resource $form , resource $component)

form にコンポーネントを追加します。

パラメータ

form

コンポーネントを追加するフォーム。

component

フォームに追加するコンポーネント。

返り値

値を返しません。

例

Example#1 newt_form_add_component() の例

```
<?php
$form = newt_form();

$options = array("Authentication configuration", "Firewall configuration",
"Mouse configuration", "Network configuration", "Printer configuration",
"System services");

$list = newt_listbox(3, 2, 10);

foreach ($options as $l_item) {
    newt_listbox_add_entry($list, $l_item, $l_item);
}

newt_form_add_component($form, $list);
?>
```

参考

- newt_form_add_components()

---

newt_form_add_components

(PECL newt:0.1-1.1)

newt_form_add_components — フォームに複数のコンポーネントを追加する

説明

void **newt_form_add_components** (resource $form , array $components)

form に複数のコンポーネントを追加します。

パラメータ

form

コンポーネントを追加するフォーム。

components

フォームに追加するコンポーネントの配列。

返り値

値を返しません。

例

Example#1 newt_form_add_components() の例

```
<?php
$form = newt_form();

$b1 = newt_button(5, 12, "Run Tool");
$b2 = newt_button(21, 12, "Quit");

newt_form_add_components($form, array($b1, $b2));
?>
```

参考

- newt_form_add_component()

---

newt_form_add_hot_key

(PECL newt:0.1-1.1)

newt_form_add_hot_key —

説明

void **newt_form_add_hot_key** (resource $form , int $key)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

key

返り値

値を返しません。

---

newt_form_destroy

(PECL newt:0.1-1.1)

newt_form_destroy — フォームを破壊する

説明

void **newt_form_destroy** (resource $form)

この関数は、フォームおよびそこに追加されたすべてのコンポーネント（サブフォーム上のコンポーネントも含みます）が使用しているメモリ リソースを開放します。いちどフォームが破壊されると、フォーム上の コンポーネントは使用できなくなります。

パラメータ

form

破壊されるフォームコンポーネント。

返り値

値を返しません。

参考

- newt_form_run()
- newt_run_form()

newt_form_get_current

(PECL newt:0.1-1.1)

newt_form_get_current —

説明

resource **newt_form_get_current** (resource $form)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

返り値

newt_form_run

(PECL newt:0.1-1.1)

newt_form_run — フォームを実行する

説明

void **newt_form_run** (resource $form , array &$exit_struct)

この関数は、引数で渡したフォームを実行します。

パラメータ

form

フォームコンポーネント。

exit_struct

フォームコンポーネントを実行した後の情報を返すために使用する 配列。キーと値については以下の表で示します。

Form の終了情報の構造

インデックスのキー	値の型	説明
reason	integer	フォームが終了した理由。とりうる値については ここ で定義しています。
watch	resource	newt_form_watch_fd() で指定したリソースリンク。
キー	整数	ホットキー
component	resource	フォームを終了させたコンポーネント。

返り値

値を返しません。

参考

- newt_run_form()

newt_form_set_background

(PECL newt:0.1-1.1)

newt_form_set_background —

説明

void **newt_form_set_background** (resource $from , int $background)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

from

background

返り値

値を返しません。

newt_form_set_height

(PECL newt:0.1-1.1)

newt_form_set_height —

説明

void **newt_form_set_height** (resource $form , int $height)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

height

返り値

値を返しません。

newt_form_set_size

(PECL newt:0.1-1.1)

newt_form_set_size —

説明

void **newt_form_set_size** (resource $form)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

返り値

値を返しません。

newt_form_set_timer

(PECL newt:0.1-1.1)

newt_form_set_timer —

説明

void **newt_form_set_timer** (resource $form , int $milliseconds)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

milliseconds

返り値

値を返しません。

newt_form_set_width

(PECL newt:0.1-1.1)

newt_form_set_width —

説明

void **newt_form_set_width** (resource $form , int $width)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

width

返り値

値を返しません。

---

newt_form_watch_fd

(PECL newt:0.1-1.1)

newt_form_watch_fd —

説明

```
void newt_form_watch_fd ( resource $form , resource $stream [, int $flags ] )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

form

stream

flags

返り値

値を返しません。

---

newt_form

(PECL newt:0.1-1.1)

newt_form — フォームを作成する

説明

```
resource newt_form ([ resource $vert_bar [, string $help [, int $flags ]]] )
```

新しいフォームを作成します。

パラメータ

vert_bar

フォームに関連付けられる垂直スクロールバー。

help

ヘルプ文字列。

flags

さまざまなフラグ。

返り値

作成されたフォームコンポーネントのリソースリンク、 あるいはエラー時に **FALSE** を返します。

例

Example#1 newt_form() の例

"Quit" ボタンを表示し、それが押されるとアプリケーションを 終了させます。

```
<?php
newt_init();
newt_cls();

$myform = newt_form();
$button = newt_button (5, 12, "Quit");

newt_form_add_component ($myform, $button);
newt_refresh ();
newt_run_form ($myform);

newt_finished ();
newt_form_destroy ($myform);
```

```
?>
```

参考

- newt_form_run()
- newt_run_form()
- newt_form_add_component()
- newt_form_add_components()
- newt_form_destroy()

---

newt_get_screen_size

(PECL newt:0.1-1.1)

newt_get_screen_size — 参照で渡された引数に、現在の端末の大きさを格納する

説明

void **newt_get_screen_size** (int &$cols , int &$rows)

参照で渡された引数に、現在の端末の大きさを格納します。

パラメータ

cols

端末のカラム数。

rows

端末の行数。

返り値

値を返しません。

例

Example#1 newt_get_screen_size() の例

このコードは、端末の画面の大きさを表示します。

```
<?php
  newt_init();
 newt_get_screen_size  (&$cols, &$rows);
 newt_finished();

 print "端末のサイズは {$cols}x{$rows} です。\n";
?>
```

上の例の出力は以下となります。

```
端末のサイズは 138x47 です。
```

---

newt_grid_add_components_to_form

(PECL newt:0.1-1.1)

newt_grid_add_components_to_form —

説明

void **newt_grid_add_components_to_form** (resource $grid , resource $form , bool $recurse)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

grid

form

recurse

返り値

値を返しません。

---

newt_grid_basic_window

(PECL newt:0.1-1.1)

newt_grid_basic_window —

説明

resource **newt_grid_basic_window** (resource $text , resource $middle , resource $buttons)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

text

middle

buttons

返り値

---

newt_grid_free

(PECL newt:0.1-1.1)

newt_grid_free —

説明

void **newt_grid_free** (resource $grid , bool $recurse)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

grid

recurse

返り値

値を返しません。

---

newt_grid_get_size

(PECL newt:0.1-1.1)

newt_grid_get_size —

説明

void **newt_grid_get_size** (resouce $grid , int &$width , int &$height)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

grid

width

height

返り値

値を返しません。

---

newt_grid_h_close_stacked

(PECL newt:0.1-1.1)

newt_grid_h_close_stacked —

説明

resource **newt_grid_h_close_stacked** (int $element1_type , resource $element1 [, int $... [, resource $...]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

element1_type

element1

返り値

---

newt_grid_h_stacked

(PECL newt:0.1-1.1)

newt_grid_h_stacked —

説明

resource ***newt_grid_h_stacked*** *(int $element1_type , resource $element1 [, int $... [, resource $...]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

element1_type

element1

返り値

---

newt_grid_place

(PECL newt:0.1-1.1)

newt_grid_place —

説明

void ***newt_grid_place*** *(resource $grid , int $left , int $top)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

grid

left

top

返り値

値を返しません。

---

newt_grid_set_field

(PECL newt:0.1-1.1)

newt_grid_set_field —

説明

void ***newt_grid_set_field*** *(resource $grid , int $col , int $row , int $type , resource $val , int $pad_left , int $pad_top , int $pad_right , int $pad_bottom , int $anchor [, int $flags])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

grid

col

row

type

val

pad_left

pad_top

pad_right

pad_bottom

anchor

flags

返り値

値を返しません。

---

newt_grid_simple_window

(PECL newt:0.1-1.1)

newt_grid_simple_window —

説明

resource ***newt_grid_simple_window*** *(resource $text , resource $middle , resource $buttons)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

text

middle

buttons

返り値

---

newt_grid_v_close_stacked

(PECL newt:0.1-1.1)

newt_grid_v_close_stacked —

説明

resource ***newt_grid_v_close_stacked*** *(int $element1_type , resource $element1 [, int $... [, resource $...]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

element1_type

element1

返り値

---

newt_grid_v_stacked

(PECL newt:0.1-1.1)

newt_grid_v_stacked —

説明

resource ***newt_grid_v_stacked*** *(int $element1_type , resource $element1 [, int $... [, resource $...]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

element1_type

element1

返り値

newt_grid_wrapped_window_at

(PECL newt:0.1-1.1)

newt_grid_wrapped_window_at —

説明

void **newt_grid_wrapped_window_at** (resource $grid , string $title , int $left , int $top)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

grid

title

left

top

返り値

値を返しません。

---

newt_grid_wrapped_window

(PECL newt:0.1-1.1)

newt_grid_wrapped_window —

説明

void **newt_grid_wrapped_window** (resource $grid , string $title)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

grid

title

返り値

値を返しません。

---

newt_init

(PECL newt:0.1-1.1)

newt_init — newt を初期化する

説明

int **newt_init** (void)

newt インターフェースを初期化します。この関数は、他の newt 関数の前に コールする必要があります。

返り値

成功した場合に 1、失敗した場合に 0 を返します。

参考

- newt_finished()

---

newt_label_set_text

(PECL newt:0.1-1.1)

newt_label_set_text —

説明

void ***newt_label_set_text*** *(resource $label , string $text)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

label

text

返り値

値を返しません。

---

newt_label

(PECL newt:0.1-1.1)

newt_label —

説明

resource ***newt_label*** *(int $left , int $top , string $text)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

text

返り値

---

newt_listbox_append_entry

(PECL newt:0.1-1.1)

newt_listbox_append_entry —

説明

void ***newt_listbox_append_entry*** *(resource $listbox , string $text , mixed $data)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

text

data

返り値

値を返しません。

---

newt_listbox_clear_selection

(PECL newt:0.1-1.1)

newt_listbox_clear_selection —

説明

void ***newt_listbox_clear_selection*** *(resource $listbox)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

返り値

値を返しません。

---

newt_listbox_clear

(PECL newt:0.1-1.1)

newt_listbox_clear —

説明

void **newt_listbox_clear** (resource $listobx)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listobx

返り値

値を返しません。

---

newt_listbox_delete_entry

(PECL newt:0.1-1.1)

newt_listbox_delete_entry —

説明

void **newt_listbox_delete_entry** (resource $listbox , mixed $key)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

key

返り値

値を返しません。

---

newt_listbox_get_current

(PECL newt:0.1-1.1)

newt_listbox_get_current —

説明

string **newt_listbox_get_current** (resource $listbox)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

返り値

---

newt_listbox_get_selection

(PECL newt:0.1-1.1)

newt_listbox_get_selection —

説明

array **newt_listbox_get_selection** (resource $listbox)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

返り値

---

newt_listbox_insert_entry

(PECL newt:0.1-1.1)

newt_listbox_insert_entry —

説明

void ***newt_listbox_insert_entry*** *(resource $listbox , string $text , mixed $data , mixed $key)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

text

data

key

返り値

値を返しません。

---

newt_listbox_item_count

(PECL newt:0.1-1.1)

newt_listbox_item_count —

説明

int ***newt_listbox_item_count*** *(resource $listbox)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

返り値

---

newt_listbox_select_item

(PECL newt:0.1-1.1)

newt_listbox_select_item —

説明

void ***newt_listbox_select_item*** *(resource $listbox , mixed $key , int $sense)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

key

sense

返り値

値を返しません。

---

newt_listbox_set_current_by_key

(PECL newt:0.1-1.1)

newt_listbox_set_current_by_key —

説明

void **newt_listbox_set_current_by_key** (resource $listbox , mixed $key)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

key

返り値

値を返しません。

---

newt_listbox_set_current

(PECL newt:0.1-1.1)

newt_listbox_set_current —

説明

void **newt_listbox_set_current** (resource $listbox , int $num)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

num

返り値

値を返しません。

---

newt_listbox_set_data

(PECL newt:0.1-1.1)

newt_listbox_set_data —

説明

void **newt_listbox_set_data** (resource $listbox , int $num , mixed $data)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

num

data

返り値

値を返しません。

---

newt_listbox_set_entry

(PECL newt:0.1-1.1)

newt_listbox_set_entry —

説明

void **newt_listbox_set_entry** (resource $listbox , int $num , string $text)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

num

text

返り値

値を返しません。

---

newt_listbox_set_width

(PECL newt:0.1-1.1)

newt_listbox_set_width —

説明

void ***newt_listbox_set_width*** *(resource $listbox , int $width)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

listbox

width

返り値

値を返しません。

---

newt_listbox

(PECL newt:0.1-1.1)

newt_listbox —

説明

resource ***newt_listbox*** *(int $left , int $top , int $height [, int $flags])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

height

flags

返り値

---

newt_listitem_get_data

(PECL newt:0.1-1.1)

newt_listitem_get_data —

説明

mixed ***newt_listitem_get_data*** *(resource $item)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

item

返り値

---

newt_listitem_set

(PECL newt:0.1-1.1)

newt_listitem_set —

説明

void **newt_listitem_set** (resource $item , string $text)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

item

text

返り値

値を返しません。

---

newt_listitem

(PECL newt:0.1-1.1)

newt_listitem —

説明

resource **newt_listitem** (int $left , int $top , string $text , bool $is_default , resouce $prev_item , mixed $data [, int $flags])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

text

is_default

prev_item

data

flags

返り値

---

newt_open_window

(PECL newt:0.1-1.1)

newt_open_window — 指定したサイズと位置でウィンドウをオープンする

説明

int **newt_open_window** (int $left , int $top , int $width , int $height [, string $title])

指定したサイズと位置でウィンドウをオープンします。

パラメータ

left

ウィンドウの左上隅の位置(カラム番号)。

top

ウィンドウの左上隅の位置(行番号)。

width

ウィンドウの幅。

height

ウィンドウの高さ。

title

ウィンドウのタイトル。

返り値

成功した場合に 1、失敗した場合に 0 を返します。

参考

- *newt_pop_window()*
- *newt_centered_window()*

---

newt_pop_help_line

(PECL newt:0.1-1.1)

newt_pop_help_line — 現在のヘルプ行をスタックの内容で置き換える

説明

void ***newt_pop_help_line*** *(void)*

現在のヘルプ行を、スタックの内容で置き換えます。

> ***注意***: *newt_push_help_line()* *をコールした回数をこえて* ***newt_pop_help_line()*** *をコールしないようにすることが 重要です。*

返り値

値を返しません。

参考

- *newt_push_help_line()*

---

newt_pop_window

(PECL newt:0.1-1.1)

newt_pop_window — トップウィンドウを画面から消去する

説明

void ***newt_pop_window*** *(void)*

トップウィンドウを画面から消去し、ウィンドウが覆っていた部分を 再描画します。

返り値

値を返しません。

参考

- *newt_open_window()*
- *newt_centered_window()*

---

newt_push_help_line

(PECL newt:0.1-1.1)

newt_push_help_line — 現在のヘルプ行をスタックに保存し、新しい行を表示する

説明

void ***newt_push_help_line*** *([string $text])*

現在のヘルプ行をスタックに保存し、新しい行を表示します。

パラメータ

text

新しいヘルプメッセージ。

> ***注意***: *指定しなかった場合、ヘルプ行はクリアされます。*

返り値

値を返しません。

参考

- newt_pop_help_line()

---

newt_radio_get_current

(PECL newt:0.1-1.1)

newt_radio_get_current —

説明

resource **newt_radio_get_current** (resource $set_member)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

set_member

返り値

---

newt_radiobutton

(PECL newt:0.1-1.1)

newt_radiobutton —

説明

resource **newt_radiobutton** (int $left , int $top , string $text , bool $is_default [, resource $prev_button])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

text

is_default

prev_button

返り値

---

newt_redraw_help_line

(PECL newt:0.1-1.1)

newt_redraw_help_line —

説明

void **newt_redraw_help_line** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

返り値

値を返しません。

---

newt_reflow_text

(PECL newt:0.1-1.1)

newt_reflow_text —

説明

string **newt_reflow_text** (string $text , int $width , int $flex_down , int $flex_up , int &$actual_width , int &$actual_height)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

text

width

flex_down

flex_up

actual_width

actual_height

返り値

---

newt_refresh

(PECL newt:0.1-1.1)

newt_refresh — 画面の変更された部分を更新する

説明

void **newt_refresh** (void)

パフォーマンスを向上させるため、newt は必要時にしか画面を更新しません。 それはプログラムが端末への書き込みを行った場合とは限りません。 変更した部分を即時に更新させるようにするために、この関数をコールします。

返り値

値を返しません。

---

newt_resize_screen

(PECL newt:0.1-1.1)

newt_resize_screen —

説明

void **newt_resize_screen** ([bool $redraw])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

redraw

返り値

値を返しません。

---

newt_resume

(PECL newt:0.1-1.1)

newt_resume — newt_suspend() をコールした後に newt インターフェースの使用を再開する

説明

void **newt_resume** (void)

newt_suspend() をコールした後に newt インターフェースの使用を再開します。

返り値

値を返しません。

参考

- newt_suspend()

---

newt_run_form

(PECL newt:0.1-1.1)

newt_run_form — フォームを実行する

説明

resource **newt_run_form** (resource $form)

この関数は、引数で渡したフォームを実行します。

パラメータ

form

フォームコンポーネント。

返り値

フォームの実行を停止させたコンポーネントを返します。

> **注意**: この関数は、newt の通常の名前付け規則に当てはまらないことに注意 しましょう。これは古いインターフェースであり、すべてのフォームで 動作するわけではありません。これは、昔のアプリケーションのために だけ newt に残されています。しかし、この関数のインターフェースは newt_form_run() よりシンプルであるため、 結果としていまだに頻繁に利用されています。 アプリケーションが終了した際には、フォームおよびそこに含まれる すべてのコンポーネントは破壊されます。

参考

- newt_form_run()
- newt_form_destroy()

---

newt_scale_set

(PECL newt:0.1-1.1)

newt_scale_set —

説明

void **newt_scale_set** (resource $scale , int $amount)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

scale

amount

返り値

値を返しません。

---

newt_scale

(PECL newt:0.1-1.1)

newt_scale —

説明

resource **newt_scale** (int $left , int $top , int $width , int $full_value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

width

full_value

返り値

---

newt_scrollbar_set

(PECL newt:0.1-1.1)

newt_scrollbar_set —

説明

void **newt_scrollbar_set** (resource $scrollbar , int $where , int $total)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

scrollbar

where

total

返り値

値を返しません。

---

newt_set_help_callback

(PECL newt:0.1-1.1)

newt_set_help_callback —

説明

void **newt_set_help_callback** (mixed $function)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

function

返り値

値を返しません。

---

newt_set_suspend_callback

(PECL newt:0.1-1.1)

newt_set_suspend_callback — ユーザがサスペンドキーを押した際に起動するコールバック関数を設定する

説明

void **newt_set_suspend_callback** (callback $function , mixed $data)

ユーザがサスペンドキー（通常は ^Z）を押した際に起動するコールバック関数を 設定します。コールバックが登録されていない場合、サスペンドのキー入力は 無視されます。

パラメータ

function

ひとつの引数 data を受け取るコールバック関数。

data

コールバック関数に渡されるデータ。

参考

- newt_suspend()
- newt_resume()

---

newt_suspend

(PECL newt:0.1-1.1)

newt_suspend — 端末を元の状態に戻すよう、newt に通知する

説明

void **newt_suspend** (void)

端末を元の状態に戻すよう、newt に通知します。いったん実行されると、（自分自身に SIGTSTP を送信する・子プログラムをフォークするなどの方法で）アプリケーションが自分自身でサスペンドすることが可能となります。

返り値

値を返しません。

参考

- newt_resume()

---

newt_textbox_get_num_lines

(PECL newt:0.1-1.1)

newt_textbox_get_num_lines —

説明

int **newt_textbox_get_num_lines** (resource $textbox)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

textbox

返り値

---

newt_textbox_reflowed

(PECL newt:0.1-1.1)

newt_textbox_reflowed —

説明

resource **newt_textbox_reflowed** (int $left , int $top , char $*text , int $width , int $flex_down , int $flex_up [, int $flags])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

*text

width

flex_down

flex_up

flags

返り値

---

newt_textbox_set_height

(PECL newt:0.1-1.1)

newt_textbox_set_height —

説明

void **newt_textbox_set_height** (resource $textbox , int $height)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

textbox

height

返り値

値を返しません。

---

newt_textbox_set_text

(PECL newt:0.1-1.1)

newt_textbox_set_text —

説明

void ***newt_textbox_set_text*** *(resource $textbox , string $text)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

textbox

text

返り値

値を返しません。

---

newt_textbox

(PECL newt:0.1-1.1)

newt_textbox —

説明

resource ***newt_textbox*** *(int $left , int $top , int $width , int $height [, int $flags])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

width

height

flags

返り値

---

newt_vertical_scrollbar

(PECL newt:0.1-1.1)

newt_vertical_scrollbar —

説明

resource ***newt_vertical_scrollbar*** *(int $left , int $top , int $height [, int $normal_colorset [, int $thumb_colorset]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

left

top

height

normal_colorset

thumb_colorset

返り値

---

newt_wait_for_key

(PECL newt:0.1-1.1)

newt_wait_for_key — キーが押されるまで結果を返さない

説明

void ***newt_wait_for_key*** *(void)*

この関数は、キーが押されるまで結果を返しません。 キー入力は無視されます。もし端末のバッファにキー入力が格納されている場合は、 この関数はその内容を捨てて結果をすぐに返します。

返り値

値を返しません。

参考

- newt_clear_key_buffer()

---

newt_win_choice

(PECL newt:0.1-1.1)

newt_win_choice —

説明

int ***newt_win_choice*** *(string $title , string $button1_text , string $button2_text , string $format [, mixed $args [, mixed $...]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

title

button1_text

button2_text

format

args

返り値

---

newt_win_entries

(PECL newt:0.1-1.1)

newt_win_entries —

説明

int ***newt_win_entries*** *(string $title , string $text , int $suggested_width , int $flex_down , int $flex_up , int $data_width , array &$items , string $button1 [, string $...])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

title

text

suggested_width

flex_down

flex_up

data_width

items

button1

button2

返り値

例

Example#1 newt_win_entries() の例

```
<?php

newt_init();
newt_cls();

$entries[] = array('text' => 'First name:', 'value' => &$f_name);
$entries[] = array('text' => 'Last name:',  'value' => &$l_name);

$rc = newt_win_entries("User information", "Please enter your credentials:", 50, 7, 7, 30, $entries, "Ok", "Back");
newt_finished ();

if ($rc != 2) {
    echo "Your name is: $f_name $l_name¥n";
}
?>
```

---

newt_win_menu

(PECL newt:0.1-1.1)

newt_win_menu —

説明

int ***newt_win_menu*** *(string $title , string $text , int $suggestedWidth , int $flexDown , int $flexUp , int $maxListHeight , array $items , int &$listItem [, string $button1 [, string $...]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

title

text

suggestedWidth

flexDown

flexUp

maxListHeight

items

listItem

button1

返り値

値を返しません。

---

newt_win_message

(PECL newt:0.1-1.1)

newt_win_message —

説明

void ***newt_win_message*** *(string $title , string $button_text , string $format [, mixed $args [, mixed $...]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

title

button_text

format

args

返り値

値を返しません。

---

newt_win_messagev

(PECL newt:0.1-1.1)

newt_win_messagev —

説明

void ***newt_win_messagev*** *(string $title , string $button_text , string $format , array $args)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

title

button_text

format

args

返り値

値を返しません。

---

newt_win_ternary

(PECL newt:0.1-1.1)

newt_win_ternary —

説明

int ***newt_win_ternary*** *(string $title , string $button1_text , string $button2_text , string $button3_text , string $format [, mixed $args [, mixed $...]])*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

title

Its description

button1_text

Its description

button2_text

Its description

button3_text

Its description

format

Its description

args

Its description

返り値

What the function returns, first on success, then on failure. See also the &return.success; entity

---

目次

- newt_checkbox_get_value — チェックボックスリソースの値を取得する
- newt_checkbox_set_flags — チェックボックスリソースを設定する
- newt_checkbox_set_value — チェックボックスの値を設定する
- newt_checkbox_tree_add_item — 新しいアイテムをチェックボックスツリーに追加する
- newt_checkbox_tree_find_item — チェックボックスツリーのアイテムを探す
- newt_checkbox_tree_get_current — チェックボックスツリーの選択されているアイテムを返す
- newt_checkbox_tree_get_entry_value — 説明
- newt_checkbox_tree_get_multi_selection — 説明
- newt_checkbox_tree_get_selection — 説明
- newt_checkbox_tree_multi — 説明
- newt_checkbox_tree_set_current — 説明
- newt_checkbox_tree_set_entry_value — 説明
- newt_checkbox_tree_set_entry — 説明
- newt_checkbox_tree_set_width — 説明
- newt_checkbox_tree — 説明
- newt_checkbox — 説明
- newt_clear_key_buffer — 追加の入力を待たずに、端末の入力バッファの内容をクリアする
- newt_cls — 説明
- newt_compact_button — 説明
- newt_component_add_callback — 説明
- newt_component_takes_focus — 説明
- newt_create_grid — 説明
- newt_cursor_off — 説明
- newt_cursor_on — 説明
- newt_delay — 説明
- newt_draw_form — 説明
- newt_draw_root_text — 指定した位置に文字列を表示する
- newt_entry_get_value — 説明
- newt_entry_set_filter — 説明
- newt_entry_set_flags — 説明
- newt_entry_set — 説明
- newt_entry — 説明
- newt_finished — newt インターフェースを終了する
- newt_form_add_component — フォームにコンポーネントを追加する
- newt_form_add_components — フォームに複数のコンポーネントを追加する
- newt_form_add_hot_key — 説明
- newt_form_destroy — フォームを破壊する
- newt_form_get_current — 説明
- newt_form_run — フォームを実行する
- newt_form_set_background — 説明
- newt_form_set_height — 説明
- newt_form_set_size — 説明
- newt_form_set_timer — 説明
- newt_form_set_width — 説明
- newt_form_watch_fd — 説明
- newt_form — フォームを作成する
- newt_get_screen_size — 参照で渡された引数に、現在の端末の大きさを格納する
- newt_grid_add_components_to_form — 説明
- newt_grid_basic_window — 説明
- newt_grid_free — 説明
- newt_grid_get_size — 説明
- newt_grid_h_close_stacked — 説明
- newt_grid_h_stacked — 説明
- newt_grid_place — 説明
- newt_grid_set_field — 説明
- newt_grid_simple_window — 説明
- newt_grid_v_close_stacked — 説明
- newt_grid_v_stacked — 説明
- newt_grid_wrapped_window_at — 説明
- newt_grid_wrapped_window — 説明
- newt_init — newt を初期化する
- newt_label_set_text — 説明
- newt_label — 説明
- newt_listbox_append_entry — 説明

NSAPI用関数

導入

これらの関数は Netscpae/iPlanet/Sun ウェブサーバ上で NSAPI モジュールとして PHP を実行している場合にのみ有効です。

インストール手順

Netscape/iPlanet/Sun ウェブサーバに PHP をインストールする方法については、 インストールの章の NSAPI セクション (UNIX、 Windows) を参照してください。

実行時設定

NSAPI PHPモジュールの振る舞いはphp.iniの設定によって影響されます。 obj.confでのphp4_executeの コールによる追加パラメータによってphp.ini上での設定は上書きされます。

NSAPI 設定オプション

名前	デフォルト	変更の可否	変更履歴
nsapi.read_timeout	"60"	PHP_INI_ALL	PHP 4.3.3 以降で有効です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

nsapi.read_timeout integer

クライアントから POST データが転送される間にプラグインがタイムアウトするまでの秒数

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

参考

NSAPI は、できるだけ互換性を保持するために Apache モジュールの関数のサブセットを実装しています。

NSAPI に実装されている Apache 関数

Apache 関数 (only as alias)	NSAPI 関数	説明
apache_request_headers()	nsapi_request_headers()	HTTP リクエストヘッダを全て取得する
apache_response_headers()	nsapi_response_headers()	HTTP レスポンスヘッダを全て取得する
getallheaders()	nsapi_request_headers()	HTTP リクエストヘッダを全て取得する
virtual()	nsapi_virtual()	NSAPI サブリクエストを発行する

---

nsapi_request_headers

(PHP 4 >= 4.3.3, PHP 5)

nsapi_request_headers — HTTP リクエストヘッダを全て取得する

説明

*array **nsapi_request_headers** (void)*

***nsapi_request_headers()** は、カレントの リクエストの HTTP ヘッダを連想配列で返します。 この関数は、PHP を NSAPI モジュールとして 実行している場合にのみサポートされます。*

> ***注意**: PHP 4.3.3 未満では、getallheaders() は Apache サーバでのみ使用可能でした。 PHP 4.3.3 以降では、NSAPI モジュールを使用している場合、 getallheaders() は **nsapi_request_headers()** へのエイリアスとなります。*
>
> ***注意**: $_SERVER スーパーグローバル変数を使うことによって Common CGI 変数を取得することも可能です。この変数は PHP を NSAPI モジュールとして実行しているか どうかにかかわらず利用できます。*

返り値

HTTP ヘッダを連想配列で返します。

例

Example#1 nsapi_request_headers() の例

```
<?php
$headers = nsapi_request_headers();

foreach ($headers as $header => $value) {
   echo "$header: $value <br />¥n";
}
?>
```

---

nsapi_response_headers

(PHP 4 >= 4.3.3, PHP 5)

nsapi_response_headers — すべての HTTP レスポンスヘッダを取得する

説明

*array **nsapi_response_headers** (void)*

Gets all the NSAPI response headers.

返り値

すべての NSAPI レスポンスヘッダの連想配列を返します。

参考

- nsapi_request_headers()
- headers_sent()

---

nsapi_virtual

(PHP 4 >= 4.3.3, PHP 5)

nsapi_virtual — NSAPI サブリクエストを発行する

説明

bool **nsapi_virtual** (string $uri)

nsapi_virtual() は NSAPI 特有の関数です。 SSI (.shtml ファイル) における <!--#include virtual...--> と等価で、NSAPI サブリクエストを実行します。 CGI スクリプトや .shtmlファイル、 あるいはその他ウェブサーバ上でパースするものを include するのに有用でしょう。

サブリクエストを実行するには、全てのバッファリングを中断し ブラウザにフラッシュする必要があります。 ペンディング状態のヘッダも送出する必要があります。

この関数を使って他の PHP スクリプトに再帰的なリクエストをすることはできません。 PHP スクリプトをインクルードしたい場合には、include() または require() を使用してください。

> **注意**: この関数は、Netscape/iPlanet/Sun ウェブサーバのドキュメント化されていない機構に依存してます。 この関数が有効かどうか確かめるには phpinfo() を使用してください。Unix 環境では常に作動するでしょう。Windows 環境では、 ns-httpdXX.dll ファイルの名称に依存します。
> もしこの問題に遭遇した場合には NSAPI セクションのサブリクエスト (UNIX、Windows) の項をご覧ください。

パラメータ

uri

スクリプトの URI 。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

目次

オブジェクトの集約/合成関数

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

導入

オブジェクト指向プログラミングでは、簡単なクラス (または インスタンス) を組み合わせてより複雑なクラスを作成するということが 一般に行われます。これは、複雑なオブジェクトやオブジェクト階層を 構築するための柔軟な方法であり、多重継承と同等のことを動的に行う 機能を有します。クラス(またはオブジェクト)を合成するには、合成される要素の間の 関係により関連(Association)と 集約(Aggregation)の 2 種類の方法があります。

関連は、独立に構築され、外部から 可視の部分を合成したものです。クラスまたはオブジェクトを 関連づける場合、各クラスは関連するクラスへのリファレンスを保持します。 複数のクラスを静的に関連づける場合は、クラスは他のクラスの インスタンスへのリファレンスを含みます。例えば、

Example#1 クラスの関連付け

```
<?php
class DateTime {

    function DateTime()
    {
        // 空のコンストラクタ
    }

    function now()
    {
        return date("Y-m-d H:i:s");
    }
}

class Report {
```

```
    var $_dt;
   // その他のプロパティ ...

    function Report()
    {
        $this->_dt = new DateTime();
        // 初期化コード ...
    }

     function generateReport()
    {
        $dateTime = $this->_dt->now();
        // その他のコード ...
    }

   // その他のメソッド ...
}

$rep = new Report();
?>
```

コンストラクタ（または他のメソッド）にリファレンスを渡すことにより 実行時に複数のインスタンスを関連づけることも可能です。 これにより、複数のオブジェクト間の関連を動的に変更することが 可能です。この例を示すために上の例を変更してみます。

Example#2 オブジェクトの関連

```
<?php
class DateTime {
   // 上の例と同じ
}

class DateTimePlus {
    var $_format;

     function DateTimePlus($format="Y-m-d H:i:s")
    {
        $this->_format = $format;
    }

     function now()
    {
         return date($this->_format);
    }
}

class Report {
    var $_dt;    // DateTime へのリファレンスをここに保持
    // その他のプロパティ ...

    function Report()
    {
        // 初期化を行う
    }

     function setDateTime(&$dt)
    {
        $this->_dt =& $dt;
    }

     function generateReport()
    {
        $dateTime = $this->_dt->now();
        // その他のコード ...
    }

   // その他のメソッド ...
}

$rep = new Report();
$dt = new DateTime();
$dtp = new DateTimePlus("l, F j, Y (h:i:s a, T)");

// Web 表示用に簡単な日付を付けたレポートを生成する
$rep->setDateTime(&$dt);
echo $rep->generateReport();

// その他のコード ...

// かっこの良いレポートを生成する
$rep->setDateTime(&$dtp);
$output = $rep->generateReport();
// データベースに $output を保存
// ... 等 ...
?>
```

一方、集約では、合成されたパーツのカプセル化（隠蔽）が行われます。(静的な）内部クラス（PHP はまだ内部クラスを サポートしていません）を使用することにより、クラスを集約することが できます。この場合、このクラスを含むクラスを通じる場合以外、集約された クラスの定義にはアクセスできません。複数のインスタンスの集約（オブジェクト集約）は、あるオブジェクトの内部にサブオブジェクトを 動的に作成することを意味し、この過程でこのオブジェクトのプロパティと メソッドを拡張します。

オブジェクトの集約は、(例えば、分子は原子を集約したものであるといった）包含関係を表す際の自然な方法であり、サブクラスを複数の親クラス およびそのインターフェイスに永続的にバインドすることなく、 多重継承と等価な機能を得るために使用できます。 実際、オブジェクトの集約はより柔軟に使用することができ、集約される オブジェクトで継承するメソッドまたはプロパティを選択することが できます。

例

3 つのクラスを定義し、各々に別々のストレージメソッドを実装します。

Example#3 storage_classes.inc

```
<?php
```

```
class FileStorage {
    var $data;

    function FileStorage($data)
    {
        $this->data = $data;
    }

    function write($name)
    {
        $fp = fopen(name, "w");
        fwrite($fp, $this->data);
        fclose($data);
    }
}

class WDDXStorage {
    var $data;
    var $version = "1.0";
    var $_id; // "private" 変数

    function WDDXStorage($data)
    {
        $this->data = $data;
        $this->_id = $this->_genID();
    }

    function store()
    {
        if ($this->_id) {
            $pid = wddx_packet_start($this->_id);
            wddx_add_vars($pid, "this->data");
            $packet = wddx_packet_end($pid);
        } else {
            $packet = wddx_serialize_value($this->data);
        }
        $dbh = dba_open("varstore", "w", "gdbm");
        dba_insert(md5(uniqid("", true)), $packet, $dbh);
        dba_close($dbh);
    }

    // プライベートメソッド
    function _genID()
    {
        return md5(uniqid(rand(), true));
    }
}

class DBStorage {
    var $data;
    var $dbtype = "mysql";

    function DBStorage($data)
    {
        $this->data = $data;
    }

    function save()
    {
        $dbh = mysql_connect();
        mysql_select_db("storage", $dbh);
        $serdata = serialize($this->data);
        mysql_query("insert into vars ('$serdata',now())", $dbh);
        mysql_close($dbh);
    }
}

?>
```

この定義済みクラスを用いていくつかのオブジェクトをインスタンス化し、 集約や集約の解除を行いつつ随時オブジェクトの情報を出力します。

Example#4 test_aggregation.php

```
<?php
include "storageclasses.inc";

// ユーティリティ関数

function p_arr($arr)
{
    foreach ($arr as $k => $v)
        $out[] = "¥t$k => $v";
    return implode("¥n", $out);
}

function object_info($obj)
{
    $out[] = "クラス: " . get_class($obj);
    foreach (get_object_vars($obj) as $var=>$val) {
        if (is_array($val)) {
            $out[] = "プロパティ: $var (array)\n" . p_arr($val);
        } else {
            $out[] = "プロパティ: $var = $val";
        }
    }
    foreach (get_class_methods($obj) as $method) {
        $out[] = "メソッド: $method";
    }
    return implode("¥n", $out);
}
```

```php
$data = array(M_PI, "kludge != cruft");

// 基本オブジェクトを作成する
$fs = new FileStorage($data);
$ws = new WDDXStorage($data);

// オブジェクトの情報を表示する
echo "\$fs オブジェクト\n";
echo object_info($fs) . "¥n";
echo "\n\$ws オブジェクト\n";
echo object_info($ws) . "¥n";

// 集約を行う

echo "\n\$fs を WDDXStorage クラスに集約します\n";
aggregate($fs, "WDDXStorage");
echo "\$fs オブジェクト\n";
echo object_info($fs) . "¥n";

echo "\nそれを DBStorage クラスに集約します\n";
aggregate($fs, "DBStorage");
echo "\$fs オブジェクト\n";
echo object_info($fs) . "¥n";

echo "\nWDDXStorage を集約から解除します\n";
deaggregate($fs, "WDDXStorage");
echo "\$fs オブジェクト\n";
echo object_info($fs) . "¥n";

?>
```

出力内容を見ながら、PHP の集約についての副作用や制限事項を 考えてみましょう。 まず、新しく作成されたオブジェクト $fs および $ws は、期待通りの（対応するクラス定義に もとづく）結果を出力します。 PHP では実際のところクラス/オブジェクトの要素についてパブリック/ プライベートの区別はありませんが、集約の際には クラス/オブジェクトのプライベート要素はアンダースコア文字（"_"）で始まるとみなします。

```
$fs オブジェクト
クラス: filestorage
プロパティ: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
メソッド: filestorage
メソッド: write

$ws オブジェクト
クラス: wddxstorage
プロパティ: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
プロパティ: version = 1.0
プロパティ: _id = ID::9bb2b640764d4370eb04808af8b076a5
メソッド: wddxstorage
メソッド: store
メソッド: _genid
```

次に $fs を WDDXStorage クラスと集約し、オブジェクトの情報を出力します。 $fs は今でも FileStorage のままですが、プロパティ $version およびメソッド **store()** が存在することがわかるでしょう。これらは いずれも WDDXStorage で定義されているものです。 注意すべき点は、クラスで定義されているプライベート要素は集約されていない ということです。それらは $ws オブジェクトの中には 存在します。また、WDDXStorage のコンストラクタも 存在しません。これを集約するのは論理的ではありません。

```
$fs を WDDXStorage クラスに集約します
$fs オブジェクト
クラス: filestorage
プロパティ: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
プロパティ: version = 1.0
メソッド: filestorage
メソッド: write
メソッド: store
```

集約処理は、積み重ねていくことが可能です。そこで今度は $fs と DBStorage を 集約します。これにより、定義されているすべてのクラスの 保存処理メソッドを使用可能なオブジェクトができあがります。

```
それを DBStorage クラスに集約します
$fs オブジェクト
クラス: filestorage
プロパティ: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
プロパティ: version = 1.0
プロパティ: dbtype = mysql
メソッド: filestorage
メソッド: write
メソッド: store
メソッド: save
```

最後に、プロパティやメソッドを動的に集約したのと同じ方法で、 集約したプロパティやメソッドを解除することも可能です。 $fs から WDDXStorage クラスの集約を解除すると、このような結果になります。

```
WDDXStorage を集約から解除します
$fs オブジェクト
クラス: filestorage
プロパティ: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
プロパティ: dbtype = mysql
メソッド: filestorage
メソッド: write
メソッド: save
```

上で説明しきれなかったことがひとつあります。それは、集約処理の際には 既存のプロパティやメソッドは上書きされないということです。例えば、FileStorage クラスでは $data というプロパティを定義しており、 WDDXStorage クラスでも同名のプロパティが 定義されていますが、このプロパティが FileStorage のインスタンス化の際に 取得したプロパティの内容を上書きすることはありません。

---

aggregate_info

(No version information available, might be only in CVS)

aggregate_info — 指定したオブジェクトの集約情報を取得する

説明

array **aggregate_info** (object $object)

object で指定したオブジェクトについての 集約情報を取得します。

パラメータ

object

返り値

指定したオブジェクトに関する集約の情報を返します。返される値は、 メソッドとプロパティの配列を要素とする連想配列形式となります。 連想配列のキーは、集約されたクラス名となります。

例

Example#1 aggregate_info() の使用例

```
<?php

class Slicer {
     var $vegetable;

     function Slicer($vegetable)
     {
         $this->vegetable = $vegetable;
     }

     function slice_it($num_cuts)
     {
          echo "Doing some simple slicing¥n";
          for ($i=0; $i < $num_cuts; $i++) {
             // 何らかのスライス処理
          }
     }
}

class Dicer {
     var $vegetable;
     var $rotation_angle = 90;    // 角度

     function Dicer($vegetable)
     {
         $this->vegetable = $vegetable;
     }

     function dice_it($num_cuts)
     {
         echo "最初のカットを行います\n";
          for ($i=0; $i < $num_cuts; $i++) {
             // 何らかのカット処理
          }
          $this->rotate($this->rotation_angle);
          echo "別の向きのカットを行います\n";
          for ($i=0; $i < $num_cuts; $i++) {
             // さらに何らかのカット処理
          }
     }

     function rotate($deg)
     {
          echo "{$this->vegetable} を {$deg} 度回転します\n";
     }

     function _secret_super_dicing($num_cuts)
     {
        // 企業秘密なので、ナイショです (^o^)
     }
}

$obj = new Slicer('タマネギ');
aggregate($obj, 'Dicer');
print_r(aggregate_info($obj));
?>
```

上の例の出力は以下となります。

```
Array
(
     [dicer] => Array
         (
               [methods] => Array
                  (
                       [0] => dice_it
```

```
                    [1] => rotate
                )

            [properties] => Array
                (
                    [0] => rotation_angle
                )

        )

)
```

ごらんのとおり、Dicer クラスのすべての プロパティとメソッドが新しいオブジェクトに集約されました。 ただし、クラスのコンストラクタと **_secret_super_dicing** メソッドは集約されていません。

参考

- aggregate()
- aggregate_methods()
- aggregate_methods_by_list()
- aggregate_methods_by_regexp()
- aggregate_properties()
- aggregate_properties_by_list()
- aggregate_properties_by_regexp()
- deaggregate()

---

aggregate_methods_by_list

(PHP 4 >= 4.2.0)

aggregate_methods_by_list — 選択したクラスメソッドを、動的にオブジェクトに集約する

説明

void **aggregate_methods_by_list** (object $object , string $class_name , array $methods_list [, bool $exclude])

メソッド名のリストを使用して、クラスのメソッドを既存のオブジェクトに集約します。

クラスのコンストラクタ、およびアンダースコア文字 (_) で始まる名前のメソッド (プライベートとみなされます) は、常に対象となりません。

パラメータ

object

class_name

methods_list

exclude

オプションのパラメータ exclude で、リストに含まれているメソッドを集約する (exclude が **FALSE** の場合。これがデフォルトです) のか含まれていないメソッドを集約する (exclude が **TRUE**) のかを指定します。

返り値

値を返しません。

参考

- aggregate()
- aggregate_info()
- aggregate_methods()
- aggregate_methods_by_regexp()
- aggregate_properties()
- aggregate_properties_by_list()
- aggregate_properties_by_regexp()
- deaggregate()

---

aggregate_methods_by_regexp

(PHP 4 >= 4.2.0)

aggregate_methods_by_regexp — 正規表現を使用して選択したクラスメソッドを、 動的にオブジェクトに集約する

説明

void ***aggregate_methods_by_regexp*** *(object $object , string $class_name , string $regexp [, bool $exclude])*

クラスのメソッドを既存のオブジェクトに集約します。 集約するメソッドを決定するために正規表現を使用します。

クラスのコンストラクタ、およびアンダースコア文字（_）で始まる名前の メソッド（プライベートとみなされます）は、常に対象となりません。

パラメータ

object

class_name

regexp

exclude

オプションのパラメータ *exclude* で、正規表現にマッチするメソッドを集約する（*exclude* が **FALSE** の場合。これがデフォルトです）のかマッチしないメソッドを集約する（*exclude* が **TRUE**）のかを指定します。

返り値

値を返しません。

参考

- aggregate()
- aggregate_info()
- aggregate_methods()
- aggregate_methods_by_list()
- aggregate_properties()
- aggregate_properties_by_list()
- aggregate_properties_by_regexp()
- deaggregate()

---

aggregate_methods

(PHP 4 >= 4.2.0)

aggregate_methods — クラスのメソッドを、動的にオブジェクトに集約する

説明

void ***aggregate_methods*** *(object $object , string $class_name)*

クラスで定義されているすべてのメソッドを既存のオブジェクトに集約します。 ただし、クラスのコンストラクタは例外です。 また名前がアンダースコア文字（_）で始まるメソッドも、 プライベートとみなされて例外となります。

パラメータ

object

class_name

返り値

値を返しません。

参考

- aggregate()
- aggregate_info()
- aggregate_methods_by_list()
- aggregate_methods_by_regexp()
- aggregate_properties()
- aggregate_properties_by_list()
- aggregate_properties_by_regexp()
- deaggregate()

---

aggregate_properties_by_list

(PHP 4 >= 4.2.0)

aggregate_properties_by_list — 選択したクラスプロパティを、動的にオブジェクトに集約する

説明

void ***aggregate_properties_by_list*** *(object $object , string $class_name , array $properties_list [, bool $exclude])*

プロパティ名のリストを使用して、クラスのプロパティを既存のオブジェクトに集約します。

アンダースコア文字（_）で始まる名前のプロパティ（プライベートとみなされます）は、常に対象となりません。

パラメータ

object

class_name

properties_list

exclude

オプションのパラメータ *exclude* で、リストに含まれているプロパティを集約する（*exclude* が **FALSE** の場合。これがデフォルトです）のか含まれていないプロパティを集約する（*exclude* が **TRUE**）のかを指定します。

返り値

値を返しません。

参考

- aggregate()
- aggregate_methods()
- aggregate_methods_by_list()
- aggregate_methods_by_regexp()
- aggregate_properties()
- aggregate_properties_by_regexp()
- aggregate_info()
- deaggregate()

---

aggregate_properties_by_regexp

(PHP 4 >= 4.2.0)

aggregate_properties_by_regexp — 正規表現を使用して選択したクラスプロパティを、 動的にオブジェクトに集約する

説明

void **aggregate_properties_by_regexp** (object $object , string $class_name , string $regexp [, bool $exclude])

クラスのプロパティを既存のオブジェクトに集約します。集約するプロパティを 決定するために正規表現を使用します。

アンダースコア文字（_）で始まる名前のプロパティ（プライベートとみなされます）は、常に対象となりません。

パラメータ

object

class_name

regexp

exclude

オプションのパラメータ *exclude* で、正規表現にマッチするプロパティを集約する（*exclude* が **FALSE** の場合。これがデフォルトです）のかマッチしないプロパティを集約する（*exclude* が **TRUE**）のかを指定します。

返り値

値を返しません。

参考

- aggregate()
- aggregate_methods()
- aggregate_methods_by_list()
- aggregate_methods_by_regexp()
- aggregate_properties()
- aggregate_properties_by_list()
- aggregate_info()
- deaggregate()

---

aggregate_properties

(PHP 4 >= 4.2.0)

aggregate_properties — *クラスのプロパティを、動的にオブジェクトに集約する*

説明

void ***aggregate_properties*** *(object $object , string $class_name)*

クラスで定義されているすべてのプロパティを既存のオブジェクトに 集約します。ただし、名前がアンダースコア文字 (_) で始まる プロパティは、プライベートとみなされて例外となります。

パラメータ

object

class_name

返り値

値を返しません。

参考

- *aggregate()*
- *aggregate_methods()*
- *aggregate_methods_by_list()*
- *aggregate_methods_by_regexp()*
- ***aggregate_properties()***
- *aggregate_properties_by_list()*
- *aggregate_properties_by_regexp()*
- *deaggregate()*

---

aggregate

(PHP 4 >= 4.2.0)

aggregate — *メソッドおよびプロパティの動的なクラス/オブジェクト集約を行う*

説明

void ***aggregate*** *(object $object , string $class_name)*

既存のオブジェクトにあるクラスで定義されたメソッドとプロパティを集約します。 アンダースコア文字 (_) で始まるメソッドとプロパティは、 集約されたクラスではプライベートとみなされ、使用されません。 コンストラクタも集約処理から除外されます。

パラメータ

object

class_name

返り値

値を返しません。

参考

- *aggregate_info()*
- *aggregate_methods()*
- *aggregate_methods_by_list()*
- *aggregate_methods_by_regexp()*
- *aggregate_properties()*
- *aggregate_properties_by_list()*
- *aggregate_properties_by_regexp()*
- *deaggregate()*

---

aggregation_info

(PHP 4 >= 4.2.0)

aggregation_info — *aggregate_info()* *のエイリアス*

説明

この関数は次の関数のエイリアスです。 aggregate_info().

---

deaggregate

(PHP 4 >= 4.2.0)

deaggregate — 集約されたメソッドやプロパティをオブジェクトから取り除く

説明

void **deaggregate** (object $object [, string $class_name])

オブジェクトに集約されたクラスのメソッドやプロパティを取り除きます。

パラメータ

object

class_name

オプションのパラメータ class_name が指定された場合は そのクラスで定義されているメソッドやプロパティのみが削除され、 それ以外の場合は集約されたすべてのメソッドやプロパティが削除されます。

返り値

値を返しません。

参考

- aggregate()
- aggregate_methods()
- aggregate_methods_by_list()
- aggregate_methods_by_regexp()
- aggregate_properties()
- aggregate_properties_by_list()
- aggregate_properties_by_regexp()
- **deaggregate()**

目次

オブジェクトプロパティとメソッドコールのオーバーロード

導入

この拡張モジュールの用途は、オブジェクトのプロパティへのアクセスと メソッドのコールのオーバーロードを可能にすることです。この 拡張モジュールで定義されている関数は 1 つだけです。この関数、 overload() は、この機能を有効にするクラスの名前を 引数とします。名前を指定されたクラスでこの機能を使用したい場合は 以下の適当なメソッドを定義する必要があります。これらは、 __get()、__set()、 __call() で、それぞれプロパティを取得、 プロパティを設定、メソッドをコールするためのものです。 オーバーロード機能は選択可能です。これらのハンドラ関数の中でオーバーロードを無効とすることができ、 この場合、オブジェクトのプロパティに普通にアクセスできます。

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

警告

この拡張モジュールは PHP 5 には含まれていません。PHP 5 では __get()、__set() および __call() をネイティブにサポートしています。詳細は PHP 5 のオーバーロード のページを参照ください。

要件

外部ライブラリを必要としません。

インストール手順

以下の関数を使用するには、オプション --enable-overload を指定して PHP を コンパイルする必要があります。この拡張モジュールは、 PHP

4.3.0 ではデフォルトで有効になっています。 --disable--overload により オーバーロードのサポートを無効とすることができます。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

注意: オーバーロードの組み込みサポートは PHP 4.3.0 で利用可能となりました。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

例

overload() 関数の簡単な使用例をいくつか示します。

Example#1 PHP クラスのオーバーロード

```
<?php

class OO {
    var $a = 111;
    var $elem = array('b' => 9, 'c' => 42);

    // プロパティを取得するためのコールバックメソッド
    function __get($prop_name, &$prop_value)
    {
        if (isset($this->elem[$prop_name])) {
            $prop_value = $this->elem[$prop_name];
            return true;
        } else {
            return false;
        }
    }

    // プロパティを設定するためのコールバックメソッド
    function __set($prop_name, $prop_value)
    {
        $this->elem[$prop_name] = $prop_value;
        return true;
    }
}

// OO オブジェクトをオーバーロードします。
overload('OO');

$o = new OO;
echo "¥$o->a: $o->a¥n"; // 出力: $o->a: 111
echo "¥$o->b: $o->b¥n"; // 出力: $o->b: 9
echo "¥$o->c: $o->c¥n"; // 出力: $o->c: 42
echo "¥$o->d: $o->d¥n"; // 出力: $o->d:

// OO の $elem 配列に新規アイテムを追加します。
$o->x = 56;

// (PHP 4 に組み込まれている) stdclass のインスタンスを生成します。
// $val はオーバーロードされていません!
$val = new stdclass;
$val->prop = 555;

// $val オブジェクトを有する配列として "a" を設定します。
// しかし、__set() はこれを $elem 配列に代入します。
$o->a = array($val);
var_dump($o->a[0]->prop);

?>
```

---

overload

(PHP 4 >= 4.2.0)

overload — クラスのプロパティおよびメソッドに関してオーバーロードを可能にする

説明

void **overload** (string $class_name)

overload() 関数は、 class_name で指定されたクラスのプロパティと メソッドコールをオーバーロードします。

パラメータ

class_name

オーバーロードするクラス名。

返り値

値を返しません。

例

このパートの導入部にある 例 を参照ください。

Oracle 関数

導入

これらの関数は Oracle コールインターフェース (OCI) を使用した Oracle 10, Oracle 9, Oracle 8, そして Oracle 7 データベースへのアクセスを可能にします。 これらは PHP 変数の Oracle プレースホルダへのバインドをサポートし、 LOB、FILE、ROWID を完全にサポートしており、 ユーザー定義の変数が使用可能です。

要件

この拡張を使用するために Oracle クライアントライブラリが必要になります。 Windows ユーザは php_oci8.dll を使用するために 少なくともバージョン 10 以降のライブラリが必要になるでしょう。

> **注意**: この拡張モジュールは、Oracle 8 のクライアントライブラリをサポートしません。 とはいえ、バージョン 9 以降のクライアントライブラリが Oracle 8 サーバへの接続をサポートする限りは Oracle 8 サーバに接続することが可能です。

要求される全てのファイルをインストールする最も簡便な方法は、 Oracle Instant Client を使用することです。これは » http://www.oracle.com/technology/tech/oci/instantclient/instantclient.html から取得可能です。 OCI8 モジュールを動作させるには、Oracle Instant Client の 「基本 (basic)」バージョンを導入するだけで十分です。 Instant Client は ORACLE_SID もしくは ORACLE_HOME 環境変数を設定する必要がありませんが、LD_LIBRARY_PATH と NLS_LANG を設定する必要があります。

この拡張モジュールを使用する前に Web デーモンのユーザでもある Oracle ユーザに対する Oracle 用環境変数が正しく設定されていることを 確認してください。これらの変数は Web サーバを起動する 前に 設定されていなければなりません。 設定されている必要がある変数を以下に示します。

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG

頻繁にはないですが、TNS_ADMIN, TWO_TASK, ORA_TZFILE、 そして ORA_NLS33, ORA_NLS10 あるいは NLS_* のような様々な Oracle の国際化設定用の変数を使用する場合は、 Oracle のドキュメントを参照してください。

Web サーバーのユーザ用に環境変数を設定した後、Web サーバーのユーザ (nobody, www) をグループ oracle に追加してください。

> **注意**: **Web サーバが起動しないか、起動時にクラッシュする場合** Apache が pthread ライブラリにリンクされているかどうか 次のように確認してください。
>
> ```
> #  ldd  /www/apache/bin/httpd
>       libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
>       libm.so.6 => /lib/libm.so.6 (0x4002f000)
>       libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
>       libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
>       libc.so.6 => /lib/libc.so.6 (0x4007e000)
>       /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
> ```
>
> libpthread がこの一覧にない場合、Apache を再インストールする必要があります。
>
> ```
> #  cd  /usr/src/apache_1.3.xx
> #  make  clean
> #  LIBS=-lpthread  ./config.status
> #  make
> #  make  install
> ```
>
> UnixWare のようないくつかのシステムでは、libpthread の代わりに libthread が使用されています。その場合、PHP と Apache は、 EXTRA_LIBS=-lthread を configure に指定する必要があります。

実行時設定

php.ini の設定により動作が変化します。

OCI8 設定オプション

名称	デフォルト	変更可否	変更履歴
oci8.privileged_connect	"0"	PHP_INI_SYSTEM	PHP 5.1.2 以降で使用可能
oci8.max_persistent	"-1"	PHP_INI_SYSTEM	PHP 5.1.2 以降で使用可能
oci8.persistent_timeout	"-1"	PHP_INI_SYSTEM	PHP 5.1.2 以降で使用可能

名称	デフォルト	変更可否	変更履歴
oci8.ping_interval	"60"	PHP_INI_SYSTEM	PHP 5.1.2 以降で使用可能
oci8.statement_cache_size	"20"	PHP_INI_SYSTEM	PHP 5.1.2 以降で使用可能
oci8.default_prefetch	"10"	PHP_INI_SYSTEM	PHP 5.1.2 以降で使用可能
oci8.old_oci_close_semantics	"0"	PHP_INI_SYSTEM	PHP 5.1.2 以降で使用可能

以下に設定ディレクティブに関する簡単な説明を示します。

oci8.privileged_connect boolean

このオプションは外部の信用 (**OCI_SYSOPER**, **OCI_SYSDBA**) を利用して権限付きの接続を有効にします。

oci8.max_persistent int

プロセスあたりの永続的な OCI8 接続の最大値を指定します。 このオプションを -1 に設定することは、制限なしを意味します。

oci8.persistent_timeout int

与えられたプロセスがアイドル状態の永続的接続を維持する最大時間（秒単位）を指定します。 このオプションを -1 に設定することは、 アイドル状態の永続的接続は永久に維持されることを意味します。

oci8.ping_interval int

oci_pconnect() の間、ping を発行するまでに経過させる時間（秒単位）を指定します。 0 に設定した場合、永続的接続は再利用される度に ping を発行します。 ping を完全に無効にするためには、このオプションを -1 に設定します。

> **注意**: ping を無効にすることで oci_pconnect() は最高の効率で処理をコールしますが、ネットワークが分断された場合や PHP が接続した後に Oracle サーバがダウンし、 その後に実行されるスクリプト中において PHP が接続の失敗を検知しなくなります。 詳細な情報は oci_pconnect() を参照ください。

oci8.statement_cache_size int

このオプションはステートメントキャッシュを有効にします。 また、キャッシュするステートメントの数を指定します。 ステートメントキャッシュを無効にする場合、このオプションを 0 に設定してください。

> **注意**: より大きなキャッシュは、メモリ使用量の増加と引き替えに パフォーマンスの改善をもたらします。

oci8.default_prefetch int

このオプションはステートメントのプリフェッチを有効にし、 ステートメントの実行後自動的にフェッチされるデフォルトの行数を 設定します。

> **注意**: より大きなプリフェッチは、メモリ使用量の増加と引き替えに パフォーマンスの改善をもたらします。

oci8.old_oci_close_semantics boolean

このオプションは oci_close() の動作を制御します。有効にすると、oci_close() は何も行いません。接続はスクリプトの終了まで閉じられません。 これは後方互換性のためのみに存在しています。 この設定を有効にする必要があると判明した場合、 このオプションを有効にする代わりに、oci_close() をアプリケーションから削除することが 強く推奨されます。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

OCI_DEFAULT (integer)
文の実行モードを指定します。 このモードを使用する場合、 文は自動的にコミットされません。
OCI_DESCRIBE_ONLY (integer)
文の実行モードを指定します。 実際に文を実行したくないが取得一覧の記述は取得したい場合に このモードを使用してください。
OCI_COMMIT_ON_SUCCESS (integer)
文の実行モードを指定します。 文は、oci_execute() コールの後に自動的にコミットされます。
OCI_EXACT_FETCH (integer)
文の取得モードを指定します。 アプリケーションがあらかじめ何行取得すればよいか分かっている場合に 使用されます。 このモードは Oracle リリース 8 以降ではプリフェッチ機能をオフにします。 カーソルは希望する行を取得した後キャンセルされ、 サーバ側のリソースの使用は軽減されます。
OCI_SYSDATE (integer)
OCI_B_BFILE (integer)
oci_bind_by_name() で BFILE をバインドする場合に使用されます。
OCI_B_CFILEE (integer)
oci_bind_by_name() で CFILE をバインドする場合に使用されます。
OCI_B_CLOB (integer)
oci_bind_by_name() で CLOB をバインドする場合に使用されます。
OCI_B_BLOB (integer)
oci_bind_by_name() で BLOB をバインドする場合に使用されます。
OCI_B_ROWID (integer)
oci_bind_by_name() で ROWID をバインドする場合に使用されます。
OCI_B_CURSOR (integer)
oci_bind_by_name() で oci_new_descriptor() によってあらかじめ割り当てられたカーソルをバインドする場合に使用されます。
OCI_B_NTY (integer)
oci_bind_by_name() で 名前付けされたデータ型をバインドする場合に使用されます。 注意: PHP < 5.0 では **OCI_B_SQLT_NTY** と呼ばれます。
OCI_B_BIN (integer)
SQLT_BFILEE (integer)
OCI_B_BFILE と等価です。
SQLT_CFILEE (integer)
OCI_B_CFILEE と等価です。
SQLT_CLOB (integer)
OCI_B_CLOB と等価です。
SQLT_BLOB (integer)
OCI_B_BLOB と等価です。
SQLT_RDD (integer)
OCI_B_ROWID と等価です。
SQLT_NTY (integer)
OCI_B_NTY と等価です。
SQLT_LNG (integer)

oci_bind_by_name() で LONG 値をバインドする際に使用されます。
SQLT_LBI (integer)
oci_bind_by_name() で LONG RAW 値をバインドする際に使用されます。
SQLT_BIN (integer)
oci_bind_by_name() で RAW 値をバインドする際に使用されます。
SQLT_NUM (integer)
oci_bind_array_by_name() で NUMBER の配列をバインドする場合に使用されます。
SQLT_INT (integer)
oci_bind_array_by_name() で INTEGER の配列をバインドする場合に使用されます。
SQLT_AFC (integer)
oci_bind_array_by_name() で CHAR の配列をバインドする場合に使用されます。
SQLT_CHR (integer)
oci_bind_array_by_name() で VARCHAR2 の配列をバインドする場合に使用されます。 oci_bind_by_name() でも使用されます。
SQLT_VCS (integer)
oci_bind_array_by_name() で VARCHAR の配列をバインドする際に使用されます。
SQLT_AVC (integer)
oci_bind_array_by_name() で CHARZ の配列をバインドする場合に使用されます。
SQLT_STR (integer)
oci_bind_array_by_name() で STRING の配列をバインドする場合に使用されます。
SQLT_LVC (integer)
oci_bind_array_by_name() で LONG VARCHAR の配列をバインドする場合に使用されます。
SQLT_FLT (integer)
oci_bind_array_by_name() で FLOAT の配列をバインドする場合に使用されます。
SQLT_ODT (integer)
oci_bind_array_by_name() で LONG の配列をバインドする場合に使用されます。
SQLT_BDOUBLE (integer)
SQLT_BFLOAT (integer)
OCI_FETCHSTATEMENT_BY_COLUMN (integer)
oci_fetch_all() のデフォルトのモードです。
OCI_FETCHSTATEMENT_BY_ROW (integer)
oci_fetch_all() でのもうひとつのモードです。
OCI_ASSOC (integer)
oci_fetch_all() と oci_fetch_array() で結果を連想配列で取得するために使用されます。
OCI_NUM (integer)
oci_fetch_all() と oci_fetch_array() で結果を配列で取得するために使用されます。
OCI_BOTH (integer)
oci_fetch_all() と oci_fetch_array() で結果を配列と連想配列の両方で取得するために使用されます。
OCI_RETURN_NULLS (integer)
oci_fetch_array() でフィールド値が **NULL** の場合に空の配列要素を取得するために使用されます。
OCI_RETURN_LOBS (integer)
oci_fetch_array() でディスクリプタの代わりに LOB の値を取得するために使用されます。
OCI_DTYPE_FILE (integer)
このフラグは oci_new_descriptor() に新しい FILE ディスクリプタを初期化するように伝えます。
OCI_DTYPE_LOB (integer)
このフラグは oci_new_descriptor() に新しい LOB ディスクリプタを初期化するように伝えます。
OCI_DTYPE_ROWID (integer)
このフラグは oci_new_descriptor() に新しい ROWID ディスクリプタを初期化するように伝えます。
OCI_D_FILE (integer)
OCI_DTYPE_FILE と等価です。
OCI_D_LOB (integer)
OCI_DTYPE_LOB と等価です。
OCI_D_ROWID (integer)
OCI_DTYPE_ROWID と等価です。
OCI_SYSOPER (integer)
外部の信任を使用する SYSOPER として接続するために oci_connect() と併用します（これには oci8.privileged_connect を有効にすべきです）。
OCI_SYSDBA (integer)
外部の信任を使用する SYSDBA として接続するために oci_connect() と併用します（これには oci8.privileged_connect を有効にすべきです）。
OCI_LOB_BUFFER_FREE (integer)
使用されたバッファを解放するために OCI-Lob->flush と併用します。
OCI_TEMP_CLOB (integer)
一時的な CLOB が生成されるよう明示的に指定するため OCI-Lob->writeTemporary と併用します。
OCI_TEMP_BLOB (integer)
一時的な BLOB が生成されるよう明示的に指定するため OCI-Lob->writeTemporary と併用します。

例

Example#1 基本的なクエリ

```
<?php

  $conn = oci_connect('hr', 'hr', 'orcl');
  if (!$conn) {
    $e = oci_error();
    print htmlentities($e['message']);
    exit;
  }

  $query = 'SELECT * FROM DEPARTMENTS';

  $stid = oci_parse($conn, $query);
  if (!$stid) {
    $e = oci_error($conn);
    print htmlentities($e['message']);
    exit;
  }

  $r = oci_execute($stid, OCI_DEFAULT);
  if (!$r) {
    $e = oci_error($stid);
    echo htmlentities($e['message']);
    exit;
  }

  print '<table  border="1">';
  while ($row = oci_fetch_array($stid, OCI_RETURN_NULLS)) {
    print '<tr>';
        foreach ($row as $item) {
          print '<td>'.($item?htmlentities($item):' ').'</td>';
```

```
        }
        print '</tr>';
    }
    print '</table>';

    oci_close($conn);
?>
```

Example#2 バインド変数を用いた挿入

```
<?php

  // 実行前に表を作成する
   //     CREATE TABLE MYTABLE (mid NUMBER, myd VARCHAR2(20));

  $conn = oci_connect('scott', 'tiger', 'orcl');

  $query = 'INSERT INTO MYTABLE VALUES(:myid, :mydata)';

  $stid = oci_parse($conn, $query);

  $id = 60;
  $data = 'Some data';

  oci_bind_by_name($stid, ':myid', $id);
  oci_bind_by_name($stid, ':mydata', $data);

  $r = oci_execute($stid);

  if ($r)
    print "One row inserted";

  oci_close($conn);

?>
```

Example#3 CLOB カラムにデータを挿入する

```
<?php

// 実行前に表を作成する
//       CREATE TABLE MYTABLE (mykey NUMBER, myclob CLOB);

$conn = oci_connect('scott', 'tiger', 'orcl');

$mykey = 12343;  // この例で用いる任意のキー

$sql = "INSERT INTO mytable (mykey, myclob)
          VALUES (:mykey, EMPTY_CLOB())
          RETURNING myclob INTO :myclob";

$stid = oci_parse($conn, $sql);
$clob = oci_new_descriptor($conn, OCI_D_LOB);
oci_bind_by_name($stid, ":mykey", $mykey, 5);
oci_bind_by_name($stid, ":myclob", $clob, -1, OCI_B_CLOB);
oci_execute($stid, OCI_DEFAULT);
$clob->save("A very long string");

oci_commit($conn);

// CLOB データをフェッチする

$query = 'SELECT myclob FROM mytable WHERE mykey = :mykey';

$stid = oci_parse ($conn, $query);
oci_bind_by_name($stid, ":mykey", $mykey, 5);
oci_execute($stid, OCI_DEFAULT);

print '<table border="1">';
while ($row = oci_fetch_array($stid, OCI_ASSOC)) {
  $result = $row['MYCLOB']->load();
  print '<tr><td>'.$result.'</td></tr>';
}
print '</table>';

?>
```

コマンドラインで実行するのと同様な手法により、ストアドプロシージャ に簡単にアクセス可能です。

Example#4 ストアドプロシージャの使用法

```
<?php
// by webmaster at remoterealty dot com
$sth = oci_parse($dbh, "begin sp_newaddress( :address_id, '$firstname',
  '$lastname', '$company', '$address1', '$address2', '$city', '$state',
  '$postalcode', '$country', :error_code );end;");

// この命令は、:address_id を入出力変数、:error_code を出力変数として
// ストアドプロシージャ sp_newaddress をコールします。
// 続いて、以下のようにバインドを実行します。

   oci_bind_by_name($sth, ":address_id", $addr_id, 10);
   oci_bind_by_name($sth, ":error_code", $errorcode, 10);
   oci_execute($sth);

?>
```

接続のハンドリング

oci8 拡張モジュールは Oracle に接続するための 3 つの異なる関数を提供しています。 アプリケーションに最適な関数を使用するのはあなた次第

です。 また、このセクションにある情報は、 インフォームド・チョイス（十分な説明を受けよく考えた上での選択）を行う助けになることを目的としています。

Oracle サーバへの接続は、完了まで要する時間という点から見ると、 かなりコストのかかる操作です。oci_pconnect() 関数は、 異なるスクリプトリクエスト間で接続の再利用が可能な 持続的キャッシュを使用します。 これは、PHP プロセス（もしくは Apache の子プロセス）毎の接続に関するオーバーヘッドを一度のみ負うということを意味しています。

もしアプリケーションが信用された異なる Web ユーザー毎に Oracle に接続する場合、oci_pconnect() による持続的キャッシュは、 同時ユーザー数の増加と共に有効ではなくなるでしょう。 これは、多くのアイドル状態の接続が維持されることが原因で、 Oracle サーバ全体のパフォーマンスに不利な影響を与え始めるためです。 もしアプリケーションがこの方法で構成されている場合、oci8.max_persistent や oci8.persistent_timeout（持続的接続のキャッシュサイズや生存期間の制御が可能になります）を使用してアプリケーションをチューニングする、もしくは代わりに oci_connect() を使用することが推奨されます。

oci_connect() と oci_pconnect() の両者とも接続キャッシュを使用します。もし、同一パラメータと共に oci_connect() を複数回コールする場合、 2 番目以降は既存の接続ハンドルを返します。oci_connect() によって使用されるキャッシュは、スクリプト実行終了時、 もしくは明示的に接続ハンドルを閉じた時にクリアされます。 oci_pconnect() も同様の動作をしますが、 キャッシュは独立して維持され、リクエスト間で残存します。

このキャッシュ機能は忘れてはならないほど重要です。 それは、2 つのハンドルがトランザクション的に独立していない（実際には同じ接続なので、どのような種類の独立もありません）ためです。もしアプリケーションが 2 つの別々でトランザクション的に独立した接続を必要とする場合、oci_new_connect() を使用すべきです。

oci_new_connect() は、他の既存の接続が存在したとしても 常に Oracle サーバへの新規接続を生成します。 特にアプリケーションの最も負荷が高い部分など、 高トラフィックな Web アプリケーションに対しては oci_new_connect() の使用を避けてください。

ドライバでサポートされるデータ型

oci_bind_by_name() 関数を使用してパラメータをバインドする場合、 ドライバは次の型をサポートします

型	マッピング
SQLT_NTY	oci_new_collection() によって生成されたような PHP のコレクションオブジェクトからネイティブのコレクション型に マップします
SQLT_BFILEE	oci_new_descriptor() によって生成されたような PHP のディスクリプタオブジェクトからネイティブのディスクリプタ型に マップします
SQLT_CFILEE	oci_new_descriptor() によって生成されたような PHP のディスクリプタオブジェクトからネイティブのディスクリプタ型に マップします
SQLT_CLOB	oci_new_descriptor() によって生成されたような PHP のディスクリプタオブジェクトからネイティブのディスクリプタ型に マップします
SQLT_BLOB	oci_new_descriptor() によって生成されたような PHP のディスクリプタオブジェクトからネイティブのディスクリプタ型に マップします
SQLT_RDD	oci_new_descriptor() によって生成されたような PHP のディスクリプタオブジェクトからネイティブのディスクリプタ型に マップします
SQLT_NUM	PHP パラメータを 'C' の long 型に変換し、 その値をバインドします
SQLT_RSET	oci_parse() によって生成されたもしくは他の OCI クエリから処理されたような PHP のステートメントハンドルからネイティブのステートメントハンドルに マップします
SQLT_CHR や他の型	PHP パラメータを文字列型に変換し、その文字列をバインドします

以下の型は結果セットからカラムを処理する際にサポートされます

型	マッピング
SQLT_RSET	カーソルを表す OCI ステートメントリソースを生成します
SQLT_RDD	ROWID オブジェクトを生成します
SQLT_BLOB	LOB オブジェクトを生成します
SQLT_CLOB	LOB オブジェクトを生成します
SQLT_BFILE	LOB オブジェクトを生成します
SQLT_LNG	SQLT_CHR としてバインドし、文字列として返します
SQLT_LBI	SQLT_BIN としてバインドし、文字列として返します
Any other type	SQLT_CHR としてバインドし、文字列として返します

oci_bind_array_by_name

(PHP 5 >= 5.1.2, PECL oci8:1.2.0-1.2.4)

oci_bind_array_by_name — PHP の配列を Oracle PL/SQL の配列に名前でバインドする

説明

bool **oci_bind_array_by_name** (resource $statement , string $name , array &$var_array , int $max_table_length [, int $max_item_length [, int $type]])

oci_bind_array_by_name() は、PHP の配列 var_array を Oracle のプレースホルダ name にバインドします。 このプレースホルダは Oracle PL/SQL の配列を指しています。 入力変数あるいは出力変数のどちらとして使用されるのかは、 実行時に決められます。

パラメータ

statement

有効な OCI ステートメント識別子

name

Oracle のプレースホルダ

var_array

配列

max_table_length

入力配列および結果の配列の両方の最大長を設定する

max_item_length

配列要素の最大長を設定する。もし指定されない、もしくは -1 が指定された場合、**oci_bind_array_by_name()** は入力の配列の中で最も長い要素を探し、その長さを項目の最大長とする。

type

PL/SQL 配列の項目の型を指定するために利用される。 指定可能な型については以下を参照のこと。

- **SQLT_NUM** - NUMBER の配列
- **SQLT_INT** - INTEGER の配列 (注意: INTEGER は、実際には NUMBER(38) のシノニムだが、 **SQLT_NUM** ではこの場合うまく動作しない)。
- **SQLT_FLT** - FLOAT の配列
- **SQLT_AFC** - CHAR の配列
- **SQLT_CHR** - VARCHAR2 の配列
- **SQLT_VCS** - VARCHAR の配列
- **SQLT_AVC** - CHARZ の配列
- **SQLT_STR** - STRING の配列
- **SQLT_LVC** - LONG VARCHAR の配列
- **SQLT_ODT** - DATE の配列

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 oci_bind_array_by_name() の例

```php
<?php

$c = oci_connect("scott", "tiger");

$create = "CREATE TABLE bind_example(name VARCHAR(20))";
$statement = oci_parse($c, $create);
oci_execute($statement);

$create_pkg = "
CREATE OR REPLACE PACKAGE ARRAYBINDPKG1 AS
   TYPE ARRTYPE IS TABLE OF VARCHAR(20) INDEX BY BINARY_INTEGER;
   PROCEDURE iobind(c1 IN OUT ARRTYPE);
END ARRAYBINDPKG1;";
$statement = oci_parse($c, $create_pkg);
oci_execute($statement);

$create_pkg_body = "
CREATE OR REPLACE PACKAGE BODY ARRAYBINDPKG1 AS
   CURSOR CUR IS SELECT name FROM bind_example;
   PROCEDURE iobind(c1 IN OUT ARRTYPE) IS
     BEGIN
     FOR i IN 1..5 LOOP
        INSERT INTO bind_example VALUES (c1(i));
     END LOOP;
     IF NOT CUR%ISOPEN THEN
       OPEN CUR;
     END IF;
     FOR i IN REVERSE 1..5 LOOP
       FETCH CUR INTO c1(i);
       IF CUR%NOTFOUND THEN
         CLOSE CUR;
         EXIT;
       END IF;
     END LOOP;
   END iobind;
END ARRAYBINDPKG1;";
$statement = oci_parse($c, $create_pkg_body);
oci_execute($statement);

$statement = oci_parse($c, "BEGIN ARRAYBINDPKG1.iobind(:c1); END;");

$array = array("one", "two", "three", "four", "five");
```

```
oci_bind_array_by_name($statement, ":c1", $array, 5, -1, SQLT_CHR);

oci_execute($statement);

var_dump($array);

 ?>
```

注意

注意: この関数は、OCI8 リリース 1.2 から利用可能です。

---

oci_bind_by_name

(PHP 5, PECL oci8:1.1-1.2.4)

oci_bind_by_name — Oracle プレースホルダーに PHP 変数をバインドする

説明

bool **oci_bind_by_name** (resource $statement , string $ph_name , mixed &$variable [, int $maxlength [, int $type]])

ocibindbyname() は、PHP 変数 variable を Oracle プレースホルダー ph_name にバインドします。実行時に入力用、 出力用に使用されるによらず、必要な記憶領域が確保されます。

パラメータ

statement

OCI ステートメント

ph_name

プレースホルダー

variable

PHP 変数

maxlength

バインド時の最大長。-1 に設定した場合、 variable の現在の長さを最大長として設定する。

type

抽象データ型 (LOB/ROWID/BFILE) をバインドする必要がある場合、まず oci_new_descriptor() 関数を使用してこれを確保する必要がある。length は抽象データ型用には 使用されず、-1 を設定する必要がある。 type パラメータは、 使用されるディスクリプタをOracle に伝える。 指定可能な型については以下を参照のこと。

- **SQLT_FILE** - BFILE 用
- **SQLT_CFILE** - CFILE 用
- **SQLT_CLOB** - CLOB 用
- **SQLT_BLOB** - BLOB 用
- **SQLT_RDD** - ROWID 用
- **SQLT_NTY** - 名前付けされたデータ型用
- **SQLT_INT** - integer 用
- **SQLT_CHR** - VARCHAR 用
- **SQLT_BIN** - RAW カラム用
- **SQLT_LNG** - LONG カラム用
- **SQLT_LBI** - LONG RAW カラム用
- **SQLT_RSET** - カーソル用。 oci_new_cursor() により、前もって生成されたもの。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 oci_bind_by_name() の例

```
<?php
/* oci_bind_by_name  の例 thies at thieso dot net (980221)
  3レコードをempに挿入し、挿入の直後にレコードを更新するために
  ROWIDを使用する。
*/

$conn = oci_connect("scott", "tiger");

$stmt = oci_parse($conn, "
                            INSERT INTO
                                        emp (empno, ename)
```

```
                                VALUES
                                    (:empno,:ename)
                             RETURNING
                                   ROWID
                                 INTO
                                     :rid
                        ");

$data = array(
              1111 => "Larry",
              2222 => "Bill",
              3333 => "Jim"
             );

$rowid = oci_new_descriptor($conn, OCI_D_ROWID);

oci_bind_by_name($stmt, ":empno", $empno, 32);
oci_bind_by_name($stmt, ":ename", $ename, 32);
oci_bind_by_name($stmt, ":rid",   $rowid, -1, OCI_B_ROWID);

$update = oci_parse($conn, "
                             UPDATE
                                  emp
                               SET
                                  sal = :sal
                              WHERE
                                  ROWID = :rid
                           ");
oci_bind_by_name($update, ":rid", $rowid, -1, OCI_B_ROWID);
oci_bind_by_name($update, ":sal", $sal,   32);

$sal = 10000;

foreach ($data as $empno => $ename) {
    oci_execute($stmt);
    oci_execute($update);
}

$rowid->free();

oci_free_statement($update);
oci_free_statement($stmt);

$stmt = oci_parse($conn, "
                           SELECT
                                *
                             FROM
                                emp
                            WHERE
                                empno
                              IN
                                 (1111,2222,3333)
                        ");
oci_execute($stmt);

while ($row = oci_fetch_assoc($stmt)) {
    var_dump($row);
}

oci_free_statement($stmt);

/* テーブル emp から "junk" を削除する.... */
$stmt = oci_parse($conn, "
                           DELETE FROM
                                   emp
                                WHERE
                                   empno
                                     IN
                                        (1111,2222,3333)
                        ");
oci_execute($stmt);
oci_free_statement($stmt);

oci_close($conn);
?>
```

この関数は余分な空白を除去する事を忘れないでください。 次の例を見てください。

Example#2 oci_bind_by_name() の例

```
<?php
     $connection = oci_connect('apelsin','kanistra');
    $query = "INSERT INTO test_table VALUES(:id, :text)";

    $statement = oci_parse($query);
    oci_bind_by_name($statement, ":id", 1);
    oci_bind_by_name($statement, ":text", "trailing spaces follow      ");
    oci_execute($statement);
    /*
      このコードは DB に文字列 'trailing spaces follow' を挿入する。
     余分な空白は付加されない。
     */
?>
```

Example#3 oci_bind_by_name() の例

```
<?php
     $connection = oci_connect('apelsin','kanistra');
    $query = "INSERT INTO test_table VALUES(:id, 'trailing spaces follow        ')";

    $statement = oci_parse($query);
```

```
    oci_bind_by_name($statement, ":id", 1);
    oci_execute($statement);
    /*
      また、このコードは後方の空白を保持したまま 'trailing spaces follow       '
     を追加する。
    */
?>
```

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

magic_quotes_gpc や addslashes() と **oci_bind_by_name()** を同時に使用しないでください。 これは、引用符を追加する必要がないためで、また、magic quote により付加された引用符は、 **oci_bind_by_name()** が magic quote により付加された引用符と意図的に付加されたものを区別できないため、 そのままデータベースに書き込まれるためです。

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocibindbyname() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_bind_by_name()** への別名として残されていますが、 推奨されません。

---

oci_cancel

(PHP 5, PECL oci8:1.1-1.2.4)

oci_cancel — カーソルからの読み込みをキャンセルする

説明

bool **oci_cancel** (resource $statement)

カーソルを無効にして関連付けられた全てのリソースを開放し、 読み込みをキャンセルします。

パラメータ

statement

OCI ステートメント

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

oci_close

(PHP 5, PECL oci8:1.1-1.2.4)

oci_close — Oracleとの接続を閉じる

説明

bool **oci_close** (resource $connection)

Oracle との接続 connection を閉じます。

パラメータ

connection

Oracle 接続 ID。 oci_connect() によって返される。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: バージョン 1.1 から **oci_close()** は正確に Oracle 接続を閉じます。この関数の古い動作をさせる場合は oci8.old_oci_close_semantics オプションを使用してください。

OCI-Collection->append

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Collection->append — コレクションに要素を追加する

説明

OCI-Collection
bool **append** (mixed $value)

コレクションの最後に要素を追加します。

パラメータ

value

コレクションに追加する値を指定します。文字列か数値です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Collection->assign

OCI-Collection->assign

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Collection->assign — コレクションに他の存在するコレクションから値を割り当てる

説明

OCI-Collection
bool **assign** (OCI-Collection $from)

事前に生成された他のコレクションからコレクションの値を割り当てます。 両方のコレクションは、使用する前に oci_new_collection() を使って生成される必要があります。

パラメータ

from

OCI-Collection のインスタンスを指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Collection->append

OCI-Collection->assignElem

(No version information available, might be only in CVS)

OCI-Collection->assignElem — コレクションの要素に値を割り当てる

説明

OCI-Collection
bool **assignElem** (int $index , mixed $value)

インデックス index の要素に値を割り当てます。

パラメータ

index

要素のインデックス。最初のインデックスは 1 です。

value

文字列あるいは数値です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Collection->getElem

OCI-Collection->free

(No version information available, might be only in CVS)

OCI-Collection->free — *コレクションオブジェクトに関連付けられたリソースを解放する*

説明

OCI-Collection
bool **free** *(void)*

コレクションオブジェクトに関連付けられたリソースを解放します。

返り値

成功した場合に **TRUE** *を、失敗した場合に* **FALSE** *を返します。*

参考

- *oci_new_collection*

OCI-Collection->getElem

(No version information available, might be only in CVS)

OCI-Collection->getElem — *要素の値を返す*

説明

OCI-Collection
mixed **getElem** *(int $index)*

インデックス index (1 から始まる) を持つ要素の値を返します。

パラメータ

index

要素のインデックス。最初のインデックスは 1 です。

返り値

もし要素が存在しない場合 **FALSE** *、要素が* **NULL** *文字列なら* **NULL** *、要素が文字データ型なら文字列、 要素が数値フィールドなら数値を返します。*

参考

- *OCI-Collection->assignElem*

OCI-Collection->max

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Collection->max — *コレクション内の要素の最大数を取得する*

説明

OCI-Collection
int ***max*** *(void)*

コレクション内の要素の最大数を返します。

返り値

コレクション内の要素の最大数、あるいはエラー時に **FALSE** *を返します。*

返り値が 0 の場合、要素の数に制限はありません。

参考

- *OCI-Collection->size*

OCI-Collection->size

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Collection->size — *コレクションのサイズを返す*

説明

OCI-Collection
int **size** *(void)*

コレクションのサイズを返します。

返り値

コレクション内の要素の数、もしくはエラー時に **FALSE** を返します。

参考

- OCI-Collection->max

OCI-Collection->trim

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Collection->trim — コレクションの最後から要素を切り取る

説明

OCI-Collection
bool **trim** (int $num)

コレクションの最後から num 個の要素を切り取ります。

パラメータ

num

切り取る要素の数。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Collection->size

---

oci_commit

(PHP 5, PECL oci8:1.1-1.2.4)

oci_commit — 未解決の文をコミットする

説明

bool **oci_commit** (resource $connection)

Oracle接続connection 上のアクティブなトランザクションに関する全ての未解決の文をコミットします。

パラメータ

connection

Oracle 接続 ID。 oci_connect() もしくは oci_pconnect() によって返されたもの。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 oci_commit() の例

```
<?php
// Oracle サーバにログインする
$conn = oci_connect('scott', 'tiger');

// SQL をパースする
$stmt = oci_parse($conn, "
                          INSERT INTO
                                       employees (name, surname)
                               VALUES
                                       ('Maxim', 'Maletsky')
                        ");

/* 文を実行する。
    OCI_DEFAULT は oci_execute() に
   文をすぐにコミットしないように伝える */
oci_execute($stmt, OCI_DEFAULT);

/*
....
ここで他の文をパース、実行 ...
....
```

```
*/

// トランザクションをコミット
$committed = oci_commit($conn);

// コミットが成功したかどうかをテストする。もしエラーが発生していたら、エラーメッセージを返す
if (!$committed) {
    $error = oci_error($conn);
    echo 'Commit failed. Oracle reports: ' . $error['message'];
}

?>
```

注意

注意: トランザクションは接続を閉じたとき、 もしくはスクリプトの終了時のいずれの場合でも すぐに自動的にロールバックされます。 トランザクションをコミットするには **oci_commit()** をコールする、 もしくはトランザクションを破棄する場合は oci_rollback() を明示的にコールする必要があります。

参考

- oci_rollback()
- oci_execute()

---

oci_connect

(PHP 5, PECL oci8:1.1-1.2.4)

oci_connect — Oracle サーバへの接続を確立する

説明

resource **oci_connect** (string $username , string $password [, string $db [, string $charset [, int $session_mode]]])

他のほとんどの OCI コールで必要な接続 ID を返します。

パラメータ

username

Oracle ユーザ名

password

username に対するパスワード

db

このオプションパラメータには、ローカル Oracle インスタンスの名前か tnsnames.ora における接続先のエントリ名を指定することができる。

指定されない場合、PHP は接続先のデータベースを決定するために環境変数 **ORACLE_SID** と **TWO_TASK** を使用して、ローカル Oracle インスタンス名と tnsnames.ora の場所を適宜決定する。

charset

Oracle サーバのバージョン 9.2 以降を使用している場合、新規接続を確立する際に charset パラメータを指定することができます。 Oracleサーバ < 9.2 を使用している場合、このパラメータは無視され、かわりに環境変数 NLS_LANG が使用されます。

session_mode

このパラメータはバージョン 1.1 から利用可能で、 次の値を受け付ける: **OCI_DEFAULT**, **OCI_SYSOPER**, **OCI_SYSDBA** 。 **OCI_SYSOPER** もしくは **OCI_SYSDBA** のいずれかが指定された場合、 この関数は外部のクレデンシャルを利用して、 権限付きの接続を確立しようと試みる。 デフォルトでは権限付きの接続は無効である。有効にするためには、oci8.privileged_connect を On にする必要がある。

返り値

接続 ID、もしくはエラー時は **FALSE** を返す。

例

Example#1 oci_connect() の例

```
<?php
echo "<pre>";
$db = "";

$c1 = oci_connect("scott", "tiger", $db);
$c2 = oci_connect("scott", "tiger", $db);

function create_table($conn)
{
  $stmt = oci_parse($conn, "create table scott.hallo (test varchar2(64))");
  oci_execute($stmt);
  echo $conn . " created table¥n¥n";
}

function drop_table($conn)
{
  $stmt = oci_parse($conn, "drop table scott.hallo");
  oci_execute($stmt);
  echo $conn . " dropped table¥n¥n";
```

```
}

function insert_data($conn)
{
  $stmt = oci_parse($conn, "insert into scott.hallo
               values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " inserted hallo¥n¥n";
}

function delete_data($conn)
{
  $stmt = oci_parse($conn, "delete from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " deleted hallo¥n¥n";
}

function commit($conn)
{
  oci_commit($conn);
  echo $conn . " committed¥n¥n";
}

function rollback($conn)
{
  oci_rollback($conn);
  echo $conn . " rollback¥n¥n";
}

function select_data($conn)
{
  $stmt = oci_parse($conn, "select * from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn."----selecting¥n¥n";
  while (oci_fetch($stmt)) {
    echo $conn . " [" . oci_result($stmt, "TEST") . "]¥n¥n";
  }
  echo $conn . "----done¥n¥n";
}

create_table($c1);
insert_data($c1);   // c1 を使って行を挿入
insert_data($c2);   // c2 を使って行を挿入

select_data($c1);   // 両方の挿入した結果が返される
select_data($c2);

rollback($c1);      // c1 を使ってロールバック

select_data($c1);   // どちらの挿入もロールバックされている
select_data($c2);

insert_data($c2);   // c2 を使って行を挿入
commit($c2);        // c2 を使ってコミット

select_data($c1);   // c2 の結果が返される

delete_data($c1);   // c1 を使ってテーブル内の全ての行を削除
select_data($c1);   // 行は返されない
select_data($c2);   // 行は返されない
commit($c1);        // c1 を使ってコミット

select_data($c1);   // 行は返されない
select_data($c2);   // 行は返されない

drop_table($c1);
echo "</pre>";
?>
```

注意

注意: もし、Oracle インスタントクライアントとPHPを使用する場合、 次に示す簡単な接続ネーミングメソッドを使用することができます： » http://download-west.oracle.com/docs/cd/B12037_01/network.101/b10775/naming.htm#i498306。 基本的にこれはデータベース名として "//db_host[:port]/database_name" を指定できることを意味する。しかし、古いネーミング方法を使用したい場合、 **ORACLE_HOME** もしくは **TNS_ADMIN** のいずれかを設定する 必要があります 。

注意: 同じパラメータを使用して 2 回目やそれ以降に **oci_connect()** がコールされた場合、 最初のコールで返された接続ハンドルを返します。 これは 1 つのハンドルに対して発行されたクエリは、 他のハンドルにも適用されることを意味します。なぜなら、 これらは 同じ ハンドルだからです。 この動作は以下の例 1 で例示されています。 もしトランザクション的にお互い独立した 2 つのハンドルが必要な場合、 oci_new_connect() を使用してください。

注意: PHP 5.0.0 以前では、代わりに ocilogon() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_connect()** への別名として残されていますが、 推奨されません。

参考

- oci_pconnect()
- oci_new_connect()
- oci_close()

---

oci_define_by_name

(PHP 5, PECL oci8:1.1-1.2.4)

oci_define_by_name — SELECT 実行中、定義用の PHP 変数を使用する

説明

bool **oci_define_by_name** (resource $statement , string $column_name , mixed &$variable [, int $type])

SQL カラムカラムをフェッチするための PHP 変数を定義します。

パラメータ

statement

有効な OCI ステートメント ID

column_name

カラム名。大文字であること。

Oracle は、全ての大文字のカラム名を使用するが、 select の中で小文字も書くことが可能であることを考慮すること。 select 文にない変数を定義する場合、エラーは発生しない。

variable

PHP 変数

type

抽象 Datatype (LOB/ROWID/BFILE) を定義する必要がある場合、 まず oci_new_descriptor() 関数を用いてその 領域を確保する必要がある。 oci_bind_by_name() 関数も参照のこと。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 oci_define_by_name() の例

```
<?php
/* oci_define_by_name の例 - thies at thieso dot net (980219) */

$conn = oci_connect("scott", "tiger");

$stmt = oci_parse($conn, "SELECT empno, ename FROM emp");

/* 定義は oci_execute の前に行われなければなりません! */

oci_define_by_name($stmt, "EMPNO", $empno);
oci_define_by_name($stmt, "ENAME", $ename);

oci_execute($stmt);

while (oci_fetch($stmt)) {
     echo "empno:" . $empno . "¥n";
     echo "ename:" . $ename . "¥n";
}

oci_free_statement($stmt);
oci_close($conn);
?>
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocidefinebyname() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_define_by_name()** への別名として残されていますが、 推奨されません。

---

oci_error

(PHP 5, PECL oci8:1.1-1.2.4)

oci_error — 最後に見つかったエラーを返す

説明

array **oci_error** ([resource $source])

最後に見つかったエラーを返します。

パラメータ

source

ほとんどのエラーに対応するため、 パラメータは適当なリソースハンドルを指定可能。 oci_connect()、 oci_new_connect() あるいは oci_pconnect() での接続エラーの場合、 パラメータを渡さない。

返り値

もしエラーが見つからない場合、**oci_error()** は **FALSE** を返す。 **oci_error()** はエラーを連想配列として返す。 この配列には、Oracle エラーコード code や Oracle エラー文字列 message が含まれる。

変更履歴

バージョン	説明
4.3	エラーの場所と原因となった SQL テキストを示すため、 *offset* と *sqltext* も返される配列に含まれる。

例

Example#1 接続エラー後、Oracle エラーメッセージを表示する

```
$conn = @oci_connect("scott", "tiger", "mydb");
if (!$conn) {
   $e = oci_error();    // oci_connect のエラーの場合、ハンドルを渡さない
    echo htmlentities($e['message']);
}
```

Example#2 パースエラー後、Oracle エラーメッセージを表示する

```
$stmt = @oci_parse($conn, "select ' from dual");  // クオートが間違っている事に注意
if (!$stmt) {
   $e = oci_error($conn);  // oci_parse のエラーの場合、接続ハンドルを渡す
    echo htmlentities($e['message']);
}
```

Example#3 実行エラー後、 Oracle エラーメッセージと問題となった文を表示する

```
$r = oci_execute($stmt);
if (!$r) {
   $e = oci_error($stmt); // oci_execute のエラーの場合、ステートメントハンドルを渡す
    echo htmlentities($e['message']);
   echo "<pre>";
    echo htmlentities($e['sqltext']);
    printf("¥n%".($e['offset']+1)."s", "^");
   echo "</pre>";
}
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocierror() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_error()** への別名として残されていますが、 推奨されません。

---

oci_execute

(PHP 5, PECL oci8:1.1-1.2.4)

oci_execute — 文を実行する

説明

bool **oci_execute** (resource $statement [, int $mode])

事前にパースされた statement を実行します。

パラメータ

statement

有効な OCI ステートメント ID

mode

実行モードを指定することを可能にする (デフォルトは **OCI_COMMIT_ON_SUCCESS**)。

自動的にコミットされたくない場合は、 mode に **OCI_DEFAULT** に指定する。

OCI_DEFAULT モードを使用する場合、 トランザクションを生成する。 トランザクションは接続を閉じたとき、 もしくはスクリプトの終了時のいずれの場合でも すぐに自動的にロールバックされる。 トランザクションをコミットするには oci_commit() を、 トランザクションを破棄する場合は oci_rollback() をそれぞれ明示的にコールする必要がある。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ociexecute() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_execute()** への別名として残されていますが、 推奨されません。

参考

- oci_parse()

---

oci_fetch_all

(PHP 5, PECL oci8:1.1-1.2.4)

oci_fetch_all — 結果データの全ての行を配列に取得する

説明

int **oci_fetch_all** (resource $statement , array &$output [, int $skip [, int $maxrows [, int $flags]]])

全ての行の結果をユーザー定義の配列に格納して取得します。

oci8 ドライバによるデータ型マッピングの詳細については、ドライバがサポートするデータ型 を参照ください。

パラメータ

statement

有効な OCI ステートメント ID。

output

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

skip

結果を取得する際に無視する行数（デフォルトの値は 0 で、最初の行から開始されます）。

maxrows

読み込む行数。 skip 番目の行から開始されます（デフォルトは -1 で、全ての行 を意味します)。

flags

パラメータ flags には次の値の組み合わせが可能です。

- **OCI_FETCHSTATEMENT_BY_ROW**
- **OCI_FETCHSTATEMENT_BY_COLUMN** （デフォルト値）
- **OCI_NUM**
- **OCI_ASSOC**

返り値

取得した行数、失敗した場合 **FALSE** を返します。

例

Example#1 oci_fetch_all() の例

```
<?php
/* oci_fetch_all の例 mbritton at verinet dot com (990624) */

$conn = oci_connect("scott", "tiger");

$stmt = oci_parse($conn, "select * from emp");

oci_execute($stmt);

$nrows = oci_fetch_all($stmt, $results);
if ($nrows > 0) {
    echo "<table border=¥"1¥">¥n";
    echo "<tr>¥n";
    foreach ($results as $key => $val) {
       echo "<th>$key</th>¥n";
    }
    echo "</tr>¥n";

    for ($i = 0; $i < $nrows; $i++) {
       echo "<tr>¥n";
       foreach ($results as $data) {
          echo "<td>$data[$i]</td>¥n";
       }
       echo "</tr>¥n";
    }
    echo "</table>¥n";
} else {
    echo "No data found<br />¥n";
}
echo "$nrows Records Selected<br />¥n";

oci_free_statement($stmt);
oci_close($conn);
?>
```

注意

注意: PHP バージョン 5.0.0 以前では、代わりに ocifetchstatement() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_fetch_all()** への別名として残されていますが、 推奨されません。

---

oci_fetch_array

(PHP 5, PECL oci8:1.1-1.2.4)

oci_fetch_array — 結果データからの次の行を連想配列または配列、またはその両方で返す

説明

array **oci_fetch_array** (resource $statement [, int $mode])

次の結果行に相当する配列を返します。

oci8 ドライバによるデータ型マッピングの詳細については、ドライバがサポートするデータ型 を参照ください。

ここで言っておくべき事は、**oci_fetch_array()** は oci_fetch_row() よりも 無意味に 遅い、ということですが、非常に使いやすい関数です。

パラメータ

statement

有効な OCI ステートメント ID。

statement

オプションの第2引数は、次の定数の組み合わせが可能です。

- **OCI_BOTH** - 連想配列と配列の両方を返します (**OCI_ASSOC** + **OCI_NUM** と同等)。 これはデフォルトの動作です。
- **OCI_ASSOC** - 連想配列を返します (oci_fetch_assoc() と同等)。
- **OCI_NUM** - 配列を返します (oci_fetch_row() と同等)。
- **OCI_RETURN_NULLS** - フィールドが **NULL** の場合、 空の要素を返します。
- **OCI_RETURN_LOBS** - ディスクリプタの LOB の値を返します。

デフォルトの mode は **OCI_BOTH** です。

返り値

連想配列あるいは配列の両方を返します。 失敗時あるいは結果行がない場合は **FALSE** を返します。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

> **注意**: Oracle は全てのフィールド名を大文字で返し、 結果の連想配列のインデックスも大文字になります。

例

Example#1 OCI_BOTH を使った oci_fetch_array() の例

```
<?php
$connection = oci_connect("apelsin", "kanistra");

$query = "SELECT id, name FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, OCI_BOTH)) {
     echo $row[0]." and ".$row['ID']." is the same<br>";
     echo $row[1]." and ".$row['NAME']." is the same<br>";
}
?>
```

Example#2 OCI_NUM を使った oci_fetch_array() の例

```
<?php
$connection = oci_connect("user", "password");

$query = "SELECT id, name, lob_field FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, OCI_NUM)) {
     echo $row[0]."<br>";
     echo $row[1]."<br>";
     echo $row[2]->read(100)."<br>";  //this will output first 100 bytes from LOB
}
?>
```

Example#3 OCI_ASSOC を使った oci_fetch_array() の例

```
<?php
$connection = oci_connect("user", "password");

$query = "SELECT id, name, lob_field FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, OCI_ASSOC)) {
     echo $row['ID']."<br>";
     echo $row['NAME']."<br>";
     echo $row['LOB_FIELD']."<br>";  //this will output "Object id #1"
}
?>
```

Example#4 OCI_RETURN_LOBS を使った oci_fetch_array() の例

```
<?php
$connection = oci_connect("user", "password");

$query = "SELECT id, name, lob_field FROM fruits";

$statement = oci_parse ($connection, $query);
oci_execute ($statement);

while ($row = oci_fetch_array ($statement, (OCI_NUM+OCI_RETURN_LOBS))) {
     echo $row[0]."<br>";
     echo $row[1]."<br>";
```

```
    echo $row['LOB_FIELD']."<br>";  //this will output LOB's content
}
?>
```

参考

- oci_fetch_assoc()
- oci_fetch_object()
- oci_fetch_row()
- oci_fetch_all()

---

oci_fetch_assoc

(PHP 5, PECL oci8:1.1-1.2.4)

oci_fetch_assoc — 結果データの次の行を連想配列で返す

説明

array **oci_fetch_assoc** (resource $statement)

次の結果行を連想配列として返します

oci_fetch_assoc() をコールするのは、 oci_fetch_array() で **OCI_ASSOC** を指定してコールするのと同じです。

続けて **oci_fetch_assoc()** をコールすると、 次の行を返します。行がない場合は **FALSE** を返します。

oci8 ドライバによるデータ型マッピングの詳細については、ドライバがサポートするデータ型 を参照ください。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

連想配列を返します。 statement にもう行がない場合は **FALSE** を返します。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。
>
> **注意**: Oracle は全てのフィールド名を大文字で返し、 結果の連想配列のインデックスも大文字になります。

参考

- oci_fetch_array()
- oci_fetch_object()
- oci_fetch_row()
- oci_fetch_all()

---

oci_fetch_object

(PHP 5, PECL oci8:1.1-1.2.4)

oci_fetch_object — 結果の次の行をオブジェクトとして返す

説明

object **oci_fetch_object** (resource $statement)

次の結果行をオブジェクトとして返します。

続けて **oci_fetch_object()** をコールすると、 次の行を返します。行がない場合は **FALSE** を返します。

oci8 ドライバによるデータ型マッピングの詳細については、ドライバがサポートするデータ型 を参照ください。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

オブジェクトを返します。 オブジェクトの属性は文中のフィールドと一致しています。 statement にもう行がない場合は **FALSE** を返します。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。
>
> **注意**: Oracle は全てのフィールド名を大文字で返し、 結果オブジェクトの属性名も同様に大文字になります。

参考

- oci_fetch_array()
- oci_fetch_assoc()
- oci_fetch_row()
- oci_fetch_all()

---

oci_fetch_row

(PHP 5, PECL oci8:1.1-1.2.4)

oci_fetch_row — 結果データの次の行を配列で返す

説明

array **oci_fetch_row** (resource $statement)

結果データの次の行を配列で返します。

oci_fetch_row() をコールすることは、 oci_fetch_array() で **OCI_NUM** を指定してコールすることと同じです。

続けて **oci_fetch_row()** をコールすると、 次の行を返します。行がない場合は **FALSE** を返します。

oci8 ドライバによるデータ型マッピングの詳細については、ドライバがサポートするデータ型 を参照ください。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

フィールド情報を含む配列を返します。 statement にもう行がない場合は **FALSE** を返します。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

参考

- oci_fetch_array()
- oci_fetch_assoc()
- oci_fetch_object()
- oci_fetch_all()

---

oci_fetch

(PHP 5, PECL oci8:1.1-1.2.4)

oci_fetch — 結果バッファの次の行を取得する

説明

bool **oci_fetch** (resource $statement)

(SELECT文の) 次の行を内部結果バッファに取得します。

oci8 ドライバによるデータ型マッピングの詳細については、ドライバがサポートするデータ型 を参照ください。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocifetch() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_fetch()** への別名として残されていますが、 推奨されません。

---

oci_field_is_null

(PHP 5, PECL oci8:1.1-1.2.4)

oci_field_is_null — フィールドが **NULL** であるかどうかを確認する

説明

bool **oci_field_is_null** (resource $statement , mixed $field)

statement の、 指定したフィールド field が **NULL** であるかどうかを調べます。

パラメータ

statement

有効な OCI ステートメント ID。

field

フィールド番号またはフィールド名（大文字）。

返り値

field が **NULL** の場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocicolumnisnull() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_field_is_null()** への別名として残されていますが、 推奨されません。

---

oci_field_name

(PHP 5, PECL oci8:1.1-1.2.4)

oci_field_name — 文からのフィールド名を返す

説明

string **oci_field_name** (resource $statement , int $field)

field の名前を返します。

パラメータ

statement

有効な OCI ステートメント ID。

field

フィールド番号（1 から始まる）あるいは名前のいずれか。

返り値

名前を表す文字列、あるいはエラー時に **FALSE** を返します。

例

Example#1 oci_field_name() の例

```
<?php
$conn = oci_connect("scott", "tiger");
$stmt = oci_parse($conn, "SELECT * FROM emp");
oci_execute($stmt);

echo "<table border=¥"1¥">";
echo "<tr>";
echo "<th>名前</th>";
echo "<th>型</th>";
echo "<th>長さ</th>";
echo "</tr>";

$ncols = oci_num_fields($stmt);

for ($i = 1; $i <= $ncols; $i++) {
    $column_name  = oci_field_name($stmt, $i);
    $column_type  = oci_field_type($stmt, $i);
    $column_size  = oci_field_size($stmt, $i);

    echo "<tr>";
    echo "<td>$column_name</td>";
    echo "<td>$column_type</td>";
    echo "<td>$column_size</td>";
    echo "</tr>";
}

echo "</table>¥n";
oci_free_statement($stmt);
oci_close($conn);
?>
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocicolumnname() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_field_name()** への別名として残されていますが、 推奨されません。

参考

- oci_num_fields()
- oci_field_type()
- oci_field_size()

---

oci_field_precision

(PHP 5, PECL oci8:1.1-1.2.4)

oci_field_precision — フィールドの精度を問い合わせる

説明

int **oci_field_precision** (resource $statement , int $field)

field の精度を返します。

FLOAT 型カラムの精度は 0 でなく桁数は -127 となります。 もし精度が 0 の場合、カラムは NUMBER 型、そうでなければ NUMBER(精度, 桁数) となります。

パラメータ

statement

有効な OCI ステートメント ID。

field

フィールド番号 (1 から始まる) あるいは名前のいずれか。

返り値

精度を表す整数値、あるいはエラー時に **FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocicolumnprecision() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_field_precision()** への別名として残されていますが、 推奨されません。

参考

- oci_field_scale()
- oci_field_type()

---

oci_field_scale

(PHP 5, PECL oci8:1.1-1.2.4)

oci_field_scale — フィールドの桁数を問い合わせる

説明

int **oci_field_scale** (resource $statement , int $field)

field に対応するカラムの桁数を返します。

FLOAT 型カラムの精度は 0 でなく桁数は -127 となります。 もし精度が 0 の場合、カラムは NUMBER 型、そうでなければ NUMBER(精度, 桁数) となります。

パラメータ

statement

有効な OCI ステートメント ID。

field

フィールド番号 (1 から始まる) あるいは名前のいずれか。

返り値

桁数を表す整数値、あるいはエラー時に **FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocicolumnscale() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_field_scale()** への別名として残されていますが、 推奨されません。

参考

- oci_field_precision()
- oci_field_type()

oci_field_size

(PHP 5, PECL oci8:1.1-1.2.4)

oci_field_size — フィールドサイズを返す

説明

int **oci_field_size** (resource $statement , mixed $field)

フィールド field のサイズを返します。

パラメータ

statement

有効な OCI ステートメント ID。

field

フィールドのインデックス (1 から始まるもの) あるいは名前。

返り値

フィールド field のサイズを表すバイト数、あるいは エラー時に **FALSE** を返します。

例

Example#1 oci_field_size() の例

```
<?php
$conn = oci_connect("scott", "tiger");
$stmt = oci_parse($conn, "SELECT * FROM emp");
oci_execute($stmt);

echo "<table border=¥"1¥">";
echo "<tr>";
echo "<th>Name</th>";
echo "<th>Type</th>";
echo "<th>Length</th>";
echo "</tr>";

$ncols = oci_num_fields($stmt);

for ($i = 1; $i <= $ncols; $i++) {
    $column_name  = oci_field_name($stmt, $i);
    $column_type  = oci_field_type($stmt, $i);
    $column_size  = oci_field_size($stmt, $i);
    echo "<tr>";
    echo "<td>$column_name</td>";
    echo "<td>$column_type</td>";
    echo "<td>$column_size</td>";
    echo "</tr>";
}

echo "</table>";

oci_free_statement($stmt);
oci_close($conn);
?>
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocicolumnsize() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_field_size()** への別名として残されていますが、 推奨されません。

参考

- oci_num_fields()
- oci_field_name()

oci_field_type_raw

(PHP 5, PECL oci8:1.1-1.2.4)

oci_field_type_raw — Oracle におけるフィールドの型を問い合わせる

説明

int **oci_field_type_raw** (resource $statement , int $field)

フィールド field の、Oracle における型を返します。

しかしながら、フィールドの方を取得したい場合、 oci_field_type() の方が良いでしょう。

パラメータ

statement

有効な OCI ステートメント ID。

field

フィールド番号 (1 から始まる) あるいは名前のいずれか。

返り値

*Oracle のデータ型を表す文字列、あるいはエラー時に **FALSE** を返します。*

注意

> ***注意**: PHP バージョン 5.0.0 以前では、代わりに oci_columntyperaw() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_field_type_raw()** への別名として残されていますが、 推奨されません。*

---

oci_field_type

(PHP 5, PECL oci8:1.1-1.2.4)

oci_field_type — フィールドのデータ型を返す

説明

*mixed **oci_field_type** (resource $statement , int $field)*

フィールドのデータ型を返します。

パラメータ

statement

有効な OCI ステートメント ID。

field

フィールド番号 (1 から始まる) あるいは名前のいずれか。

返り値

*フィールドのデータ型を表す文字列、あるいはエラー時に **FALSE** を返します。*

返り値

Example#1 oci_field_type() の例

```
<?php
$conn = oci_connect("scott", "tiger");
$stmt = oci_parse($conn, "SELECT * FROM emp");
oci_execute($stmt);

echo "<table border=¥"1¥">";
echo "<tr>";
echo "<th>名前</th>";
echo "<th>型</th>";
echo "<th>長さ</th>";
echo "</tr>";

$ncols = oci_num_fields($stmt);

for ($i = 1; $i <= $ncols; $i++) {
    $column_name  = oci_field_name($stmt, $i);
    $column_type  = oci_field_type($stmt, $i);
    $column_size  = oci_field_size($stmt, $i);

    echo "<tr>";
    echo "<td>$column_name</td>";
    echo "<td>$column_type</td>";
    echo "<td>$column_size</td>";
    echo "</tr>";
}

echo "</table>¥n";

oci_free_statement($stmt);
oci_close($conn);
?>
```

注意

> ***注意**: PHP バージョン 5.0.0 以前では、代わりに oci_columntype() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_field_type()** への別名として残されていますが、 推奨されません。*

参考

- *oci_num_fields()*
- *oci_field_name()*
- *oci_field_size()*

oci_free_statement

(PHP 5, PECL oci8:1.1-1.2.4)

oci_free_statement — 文やカーソルに関連付けられた全てのリソースを解放する

説明

bool **oci_free_statement** (resource $statement)

oci_parse() の結果や Oracle から取得した、 Oracle のカーソルおよび文に関連付けられた全てのリソースを解放します。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

oci_internal_debug

(PHP 5, PECL oci8:1.1-1.2.4)

oci_internal_debug — 内部デバッグ用出力有効または無効にする

説明

void **oci_internal_debug** (bool $onoff)

内部デバッグ用出力を有効あるいは無効にします。

パラメータ

onoff

これを **FALSE** にするとデバッグ出力をオフにし、 **TRUE** にするとオンになります。

返り値

値を返しません。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ociinternaldebug() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_internal_debug()** への別名として残されていますが、 推奨されません。

OCI-Lob->append

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->append — ラージオブジェクトのデータを他のラージオブジェクトに追加する

説明

OCI-Lob
bool **append** (OCI-Lob $lob_from)

ラージオブジェクトのデータを他のラージオブジェクトに追加します。

前もってバッファリングが有効になっている場合、 **OCI-Lob->append()** を使ったラージオブジェクトの書き込みは失敗するでしょう。 追加の前にバッファリングを無効にしなければなりません。 バッファリングを無効にする前に、 OCI-Lob->flush によってバッファをフラッシュする必要があるかも知れません。

パラメータ

lob_from

コピーする LOB。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->flush

- OCI-Lob->setBuffering
- OCI-Lob->getBuffering

OCI-Lob->close

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->close — LOB ディスクリプタを閉じる

説明

OCI-Lob
bool **close** (void)

OCI-Lob->close() は LOB あるいは FILE のディスクリプタを閉じます。この関数は OCI-Lob->writeTemporary を併用した場合のみ使用されるべきです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->writeTemporary

---

oci_lob_copy

(PHP 5, PECL oci8:1.1-1.2.4)

oci_lob_copy — ラージオブジェクトをコピーする

説明

bool **oci_lob_copy** (OCI-Lob $lob_to , OCI-Lob $lob_from [, int $length])

ラージオブジェクトあるいはラージオブジェクトの一部を 他のラージオブジェクトにコピーします。 受け取り側の LOB の古いデータは上書きされます。

もし LOB の特定部分を LOB の特定の位置にコピーする必要がある場合、 oci_lob_seek() を使用して LOB の内部ポインタを移動させることができます。

パラメータ

lob_to

コピー先の LOB。

lob_from

コピー元の LOB。

length

コピーされるデータの長さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

OCI-Lob->eof

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->eof — ラージオブジェクトのディスクリプタが EOF かどうかを調べる

説明

OCI-Lob
bool **eof** (void)

ラージオブジェクトの内部ポインタが LOB の最後にあるかどうかを調べます。

返り値

もし、ラージオブジェクトの内部ポインタが LOB の最後にあるとき、 **TRUE** を返します。その他の場合は **FALSE** を返します。

参考

- OCI-Lob->size

OCI-Lob->erase

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->erase — 内部 LOB データの特定の位置を消去する

説明

OCI-Lob
int **erase** ([int $offset [, int $length]])

指定された offset から始まる内部 LOB データの特定の位置を消去します。 パラメータなしでコールすると、すべての LOB データを消去します。

BLOB の場合、消去するということは存在する LOB の値が 0 バイトで上書きされる、ということを意味します。 CLOB の場合、存在する LOB の値はスペースで上書きされる、ということです。

パラメータ

offset

length

返り値

消去された実際の文字数あるいはバイト数、エラーの場合は **FALSE** を返します。

参考

- OCI-Lob->truncate

OCI-Lob->export

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->export — LOB の内容をファイルに出力する

説明

OCI-Lob
bool **export** (string $filename [, int $start [, int $length]])

LOB の内容をファイルに出力します。

パラメータ

filename

ファイルへのパス。

start

出力を開始する位置。

length

出力するデータの長さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->import

OCI-Lob->flush

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->flush — LOB のバッファをサーバにフラッシュする、あるいは書き込む

説明

OCI-Lob
bool **flush** ([int $flag])

OCI-Lob->flush() は実際にサーバにデータを書き込みます。

パラメータ

flag

デフォルトではリソースは解放されませんが、フラグ **OCI_LOB_BUFFER_FREE** を使用することで明示的に行うことができます。 あなたが何をしようとしているかを理解しておいてください - 次に LOB の同じ箇所に読み書きしようとする場合、 サーバへのラウンドトリップを伴い、 新しいバッファリソースが初期化されるでしょう。 もはや LOB に対して何も行わない場合のみ、 **OCI_LOB_BUFFER_FREE** フラグを使用することが推奨されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

バッファリングが有効でない、あるいはエラーが発生した場合、 **FALSE** を返します。

参考

- OCI-Lob->getBuffering
- OCI-Lob->setBuffering

OCI-Lob->free

(No version information available, might be only in CVS)

OCI-Lob->free — LOB ディスクリプタに関連付けられたりソースを解放する

説明

OCI-Lob
bool **free** (void)

事前に oci_new_descriptor() を使用して割り当てられた ディスクリプタに関連付けられたりソースを解放します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

OCI-Lob->getBuffering

(No version information available, might be only in CVS)

OCI-Lob->getBuffering — ラージオブジェクトに対する現在のバッファリングの状態を返す

説明

OCI-Lob
bool **getBuffering** (void)

ラージオブジェクトに対するバッファリングが有効か無効かを調べます。

返り値

もしラージオブジェクトに対するバッファリングが無効な場合は **FALSE**、 有効な場合は **TRUE** を返します。

参考

- OCI-Lob->setBuffering

OCI-Lob->import

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->import — ファイルデータを LOB にインポートする

説明

OCI-Lob
bool **import** (string $filename)

filename からのデータをラージオブジェクトの現在の位置に書き込みます。

パラメータ

filename

ファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->export
- OCI-Lob->write

---

oci_lob_is_equal

(PHP 5, PECL oci8:1.1-1.2.4)

oci_lob_is_equal — 2 つの LOB/FILE ロケータの等価性を比較する

説明

bool **oci_lob_is_equal** (OCI-Lob $lob1 , OCI-Lob $lob2)

2 つの LOB/FILE ロケータを比較します。

パラメータ

lob1

LOB の ID。

lob2

LOB の ID。

返り値

これらのオブジェクトが等しい場合 **TRUE** を返し、 そうでなければ **FALSE** を返します。

OCI-Lob->load

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->load — ラージオブジェクトの内容を返す

説明

OCI-Lob
string **load** (void)

ラージオブジェクトの内容を返します。 memory_limit に達した場合、 スクリプトの実行が終了されますので、 LOB がこの制限を超えないことを確認してください。 多くの場合、代わりに OCI-Lob->read を使用することが推奨されます。

返り値

オブジェクトの内容、あるいはエラー時に **FALSE** を返します。

参考

- OCI-Lob->read

OCI-Lob->read

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->read — ラージオブジェクトの一部を読み込む

説明

OCI-Lob
string **read** (int $length)

LOB の内部ポインタの現在位置から length バイト読み込みます。

length バイトが読み込まれた、 あるいはラージオブジェクトの終わりに達したとき、読み込みを停止します。 ラージオブジェクトの内部ポインタは、 読み込まれたバイト数分だけシフトされます。

パラメータ

length

読み込むバイト数。

返り値

読み込んだ内容を表す文字列、あるいはエラーの場合に **FALSE** を返します。

参考

- OCI-Lob->load
- OCI-Lob->write

OCI-Lob->rewind

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->rewind — 内部ポインタをラージオブジェクトの先頭に移動する

説明

OCI-Lob
bool **rewind** (void)

内部ポインタをラージオブジェクトの先頭にセットします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->seek
- OCI-Lob->tell

OCI-Lob->save

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->save — データをラージオブジェクトに保存する

説明

OCI-Lob
bool **save** (string $data [, int $offset])

data をラージオブジェクトに保存します。

パラメータ

data

保存するデータ。

offset

ラージオブジェクトの先頭からのオフセットを指定するために利用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->write
- OCI-Lob->import

OCI-Lob->saveFile

(No version information available, might be only in CVS)

OCI-Lob->saveFile — oci_lob_import() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_lob_import().

OCI-Lob->seek

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->seek — ラージオブジェクトの内部ポインタをセットする

説明

OCI-Lob
bool **seek** (int $offset [, int $whence])

ラージオブジェクトの内部ポインタをセットします。

パラメータ

offset

内部ポインタを whence で示された位置から移動させるバイト数を表します。

whence

以下のいずれかです。

- **OCI_SEEK_SET** - offset と等しい位置にセットします
- **OCI_SEEK_CUR** - 現在の位置に offset バイト追加します
- **OCI_SEEK_END** - ラージオブジェクトの終端に offset バイト追加します（ラージオブジェクトの終端より前の位置に移動するには負の値を指定します）

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->rewind
- OCI-Lob->tell
- OCI-Lob->eof

OCI-Lob->setBuffering

(No version information available, might be only in CVS)

OCI-Lob->setBuffering — 現在のラージオブジェクト用のバッファリング状態を変更する

説明

OCI-Lob
bool **setBuffering** (bool $on_off)

ラージオブジェクト用のバッファリングを on_off パラメータの値で設定します。

この関数を使用することで、LOB の細かな読み込みや書き込みがバッファリングされことにより ネットワークのラウンドトリップや LOB バージョニングの回数が低減され、 パフォーマンスが改善されます。 oci_lob_flush() はラージオブジェクトの処理が完了した際、 バッファをフラッシュするために使用します。

パラメータ

on_off

TRUE は有効、**FALSE** は無効を表します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 同じフラグで繰り返しコールすると **TRUE** を返します。

参考

- OCI-Lob->getBuffering

OCI-Lob->size

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->size — ラージオブジェクトのサイズを返す

説明

OCI-Lob
int **size** (void)

ラージオブジェクトのサイズを取得します。

返り値

ラージオブジェクトのサイズを返します。 エラーの場合、**FALSE** を返します。 空のオブジェクトは長さ 0 となります。

OCI-Lob->tell

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->tell — ラージオブジェクトの内部ポインタの現在位置を返す

説明

OCI-Lob
int **tell** (void)

LOB の内部ポインタの現在位置を取得します。

返り値

LOB の内部ポインタの現在位置を返します。 エラーの場合、**FALSE** を返します。

参考

- OCI-Lob->rewind
- OCI-Lob->size
- OCI-Lob->eof

OCI-Lob->truncate

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->truncate — ラージオブジェクトを切りつめる

説明

OCI-Lob
bool **truncate** ([int $length])

LOB を切りつめます。

パラメータ

length

指定した場合は、このメソッドはラージオブジェクトを length バイトに切りつめます。 そうでない場合は、LOB を完全に消去します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->erase

OCI-Lob->write

(PHP 5, PECL oci8:1.1-1.2.4)

OCI-Lob->write — データをラージオブジェクトに書き込む

説明

OCI-Lob
int **write** (string $data [, int $length])

パラメータ data からのデータを LOB の内部ポインタの現在位置に書き込みます。

パラメータ

data

LOB に書き込むデータ。

length

このパラメータを指定すると、 length バイト書き込むか data の終端に達するか、 いずれか早い方の後に書き込みが停止します。

返り値

書き込んだバイト数、あるいはエラー時に **FALSE** を返します。

参考

- OCI-Lob->read

OCI-Lob->writeTemporary

(No version information available, might be only in CVS)

OCI-Lob->writeTemporary — 一時的なラージオブジェクトを書き込む

説明

OCI-Lob
bool **writeTemporary** (string $data [, int $lob_type])

一時的なラージオブジェクトを生成し、 data を書き込みます。

オブジェクトの使用後、OCI-Lob->close を使用するべきです。

パラメータ

data

書き込むデータ。

lob_type

以下のいずれかです。

- **OCI_TEMP_BLOB** は一時的な BLOB を生成するために使用します
- **OCI_TEMP_CLOB** (デフォルト値) は一時的な CLOB を生成するために使用します

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- OCI-Lob->close

OCI-Lob->writeToFile

(No version information available, might be only in CVS)

OCI-Lob->writeToFile — oci_lob_export() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_lob_export().

---

oci_new_collection

(PHP 5, PECL oci8:1.1-1.2.4)

oci_new_collection — 新しいコレクションオブジェクトを割り当てる

説明

OCI-Collection **oci_new_collection** (resource $connection , string $tdo [, string $schema])

新しいコレクションオブジェクトを割り当てます。

パラメータ

connection

oci_connect() あるいは oci_pconnect() が返す Oracle 接続 ID。

tdo

有効な型名 (大文字)。

schema

型を作成したスキーマを指定しなければなりません。 現在のユーザ名がデフォルト値となります。

返り値

新しい OCICollection オブジェクト、 あるいはエラー時に **FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocinewcollection() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_new_collection()** への別名として残されていますが、 推奨されません。

---

oci_new_connect

(PHP 5, PECL oci8:1.1-1.2.4)

oci_new_connect — Oracle サーバへの新規接続を確立する

説明

resource **oci_new_connect** (string $username , string $password [, string $db [, string $charset [, int $session_mode]]])

Oracle サーバへの新規接続を確立し、ログオンします。

oci_connect() や oci_pconnect() と異なり、**oci_new_connect()** は接続をキャッシュしません。また、 常に新しくオープンされた接続ハンドルを返します。 これは、もしアプリケーションが 2 セットのクエリ間でトランザクション的な独立を必要とする場合に有効です。

パラメータ

username

Oracle のユーザ名。

password

username のパスワード。

db

オプションのパラメータで、ローカルの Oracle インスタンスの名前か tnsnames.ora における接続先のエントリ名を指定することができます。

指定しない場合は、PHP は環境変数 ORACLE_SID および TWO_TASK を用いてローカルの Oracle インスタンス名および tnsnames.ora の場所を見つけます。

charset

Oracle サーバのバージョン 9.2 以降を使用している場合、新規接続を確立する際に charset パラメータを指定することができます。 Oracleサーバ < 9.2 を使用している場合、このパラメータは無視され、かわりに環境変数 NLS_LANG が使用されます。

session_mode

このパラメータはバージョン 1.1 から利用可能で、次の値を受け付けます: **OCI_DEFAULT**, **OCI_SYSOPER**, **OCI_SYSDBA** 。 **OCI_SYSOPER** もしくは **OCI_SYSDBA** のいずれかが指定された場合、 oci_connect() は外部の信用を利用して 権限付きの接続を確立しようとします。 デフォルトでは権限付きの接続は無効です。有効にするためには、oci8.privileged_connect をオンにしてください。

返り値

接続 ID、あるいはエラー時に **FALSE** を返します。

例

以下の例は、接続がどのように分割されるかを示すものです。

Example#1 oci_new_connect() の例

```
<?php
echo "<html><pre>";
$db = "";

$c1 = oci_connect("scott", "tiger", $db);
$c2 = oci_new_connect("scott", "tiger", $db);

function create_table($conn)
{
  $stmt = oci_parse($conn, "create table scott.hallo (test
varchar2(64))");
  oci_execute($stmt);
  echo $conn . " created table¥n¥n";
}

function drop_table($conn)
{
  $stmt = oci_parse($conn, "drop table scott.hallo");
  oci_execute($stmt);
  echo $conn . " dropped table¥n¥n";
}

function insert_data($conn)
{
  $stmt = oci_parse($conn, "insert into scott.hallo
               values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " inserted hallo¥n¥n";
}

function delete_data($conn)
{
  $stmt = oci_parse($conn, "delete from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . " deleted hallo¥n¥n";
}

function commit($conn)
{
  oci_commit($conn);
  echo $conn . " committed¥n¥n";
}
```

```
function rollback($conn)
{
  oci_rollback($conn);
  echo $conn . "  rollback¥n¥n";
}

function select_data($conn)
{
  $stmt = oci_parse($conn, "select * from scott.hallo");
  oci_execute($stmt, OCI_DEFAULT);
  echo $conn . "----selecting¥n¥n";
  while (oci_fetch($stmt)) {
    echo $conn . " <" . oci_result($stmt, "TEST") . ">¥n¥n";
  }
  echo $conn . "----done¥n¥n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);

drop_table($c1);
echo "</pre></html>";
?>
```

注意

注意: もし PHP を Oracle Instant Client と併用している場合、 次に解説されている簡単な接続ネーミングメソッドを使用することができます: » http://download-west.oracle.com/docs/cd/B12037_01/network.101/b10775/naming.htm#i498306 。 基本的にデータベース名として "//db_host[:port]/database_name" を指定できることを意味します。しかし、 古いネーミングメソッドを使用したい場合、**ORACLE_HOME** もしくは **TNS_ADMIN** のいずれかを設定 しなければなりません。

注意: PHP バージョン 5.0.0 以前では、代わりに ocinlogon() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_new_connect()** への別名として残されていますが、 推奨されません。

参考

- oci_connect()
- oci_pconnect()

---

oci_new_cursor

(PHP 5, PECL oci8:1.1-1.2.4)

oci_new_cursor — 新規カーソル（ステートメントハンドル）を割り当て、それを返す

説明

resource **oci_new_cursor** (resource $connection)

指定された接続に新規カーソルを割り当てます。

パラメータ

connection

oci_connect() あるいは oci_pconnect() が返す Oracle 接続 ID。

返り値

新しいステートメントハンドル、あるいはエラー時に **FALSE** を返します。

例

Example#1 Oracle ストアドプロシージャにおける REF CURSOR の使用例

```
<?php
// ストアドプロシージャ info.output が :data の REF CURSOR を返すと仮定する

$conn = oci_connect("scott", "tiger");
```

```
$curs = oci_new_cursor($conn);
$stmt = oci_parse($conn, "begin  info.output(:data);  end;");

oci_bind_by_name($stmt, "data", $curs, -1, OCI_B_CURSOR);
oci_execute($stmt);
oci_execute($curs);

while ($data = oci_fetch_row($curs)) {
    var_dump($data);
}

oci_free_statement($stmt);
oci_free_statement($curs);
oci_close($conn);
?>
```

Example#2 Oracle の select 文における REF CURSOR の使用例

```
<?php
echo "<html><body>";
$conn = oci_connect("scott", "tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = oci_parse($conn, "select  deptno,dname,$count_cursor");

oci_execute($stmt);
echo "<table  border=¥"1¥">";
echo "<tr>";
echo "<th>DEPT  NAME</th>";
echo "<th>DEPT  #</th>";
echo "<th>#  EMPLOYEES</th>";
echo "</tr>";

while ($data = oci_fetch_assoc($stmt)) {
    echo "<tr>";
    $dname   = $data["DNAME"];
    $deptno  = $data["DEPTNO"];
    echo "<td>$dname</td>";
    echo "<td>$deptno</td>";
    oci_execute($data["EMPCNT"]);
    while ($subdata = oci_fetch_assoc($data["EMPCNT"])) {
        $num_emps = $subdata["NUM_EMPS"];
        echo  "<td>$num_emps</td>";
    }
    echo "</tr>";
}
echo "</table>";
echo "</body></html>";
oci_free_statement($stmt);
oci_close($conn);
?>
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocinewcursor() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_new_cursor()** への別名として残されていますが、 推奨されません。

---

oci_new_descriptor

(PHP 5, PECL oci8:1.1-1.2.4)

oci_new_descriptor — 空の新規 LOB あるいは FILE ディスクリプタを初期化する

説明

OCI-Lob **oci_new_descriptor** (resource $connection [, int $type])

ディスクリプタあるいは LOB ロケータを保持するリソースを確保します。

パラメータ

connection

oci_connect() あるいは oci_pconnect() が返す Oracle 接続 ID。

type

type として有効な値は、 **OCI_D_FILE**、 **OCI_D_LOB** および **OCI_D_ROWID** です。

返り値

成功した場合に新しい LOB あるいは FILE ディスクリプタ、 エラー時に **FALSE** を返します。

例

Example#1 oci_new_descriptor() の例

```
<?php
/* このスクリプトは HTML フォームからコールされるよう設計されています。
 * また、$user, $passwor, $table, $where, $commitsizeがフォームから
 * 渡されることを前提にしています。このスクリプトは、ROWID を用いて
 * 選択された行を削除し、$commitsize 行毎にコミットします。
 * (ロールバックがないので、注意して使用してください。)
```

```
 */
$conn = oci_connect($user, $password);
$stmt = oci_parse($conn, "select rowid from $table $where");
$rowid = oci_new_descriptor($conn, OCI_D_ROWID);
oci_define_by_name($stmt, "ROWID", $rowid);
oci_execute($stmt);
while (oci_fetch($stmt)) {
    $nrows = oci_num_rows($stmt);
    $delete = oci_parse($conn, "delete from $table where ROWID = :rid");
    oci_bind_by_name($delete, ":rid", $rowid, -1, OCI_B_ROWID);
    oci_execute($delete);
    echo "$nrows¥n";
    if (($nrows % $commitsize) == 0) {
        oci_commit($conn);
    }
}
$nrows = oci_num_rows($stmt);
echo "$nrows deleted...¥n";
oci_free_statement($stmt);
oci_close($conn);
?>
<?php
    /* このスクリプトは LOB カラムにファイルをアップロードする例を示します。
     * LOBカラムにアップロードを行うこの例に関するフォームは、
       * <form action="upload.php3" method="post" enctype="multipart/form-data">
      * <input type="file" name="lob_upload" />
     * ... のようなものが使用されます。
     */
  if (!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload" /><br />
<input type="submit" value="Upload" /> - <input type="reset" value="Reset" />
</form>
<?php
  } else {

    // $lob_upload には、アップロードされたファイルの一時的なファイル名が含まれます

     // 安全にアップロードしたい場合は、マニュアルの
     // ファイルアップロードに関する節を参照ください

     $conn = oci_connect($user, $password);
     $lob = oci_new_descriptor($conn, OCI_D_LOB);
     $stmt = oci_parse($conn, "insert into $table (id, the_blob)
                  values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     oci_bind_by_name($stmt, ':the_blob', $lob, -1, OCI_B_BLOB);
     oci_execute($stmt, OCI_DEFAULT);
     if ($lob->savefile($lob_upload)){
        oci_commit($conn);
        echo "Blob のアップロードは成功しました\n";
     }else{
        echo "Blob をアップロードできませんでした\n";
     }
     oci_free_descriptor($lob);
     oci_free_statement($stmt);
     oci_close($conn);
  }
?>
```

Example#2 oci_new_descriptor() の例

```
<?php
/* 入力パラメータとして clob を含む PL/SQLストアドプロシージャを
 * コールする (PHP 4 >= 4.0.6) 。
 * PL/SQL ストアドプロシージャのシグネチャは次の通り。
 *
 * PROCEDURE save_data
 *   Argument Name                  Type                      In/Out Default?
 *   ------------------------------ ------------------------  ------ --------
 *   KEY                            NUMBER(38)                IN
 *   DATA                           CLOB                      IN
 *
 */

$conn = oci_connect($user, $password);
$stmt = oci_parse($conn, "begin save_data(:key, :data); end;");
$clob = oci_new_descriptor($conn, OCI_D_LOB);
oci_bind_by_name($stmt, ':key', $key);
oci_bind_by_name($stmt, ':data', $clob, -1, OCI_B_CLOB);
$clob->write($data);
oci_execute($stmt, OCI_DEFAULT);
oci_commit($conn);
$clob->free();
oci_free_statement($stmt);
?>
```

注意

> ***注意***: PHP バージョン 5.0.0 以前では、代わりに ocinewdescriptor() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため ***oci_new_descriptor()*** への別名として残されていますが、 推奨されません。

---

oci_num_fields

(PHP 5, PECL oci8:1.1-1.2.4)

oci_num_fields — ある文における結果のカラム数を返す

説明

int **oci_num_fields** (resource $statement)

指定した文 statement におけるカラムの数を返します。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

カラムの数を表す整数値、あるいはエラー時に **FALSE** を返します。

例

Example#1 oci_num_fields() の例

```
<?php
$conn = oci_connect("scott", "tiger");
$stmt = oci_parse($conn, "select * from emp");

oci_execute($stmt);

while (oci_fetch($stmt)) {
    echo "¥n";
    $ncols = oci_num_fields($stmt);
    for ($i = 1; $i <= $ncols; $i++) {
        $column_name  = oci_field_name($stmt, $i);
        $column_value = oci_result($stmt, $i);
        echo $column_name . ': ' . $column_value . "¥n";
    }
    echo "¥n";
}

oci_free_statement($stmt);
oci_close($conn);
?>
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocinumcols() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_num_fields()** への別名として残されていますが、 推奨されません。

---

oci_num_rows

(PHP 5, PECL oci8:1.1-1.2.4)

oci_num_rows — 文の実行で作用された行数を取得する

説明

int **oci_num_rows** (resource $statement)

文の実行で作用された行数を取得します。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

作用された行の数を表す整数値、あるいはエラー時に **FALSE** を返します。

例

Example#1 oci_num_rows() の例

```
<?php
$conn = oci_connect("scott", "tiger");

$stmt = oci_parse($conn, "create table emp2 as select * from emp");
oci_execute($stmt);
echo oci_num_rows($stmt) . " rows inserted.<br />";
oci_free_statement($stmt);

$stmt = oci_parse($conn, "delete from emp2");
oci_execute($stmt, OCI_DEFAULT);
echo oci_num_rows($stmt) . " rows deleted.<br />";
oci_commit($conn);
oci_free_statement($stmt);

$stmt = oci_parse($conn, "drop table emp2");
oci_execute($stmt);
```

```
oci_free_statement($stmt);

oci_close($conn);
?>
```

注意

注意: この関数は、select が返す行の数は 返しません ! SELECT 文の場合、この関数は **oci_fetch*()** 関数によってバッファに取得された行数を返します。

注意: PHP バージョン 5.0.0 以前では、代わりに ocirowcount() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_num_rows()** への別名として残されていますが、 推奨されません。

---

oci_parse

(PHP 5, PECL oci8:1.1-1.2.4)

oci_parse — 実行のために Oracle の文をパースする

説明

resource **oci_parse** (resource $connection , string $query)

connection を使って query をパースし、ステートメント ID を返します。この ID は、oci_bind_by_name(), oci_execute() や他の関数で使用されます。

パラメータ

connection

oci_connect() あるいは oci_pconnect() が返す Oracle 接続 ID。

query

SQL クエリ。

返り値

成功した場合にステートメントハンドル、あるいはエラー時に or **FALSE** を返します。

注意

注意: この関数は query を検証 しません。 query が有効な SQL あるいは PL/SQL 文かどうかを検証する唯一の方法 - それは実行することです。

注意: PHP バージョン 5.0.0 以前では、代わりに ociparse() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_parse()** への別名として残されていますが、 推奨されません。

---

oci_password_change

(PHP 5, PECL oci8:1.1-1.2.4)

oci_password_change — Oracle ユーザのパスワードを変更する

説明

bool **oci_password_change** (resource $connection , string $username , string $old_password , string $new_password)
resource **oci_password_change** (string $dbname , string $username , string $old_password , string $new_password)

ユーザ username のパスワードを変更します。

パラメータ

connection

oci_connect() あるいは oci_pconnect() が返す Oracle 接続 ID。

username

Oracle のユーザ名。

old_password

旧パスワード。

new_password

設定したい、新しいパスワード。

dbname

データベース名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: **oci_password_change()** の 2 番目の書式はバージョン 1.1 から利用可能です。

注意: PHP バージョン 5.0.0 以前では、代わりに **ocipasswordchange()** を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_password_change()** への別名として残されていますが、 推奨されません。

---

oci_pconnect

(PHP 5, PECL oci8:1.1-1.2.4)

oci_pconnect — 持続的接続を使用してOracle データベースに接続する

説明

resource **oci_pconnect** (string $username , string $password [, string $db [, string $charset [, int $session_mode]]])

Oracle サーバへの持続的接続を生成し、ログオンします。

持続的接続はキャッシュされ、リクエスト間で再利用されることで、 各ページロードのオーバーヘッドを軽減します。 典型的な PHP アプリケーションでは、Apache の子プロセス (もしくは PHP FastCGI/CGI プロセス) ごとに Oracle サーバに対してオープンされた単一の持続的接続を有します。 より詳細な情報については、持続的データベース接続 のセクションを参照してください。

パラメータ

username

Oracle のユーザ名。

password

username のパスワード。

db

このオプションのパラメータには、ローカル Oracle インスタンスの名前か tnsnames.ora における接続先のエントリ名を指定することができます。

指定しない場合は、PHP は環境変数 ORACLE_SID および TWO_TASK を用いてローカルの Oracle インスタンス名および tnsnames.ora の場所を見つけます。

charset

Oracle サーバのバージョン 9.2 以降を使用している場合、新規接続を確立する際に charset パラメータを指定することができます。 Oracleサーバ < 9.2 を使用している場合、このパラメータは無視され、かわりに環境変数 NLS_LANG が使用されます。

session_mode

パラメータ session_mode はバージョン 1.1 からり用可能で、次の値を受け付けます: **OCI_DEFAULT**, **OCI_SYSOPER**, **OCI_SYSDBA** 。 **OCI_SYSOPER** もしくは **OCI_SYSDBA** のいずれかが指定された場合、 oci_connect() は外部の信用を利用して 権限付きの接続を確立しようとします。 デフォルトでは権限付きの接続は無効です。有効にするためには、oci8.privileged_connect をオンにしてください。

返り値

接続 ID、あるいはエラー時に **FALSE** を返します。

注意

注意: oci8 拡張モジュールのバージョン 1.1 から、持続的 Oracle 接続の生存時間と最大数が次の設定値を設定することで調整可能になりました: oci8.persistent_timeout, oci8.ping_interval, oci8.max_persistent 。

注意: もし PHP を Oracle Instant Client と併用している場合、 次に解説されている簡単な接続ネーミングメソッドを使用することができます: » http://download-west.oracle.com/docs/cd/B12037_01/network.101/b10775/naming.htm#i498306 。 基本的にデータベース名として "//db_host[:port]/database_name" を指定できることを意味します。しかし、 古いネーミングメソッドを使用したい場合、**ORACLE_HOME** もしくは **TNS_ADMIN** のいずれかを設定 しなければなりません。

注意: PHP バージョン 5.0.0 以前では、代わりに ociplogon() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_pconnect()** への別名として残されていますが、 推奨されません。

参考

- oci_connect()
- oci_new_connect()

---

oci_result

(PHP 5, PECL oci8:1.1-1.2.4)

oci_result — フェッチした行からフィールドの値を取得する

説明

mixed **oci_result** (resource $statement , mixed $field)

oci_fetch() によってフェッチされた現在の行のフィールド field からデータを返します。

oci8 ドライバによるデータ型マッピングの詳細については、ドライバがサポートするデータ型 を参照ください。

パラメータ

statement

field

カラム番号（1から始まる）またはカラム名（大文字）のどちらかを使用可能です。

返り値

抽象型（ROWID, LOB, FILE）を除き、全てを文字列として返します。 エラーの場合、**FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ociresult() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_result()** への別名として残されていますが、 推奨されません。

参考

- oci_fetch_array()
- oci_fetch_assoc()
- oci_fetch_object()
- oci_fetch_row()
- oci_fetch_all()

---

oci_rollback

(PHP 5, PECL oci8:1.1-1.2.4)

oci_rollback — 未解決のトランザクションをロールバックする

説明

bool **oci_rollback** (resource $connection)

Oracle 接続 connection に関する 全ての未解決の文をロールバックします。

パラメータ

connection

oci_connect() あるいは oci_pconnect() が返す Oracle 接続 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: トランザクションは接続を閉じたとき、 もしくはスクリプトの終了時のいずれの場合でも すぐに自動的にロールバックされます。トランザクションをコミットするには oci_commit() をコールする、 もしくはトランザクションを破棄する場合は **oci_rollback()** を明示的にコールする必要があります。
>
> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocirollback() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_rollback()** への別名として残されていますが、 推奨されません。

参考

- oci_commit()

---

oci_server_version

(PHP 5, PECL oci8:1.1-1.2.4)

oci_server_version — サーバのバージョンを返す

説明

string **oci_server_version** (resource $connection)

Oracle サーバのバージョン情報を文字列で返します。 その際、指定した接続 connection を使用します。

パラメータ

connection

返り値

バージョン情報を表す文字列、あるいはエラー時に **FALSE** を返します。

例

Example#1 oci_server_version() の例

```
<?php
    $conn = oci_connect("scott", "tiger");
    echo "サーバのバージョン: " . oci_server_version($conn);
    oci_close($conn);
?>
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ociserverversion() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_server_version()** への別名として残されていますが、 推奨されません。

---

oci_set_prefetch

(PHP 5, PECL oci8:1.1-1.2.4)

oci_set_prefetch — プリフェッチする行数を設定する

説明

bool **oci_set_prefetch** (resource $statement , int $rows)

oci_execute() のコールが成功した後にプリフェッチする行数を設定します。

パラメータ

statement

rows

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocisetprefetch() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため **oci_set_prefetch()** への別名として残されていますが、 推奨されません。

参考

- oci8_.default_prefetch ini オプション

---

oci_statement_type

(PHP 5, PECL oci8:1.1-1.2.4)

oci_statement_type — OCI ステートメントの種類を返す

説明

string **oci_statement_type** (resource $statement)

OCI ステートメント statement の種類を返します。

パラメータ

statement

有効な OCI ステートメント ID。

返り値

ステートメント statement のクエリの種類を次のいずれかの値で返します。

1. SELECT
2. UPDATE
3. DELETE
4. INSERT
5. CREATE
6. DROP
7. ALTER
8. BEGIN
9. DECLARE
10. CALL (PHP 5.2.1 および OCI8 1.2.3 以降)
11. UNKNOWN

エラー時には **FALSE** を返します。

例

Example#1 oci_statement_type() の例

```
<?php
     $conn = oci_connect("scott", "tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = oci_parse($conn, $sql);
    if (oci_statement_type($stmt) == "DELETE") {
        die("このテーブルを削除することはできません<br />");
    }

    oci_close($conn);
?>
```

注意

> **注意**: PHP バージョン 5.0.0 以前では、代わりに ocistatementtype() を使用しなければなりません。 まだこの名前を使用することができ、下位互換性のため oci_fetch_all() への別名として残されていますが、 推奨されません。

---

ocibindbyname

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocibindbyname — oci_bind_by_name() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_bind_by_name().

---

ocicancel

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicancel — oci_cancel() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_cancel().

---

ocicloselob

(PHP 4 >= 4.0.6, PECL oci8:1.0)

ocicloselob — OCI-Lob->close のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Lob->close.

---

ocicollappend

(PHP 4 >= 4.0.6, PHP 5, PECL oci8:1.0-1.2.4)

ocicollappend — OCI-Collection->append のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->append.

---

ocicollassign

(PHP 4 >= 4.0.6, PECL oci8:1.0)

ocicollassign — OCI-Collection->assign のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->assign.

---

ocicollassignelem

(PHP 4 >= 4.0.6, PHP 5, PECL oci8:1.0-1.2.4)

ocicollassignelem — OCI-Collection->assignElem のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->assignElem.

---

ocicollgetelem

(PHP 4 >= 4.0.6, PHP 5, PECL oci8:1.0-1.2.4)

ocicollgetelem — OCI-Collection->getElem のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->getElem.

---

ocicollmax

(PHP 4 >= 4.0.6, PHP 5, PECL oci8:1.0-1.2.4)

ocicollmax — OCI-Collection->max のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->max.

---

ocicollsize

(PHP 4 >= 4.0.6, PHP 5, PECL oci8:1.0-1.2.4)

ocicollsize — OCI-Collection->size のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->size.

---

ocicolltrim

(PHP 4 >= 4.0.6, PHP 5, PECL oci8:1.0-1.2.4)

ocicolltrim — OCI-Collection->trim のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->trim.

---

ocicolumnisnull

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicolumnisnull — oci_field_is_null() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_field_is_null().

---

ocicolumnname

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicolumnname — oci_field_name() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_field_name().

---

ocicolumnprecision

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicolumnprecision — oci_field_precision() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_field_precision().

---

ocicolumnscale

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicolumnscale — oci_field_scale() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_field_scale().

---

ocicolumnsize

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicolumnsize — oci_field_size() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_field_size().

---

ocicolumntype

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicolumntype — oci_field_type() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_field_type().

---

ocicolumntyperaw

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicolumntyperaw — oci_field_type_raw() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_field_type_raw().

---

ocicommit

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocicommit — oci_commit() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_commit().

---

ocidefinebyname

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocidefinebyname — oci_define_by_name() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_define_by_name().

---

ocierror

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocierror — oci_error() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_error().

---

ociexecute

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociexecute — oci_execute() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_execute().

---

ocifetch

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocifetch — oci_fetch() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_fetch().

---

ocifetchinto

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocifetchinto — 結果配列の次の行を取得する (非推奨)

説明

int **ocifetchinto** (resource $statement , array &$result [, int $mode])

この関数は推奨されません。代わりに推奨されるものは、 oci_fetch_array()、 oci_fetch_object()、 oci_fetch_assoc() および oci_fetch_row() です。

---

ocifetchstatement

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocifetchstatement — oci_fetch_all() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_fetch_all().

---

ocifreecollection

(PHP 4 >= 4.0.7, PHP 5, PECL oci8:1.0-1.2.4)

ocifreecollection — OCI-Collection->free のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Collection->free.

---

ocifreecursor

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocifreecursor — oci_free_statement() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_free_statement().

---

ocifreedesc

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocifreedesc — OCI-Lob->free のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Lob->free.

---

ocifreestatement

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocifreestatement — oci_free_statement() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_free_statement().

---

ociinternaldebug

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociinternaldebug — oci_internal_debug() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_internal_debug().

---

ociloadlob

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociloadlob — OCI-Lob->load のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Lob->load.

---

ocilogoff

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocilogoff — oci_close() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_close().

---

ocilogon

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocilogon — oci_connect() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_connect().

---

ocinewcollection

(PHP 4 >= 4.0.6, PHP 5, PECL oci8:1.0-1.2.4)

ocinewcollection — oci_new_collection() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_new_collection().

---

ocinewcursor

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocinewcursor — oci_new_cursor() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_new_cursor().

---

ocinewdescriptor

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocinewdescriptor — oci_new_descriptor() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_new_descriptor().

---

ocinlogon

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocinlogon — oci_new_connect() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_new_connect().

---

ocinumcols

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocinumcols — oci_num_fields() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_num_fields().

---

ociparse

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociparse — oci_parse() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_parse().

---

ociplogon

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociplogon — oci_pconnect() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_pconnect().

---

ociresult

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociresult — oci_result() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_result().

---

ocirollback

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocirollback — oci_rollback() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_rollback().

---

ocirowcount

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocirowcount — oci_num_rows() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_num_rows().

---

ocisavelob

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocisavelob — OCI-Lob->save のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Lob->save.

---

ocisavelobfile

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocisavelobfile — OCI-Lob->import のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Lob->import.

---

ociserverversion

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociserverversion — oci_server_version() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_server_version().

---

ocisetprefetch

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocisetprefetch — oci_set_prefetch() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_set_prefetch().

---

ocistatementtype

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ocistatementtype — oci_statement_type() のエイリアス

説明

この関数は次の関数のエイリアスです。 oci_statement_type().

---

ociwritelobtofile

(PHP 4, PHP 5, PECL oci8:1.0-1.2.4)

ociwritelobtofile — OCI-Lob->export のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Lob->export.

---

ociwritetemporarylob

(PHP 4 >= 4.0.6, PECL oci8:1.0)

ociwritetemporarylob — OCI-Lob->writeTemporary のエイリアス

説明

この関数は次の関数のエイリアスです。 OCI-Lob->writeTemporary.

目次

- oci_field_type_raw — Oracle におけるフィールドの型を問い合わせる
- oci_field_type — フィールドのデータ型を返す
- oci_free_statement — 文やカーソルに関連付けられた全てのリソースを解放する
- oci_internal_debug — 内部デバッグ用出力有効または無効にする
- OCI-Lob->append — ラージオブジェクトのデータを他のラージオブジェクトに追加する
- OCI-Lob->close — LOB ディスクリプタを閉じる
- oci_lob_copy — ラージオブジェクトをコピーする
- OCI-Lob->eof — ラージオブジェクトのディスクリプタが EOF かどうかを調べる
- OCI-Lob->erase — 内部 LOB データの特定の位置を消去する
- OCI-Lob->export — LOB の内容をファイルに出力する
- OCI-Lob->flush — LOB のバッファをサーバにフラッシュする、あるいは書き込む
- OCI-Lob->free — LOB ディスクリプタに関連付けられたりソースを解放する
- OCI-Lob->getBuffering — ラージオブジェクトに対する現在のバッファリングの状態を返す
- OCI-Lob->import — ファイルデータを LOB にインポートする
- oci_lob_is_equal — 2 つの LOB/FILE ロケータの等価性を比較する
- OCI-Lob->load — ラージオブジェクトの内容を返す
- OCI-Lob->read — ラージオブジェクトの一部を読み込む
- OCI-Lob->rewind — 内部ポインタをラージオブジェクトの先頭に移動する
- OCI-Lob->save — データをラージオブジェクトに保存する
- OCI-Lob->saveFile — oci_lob_import のエイリアス
- OCI-Lob->seek — ラージオブジェクトの内部ポインタをセットする
- OCI-Lob->setBuffering — 現在のラージオブジェクト用のバッファリング状態を変更する
- OCI-Lob->size — ラージオブジェクトのサイズを返す
- OCI-Lob->tell — ラージオブジェクトの内部ポインタの現在位置を返す
- OCI-Lob->truncate — ラージオブジェクトを切りつめる
- OCI-Lob->write — データをラージオブジェクトに書き込む
- OCI-Lob->writeTemporary — 一時的なラージオブジェクトを書き込む
- OCI-Lob->writeToFile — oci_lob_export のエイリアス
- oci_new_collection — 新しいコレクションオブジェクトを割り当てる
- oci_new_connect — Oracle サーバへの新規接続を確立する
- oci_new_cursor — 新規カーソル（ステートメントハンドル）を割り当て、それを返す
- oci_new_descriptor — 空の新規 LOB あるいは FILE ディスクリプタを初期化する
- oci_num_fields — ある文における結果のカラム数を返す
- oci_num_rows — 文の実行で作用された行数を取得する
- oci_parse — 実行のために Oracle の文をパースする
- oci_password_change — Oracle ユーザのパスワードを変更する
- oci_pconnect — 持続的接続を使用してOracle データベースに接続する
- oci_result — フェッチした行からフィールドの値を取得する
- oci_rollback — 未解決のトランザクションをロールバックする
- oci_server_version — サーバのバージョンを返す
- oci_set_prefetch — プリフェッチする行数を設定する
- oci_statement_type — OCI ステートメントの種類を返す
- ocibindbyname — oci_bind_by_name のエイリアス
- ocicancel — oci_cancel のエイリアス
- ocicloselob — のエイリアス
- ocicollappend — のエイリアス
- ocicollassign — のエイリアス
- ocicollassignelem — のエイリアス
- ocicollgetelem — のエイリアス
- ocicollmax — のエイリアス
- ocicollsize — のエイリアス
- ocicolltrim — のエイリアス
- ocicolumnisnull — oci_field_is_null のエイリアス
- ocicolumnname — oci_field_name のエイリアス
- ocicolumnprecision — oci_field_precision のエイリアス
- ocicolumnscale — oci_field_scale のエイリアス
- ocicolumnsize — oci_field_size のエイリアス
- ocicolumntype — oci_field_type のエイリアス
- ocicolumntyperaw — oci_field_type_raw のエイリアス
- ocicommit — oci_commit のエイリアス
- ocidefinebyname — oci_define_by_name のエイリアス
- ocierror — oci_error のエイリアス
- ociexecute — oci_execute のエイリアス
- ocifetch — oci_fetch のエイリアス

- ocifetchinto — 結果配列の次の行を取得する（非推奨）
- ocifetchstatement — oci_fetch_all のエイリアス
- ocifreecollection — のエイリアス
- ocifreecursor — oci_free_statement のエイリアス
- ocifreedesc — のエイリアス
- ocifreestatement — oci_free_statement のエイリアス
- ociinternaldebug — oci_internal_debug のエイリアス
- ociloadlob — のエイリアス
- ocilogoff — oci_close のエイリアス
- ocilogon — oci_connect のエイリアス
- ocinewcollection — oci_new_collection のエイリアス
- ocinewcursor — oci_new_cursor のエイリアス
- ocinewdescriptor — oci_new_descriptor のエイリアス
- ocinlogon — oci_new_connect のエイリアス
- ocinumcols — oci_num_fields のエイリアス
- ociparse — oci_parse のエイリアス
- ociplogon — oci_pconnect のエイリアス
- ociresult — oci_result のエイリアス
- ocirollback — oci_rollback のエイリアス
- ocirowcount — oci_num_rows のエイリアス
- ocisavelob — のエイリアス
- ocisavelobfile — のエイリアス
- ociserverversion — oci_server_version のエイリアス
- ocisetprefetch — oci_set_prefetch のエイリアス
- ocistatementtype — oci_statement_type のエイリアス
- ociwritelobtofile — のエイリアス
- ociwritetemporarylob — のエイリアス

Unified ODBC 関数

導入

通常の ODBC サポートに加えて、PHP の Unified ODBC 関数では、各々の API を実装するために ODBC API のセマンティックスを借用する複数のデータベースにアクセスすることが可能です。ほとんど同じ複数のデータベース ドライバを維持管理する代わりに、これらのドライバは単一の ODBC 関数セットに 統合されています。

以下のデータベースが Unified ODBC でサポートされています。 » Adabas D, » IBM DB2, » iODBC, » Solid, » Sybase SQL Anywhere

> **注意**: iODBC を除き、上のデータベースと接続する際には ODBC は使用しません。 内部ではネイティブに通信しており、単に ODBC 関数と同じ名前や構文を用いているだけです。 しかし、PHP を iODBC サポートつきでビルドすると、 任意の ODBC 準拠のドライバを PHP アプリケーションから使用できるようになります。 iODBC についてのより詳細な情報は » www.iodbc.org にあります。 またもうひとつの方法である unixODBC は » www.unixodbc.org にあります。

要件

サポートされるデータベースにアクセスするためには、 所定のライブラリがインストールされている必要があります。

インストール手順

--with-adabas[=DIR]

Adabas D サポートを有効にします。DIR はAdabas をインストールしたディレクトリで、 /usr/local がデフォルトです。

--with-sapdb[=DIR]

SAP DB サポートを有効にします。DIR は SAP DB のベースインストールディレクトリで、 /usr/local がデフォルトです。

--with-solid[=DIR]

Solidサポートを有効にします。DIR は Solid のベースインストールディレクトリで、 /usr/local/solid がデフォルトです。

--with-ibm-db2[=DIR]

IBM DB2 サポートを有効にします。DIR は IBM DB2 のベースインストールディレクトリで、 /home/db2inst1/sqllib がデフォルトです。

--with-empress[=DIR]

Empress サポートを有効にします。DIR は Empress のベースインストールディレクトリで、 $EMPRESSPATH がデフォルトです。PHP 4 以降、 このオプションは Empress バージョン 8.60 以降のみをサポートします。

--with-empress-bcs[=DIR]

Empress ローカルアクセスサポートを有効にします。DIR は Empress の ベースインストールディレクトリで、$EMPRESSPATH がデフォルトです。PHP 4 以降、 このオプションは Empress バージョン 8.60 以降のみをサポートします。

--with-birdstep[=DIR]

Birdstep サポートを有効にします。DIR は Birdstep のベースインストールディレクトリで、 /usr/local/birdstep がデフォルトです。

--with-custom-odbc[=DIR]

ユーザ定義の ODBC サポートを有効にします。DIR は ODBC のベースインストールディレクトリで、 /usr/local がデフォルトです。 CUSTOM_ODBC_LIBS が定義され、odbc.h がインクルードパスに あることを確認してください。 例えば、QNX上 の Sybase SQL Anywhere 5.5.00 では、configure スクリプトを実行する 前に以下を定義する必要があります。

```
CPPFLAGS="-DODBC_QNX -DSQLANY_BUG"
LDFLAGS=-lunix
CUSTOM_ODBC_LIBS="-ldblib -lodbc".
```

--with-iodbc[=DIR]

iODBC サポートを有効にします。DIR は iODBC のベースインストールディレクトリで、 /usr/local がデフォルトです。

--with-esoob[=DIR]

Easysoft OOB サポートを有効にします。DIR は OOB のベースインストールディレクトリで、 /usr/local/easysoft/oob/client がデフォルトです。

--with-unixODBC[=DIR]

unixODBC サポートを有効にします。DIR は unixODBC のベースインストールディレクトリで、 /usr/local がデフォルトです。

--with-openlink[=DIR]

OpenLink ODBC サポートを有効にします。DIR は OpenLink のベースインストールディレクトリで、 /usr/local がデフォルトです。 これは、iODBC と同じものです。

--with-dbmaker[=DIR]

DBMaker サポートを有効にします。DIR は DBMaker のベースインストールディレクトリで、 デフォルトは最新版の DBMaker がインストールされている場所 (例えば /home/dbmaker/3.6)です。

PHP 3 で unified ODBC サポートを無効にするには、 --disable-unified-odbc を configure 実行時に 指定します。このオプションは、iODBC, Adabas, Solid, Velocis custom ODBC インターフェイスを 有効にしている場合のみ適用可能です。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

php.ini の設定により動作が変化します。

Unified ODBC 設定オプション

名前	デフォルト	変更の可否	変更履歴
odbc.default_db *	NULL	PHP_INI_ALL	
odbc.default_user *	NULL	PHP_INI_ALL	
odbc.default_pw *	NULL	PHP_INI_ALL	
odbc.allow_persistent	"1"	PHP_INI_SYSTEM	
odbc.check_persistent	"1"	PHP_INI_SYSTEM	
odbc.max_persistent	"-1"	PHP_INI_SYSTEM	
odbc.max_links	"-1"	PHP_INI_SYSTEM	
odbc.defaultlrl	"4096"	PHP_INI_ALL	
odbc.defaultbinmode	"1"	PHP_INI_ALL	

注意: * マークがついているエントリは未実装です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

odbc.default_db string

odbc_connect() または odbc_pconnect()でODBCデータソースが 指定されない場合に使用される ODBC データソース。

odbc.default_user string

odbc_connect() または odbc_pconnect() で名前が指定されない場合 に使用される名前。

odbc.default_pw string

odbc_connect() または odbc_pconnect() でパスワードが指定されない場合 に使用されるパスワード。

odbc.allow_persistent boolean

持続的 ODBC 接続を許可するかどうか。

odbc.check_persistent boolean

再利用する前に接続が有効であることを確認します。

odbc.max_persistent integer

プロセス毎の持続的 ODBC 接続の最大数。

odbc.max_links integer

プロセス毎の持続的接続を含む ODBC 接続の最大数。

odbc.defaultlrl integer

LONG フィールドの処理。変数に返されるバイト数を指定します。

integerを使用する際、その値はバイト単位で測られます。 この FAQ に記載された短縮表記を使用することも可能です。
odbc.defaultbinmode integer

バイナリデータの処理モード。

リソース型

この拡張モジュールでは、ODBC 接続 ID および ODBC 結果 ID の二種類のリソースを定義しています。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

ODBC_TYPE (integer)
ODBC_BINMODE_PASSTHRU (integer)
ODBC_BINMODE_RETURN (integer)
ODBC_BINMODE_CONVERT (integer)
SQL_ODBC_CURSORS (integer)
SQL_CUR_USE_DRIVER (integer)
SQL_CUR_USE_IF_NEEDED (integer)
SQL_CUR_USE_ODBC (integer)
SQL_CONCURRENCY (integer)
SQL_CONCUR_READ_ONLY (integer)
SQL_CONCUR_LOCK (integer)
SQL_CONCUR_ROWVER (integer)
SQL_CONCUR_VALUES (integer)
SQL_CURSOR_TYPE (integer)
SQL_CURSOR_FORWARD_ONLY (integer)
SQL_CURSOR_KEYSET_DRIVEN (integer)
SQL_CURSOR_DYNAMIC (integer)
SQL_CURSOR_STATIC (integer)
SQL_KEYSET_SIZE (integer)
SQL_CHAR (integer)
SQL_VARCHAR (integer)
SQL_LONGVARCHAR (integer)
SQL_DECIMAL (integer)
SQL_NUMERIC (integer)
SQL_BIT (integer)
SQL_TINYINT (integer)
SQL_SMALLINT (integer)
SQL_INTEGER (integer)
SQL_BIGINT (integer)
SQL_REAL (integer)
SQL_FLOAT (integer)
SQL_DOUBLE (integer)
SQL_BINARY (integer)
SQL_VARBINARY (integer)
SQL_LONGVARBINARY (integer)
SQL_DATE (integer)
SQL_TIME (integer)
SQL_TIMESTAMP (integer)
SQL_TYPE_DATE (integer)
SQL_TYPE_TIME (integer)
SQL_TYPE_TIMESTAMP (integer)
SQL_BEST_ROWID (integer)
SQL_ROWVER (integer)
SQL_SCOPE_CURROW (integer)
SQL_SCOPE_TRANSACTION (integer)
SQL_SCOPE_SESSION (integer)
SQL_NO_NULLS (integer)
SQL_NULLABLE (integer)
SQL_INDEX_UNIQUE (integer)
SQL_INDEX_ALL (integer)
SQL_ENSURE (integer)
SQL_QUICK (integer)

---

odbc_autocommit

(PHP 4, PHP 5)

odbc_autocommit — 自動コミットの動作をオンまたはオフにする

説明

mixed **odbc_autocommit** (resource $connection_id [, bool $OnOff])

OnOff パラメータを指定しない場合、 この関数は、connection_id に関する 自動コミットの状態を返します。自動コミットがオンの場合に 非ゼロ、オフの場合にゼロ、エラーを生じた場合に **FALSE** を返します。

OnOff が **TRUE** の場合は自動コミットが可能であり、 **FALSE** の場合は自動コミットが使用不可となっています。 成功時に **TRUE**、失敗したときに **FALSE** を返します。

デフォルトで接続の自動コミットはオンとなっています。自動コミットを 使用不可にするのは、トランザクションを開始することと等価です。

odbc_commit() および odbc_rollback() も参照ください。

odbc_binmode

(PHP 4, PHP 5)

odbc_binmode — バイナリカラムデータを処理する

説明

bool **odbc_binmode** (resource $result_id , int $mode)

(関係するODBC SQL型: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: BINARY データとして通過
- ODBC_BINMODE_RETURN: そのまま返す
- ODBC_BINMODE_CONVERT: char に変換し返す

バイナリ SQL データが文字データに変換される時、ソースデータの 各バイト (8 ビット) は、2 つのアスキー文字で表現されます。 これらの文字は、16 進表現で数値をアスキー文字で表現したものです。 例えば、2 進数 000000001 は "01" に変換され、 11111111 は "FF" に変換されます。

LONGVARBINARY 処理

binmode	longreadlen	結果
ODBC_BINMODE_PASSTHRU	0	通過
ODBC_BINMODE_RETURN	0	通過
ODBC_BINMODE_CONVERT	0	通過
ODBC_BINMODE_PASSTHRU	0	通過
ODBC_BINMODE_PASSTHRU	>0	通過
ODBC_BINMODE_RETURN	>0	そのまま返す
ODBC_BINMODE_CONVERT	>0	char として返す

odbc_fetch_into() を使用した場合、 「通過」は空文字列が対応するカラムに返されることを意味します。

result_id に 0 を指定した場合、 ここで設定した値は、新規の結果に関するデフォルト値として用いられます。

注意: longreadlen のデフォルト値は 4096 で、binmode のデフォルト値は ODBC_BINMODE_RETURN です。 バイナリロングカラムの処理は、odbc_longreadlen() の影響も受けます。

odbc_close_all

(PHP 4, PHP 5)

odbc_close_all — 全ての ODBC 接続を閉じる

説明

void **odbc_close_all** (void)

odbc_close_all()は、 データベースサーバへの全ての接続を閉じます。

注意: この関数は、ある接続においてオープンされたトランザクションがある場合に失敗します。 この場合、この接続はオープンされたままとなります。

odbc_close

(PHP 4, PHP 5)

odbc_close — ODBC 接続を閉じる

説明

void **odbc_close** (resource $connection_id)

odbc_close() は、指定された接続 ID が指すデータベースサーバへの接続を閉じます。

注意: この関数の処理は、この接続に関してオープンされたトランザクションがある場合に失敗します。 この場合、接続はオープンされたままとなります。

odbc_columnprivileges

(PHP 4, PHP 5)

odbc_columnprivileges — カラムおよび付随する権限のリストを取得する際に使用する結果 ID を返す

説明

resource **odbc_columnprivileges** (resource $connection_id , string $qualifier , string $owner , string $table_name , string $column_name)

指定したテーブルに関するカラムおよび付随する権限のリストを取得します。 ODBC 結果 ID または失敗した場合に **FALSE** を返します。

結果は以下のカラムを有します。

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

結果セットは TABLE_QUALIFIER、TABLE_OWNER、TABLE_NAME でソートされます。

引数 column_name には検索パターン（ゼロ以上の文字にマッチする '%' や単一の文字にマッチする '_'）を使用可能です。

---

odbc_columns

(PHP 4, PHP 5)

odbc_columns — 指定したテーブルにあるカラム名のリストを取得する

説明

resource **odbc_columns** (resource $connection_id [, string $qualifier [, string $schema [, string $table_name [, string $column_name]]]])

指定した範囲の全てのカラムのリストを取得します。情報を含んでいる ODBC 結果 ID または失敗した場合に **FALSE** を返します。

結果セットは以下のカラムを有しています。

- TABLE_QUALIFIER
- TABLE_SCHEM
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

結果セットは TABLE_QUALIFIER、TABLE_SCHEM、TABLE_NAME でソートされます。

引数 schema 、table_name 、 column_name には検索パターン（ゼロ以上の文字にマッチする '%' や単一の文字にマッチする '_'）を使用可能です。

付随する権限を取得するには odbc_columnprivileges() を参照ください。

---

odbc_commit

(PHP 4, PHP 5)

odbc_commit — ODBC トランザクションをコミットする

説明

bool **odbc_commit** (resource $connection_id)

odbc_commit() は、connection_id に関するすべての実行中のトランザクションをコミットします。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

odbc_connect

(PHP 4, PHP 5)

odbc_connect — データソースに接続する

説明

resource ***odbc_connect*** *(string $dsn , string $user , string $password [, int $cursor_type])*

ODBC 接続 ID 、またはエラーの場合に 0 (**FALSE**) を返します。

この関数により返される接続 ID は、他の ODBC 関数により使用されます。 異なる db や異なる権限を使用する限り、 複数の接続を同時にオープンすることができます。オプションの 4 番目のパラメータは、 この接続で使用されるカーソルの型を設定します。 通常はこのパラメータは必要ありませんが、いくつかの ODBC ドライバの問題に対処する際には有用です。

いくつかの ODBC ドライバでは、複雑なストアド・プロシージャの 実行時に次のようなエラーにより失敗する可能性があります。 "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it" SQL_CUR_USE_ODBC を使用することにより、 このようなエラーを回避できる可能性があります。 また、いくつかのドライバは odbc_fetch_row() においてオプションの row_number パラメータをサポートしません。 この場合でも、SQL_CUR_USE_ODBC により解決できる可能性があります。

次のような定数がカーソル型として定義されています。

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

持続的な接続を行うには、odbc_pconnect() を参照ください。

---

odbc_cursor

(PHP 4, PHP 5)

odbc_cursor — カーソル名を得る

説明

string ***odbc_cursor*** *(resource $result_id)*

odbc_cursor は、指定された接続 ID (result_id) に関するカーソル名を返します。

---

odbc_data_source

(PHP 4 >= 4.3.0, PHP 5)

odbc_data_source — 現在の接続についての情報を返す

説明

array ***odbc_data_source*** *(resource $connection_id , int $fetch_type)*

エラー時には **FALSE** 、成功時には配列を返します。

この関数は、有効な DNS（複数回のコールの後）のリストを返します。 有効な ODBC 接続の connection_id が必要です。 fetch_type は次のふたつの定数 SQL_FETCH_FIRST、SQL_FETCH_NEXT のうちのどちらかです。 この関数を最初にコールする際には SQL_FETCH_FIRST を、それ以降は SQL_FETCH_NEXT を使用します。

---

odbc_do

(PHP 4, PHP 5)

odbc_do — odbc_exec() の同義語

説明

resource ***odbc_do*** *(resource $conn_id , string $query)*

odbc_do()は、指定した接続においてクエリを実行します。

---

odbc_error

(PHP 4 >= 4.0.5, PHP 5)

odbc_error — 直近のエラーコードを得る

説明

string ***odbc_error*** *([resource $connection_id])*

6 桁の ODBC ステートを返します。エラーがない場合には、空の文字列を返します。 connection_id が指定された場合、 その接続の直近の状態が返されます。そうでない場合、 他の接続の直近の状態が返されます。

この関数の返り値が意味を持つのは、直近の odbc クエリが失敗した場合（すなわち odbc_exec() が **FALSE** を返した場合）のみです。

odbc_errormsg() および odbc_exec() も参照ください。

---

odbc_errormsg

(PHP 4 >= 4.0.5, PHP 5)

odbc_errormsg — 直近のエラーメッセージを得る

説明

string **odbc_errormsg** ([resource $connection_id])

直近の ODBC エラーメッセージを含む文字列を返します。 エラーが発生していない場合は、空の文字列を返します。 connection_id が指定された場合、 その接続の直近の状態が返されます。さもなくば、他の接続の直近の状態が返されます。

この関数の返り値が意味を持つのは、直近の odbc クエリが失敗した場合（すなわち odbc_exec() が **FALSE** を返した場合）のみです。

odbc_error() および odbc_exec() も参照ください。

---

odbc_exec

(PHP 4, PHP 5)

odbc_exec — SQL文を準備し、実行する

説明

resource **odbc_exec** (resource $connection_id , string $query_string [, int $flags])

エラー時には、**FALSE**を返します。SQL コマンドの実行に成功した場合には、 ODBC 結果 ID を返します。

odbc_exec()は、SQL 命令を connection_id で指定されたデータベースサーバに送ります。 このパラメータは、odbc_connect() または odbc_pconnect() より返された有効な ID でなければなりません。

複数の SQL 命令を実行するためには、 odbc_prepare() および odbc_execute() も参照ください。

---

odbc_execute

(PHP 4, PHP 5)

odbc_execute — プリペアドステートメントを実行する

説明

bool **odbc_execute** (resource $result_id [, array $parameters_array])

odbc_prepare() で準備された命令を実行します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 配列 parameters_array は、 命令の中にパラメータを有する場合にのみ指定する必要があります。

プリペアドステートメントの中のプレースホルダが、 parameter_array 内のパラメータで順に置き換えられます。 この関数をコールした際に、配列の要素は文字列に変換されます。

parameter_array の中でシングルクォートで括られたデータがある場合、 それはファイル名と解釈されます。そのファイルの内容が、 該当するプレースホルダのデータとしてデータベースサーバに送信されます。

> **注意**: PHP 4.1.1 以降、 このファイル読み込み機能には以下のような制限が加えられるようになりました。
>
> - ファイルの読み込み機能は、 セーフモード や open-basedir による制限を一切受けていませんでした。 これは PHP 4.2.0 で修正されました。
> - リモートファイル はサポートされません。
> - シングルクォートで括られたデータを純粋に文字列として使用したい場合は、 空白などの別の文字を前後に付加する必要があります。 それにより、パラメータがファイル名とみなされることがなくなります（もしこのオプションが不要なら、別の仕組み、たとえば odbc_exec() で直接クエリを実行するなどを使用する必要があります）。

---

odbc_fetch_array

(PHP 4 >= 4.0.2, PHP 5)

odbc_fetch_array — 連想配列として結果の行を取得する

説明

array **odbc_fetch_array** (resource $result [, int $rownumber])

ODBC クエリから、連想配列を取得します。この関数が使用可能なバージョンに ついては、以下の更新履歴を参照ください。

パラメータ

result

odbc_exec() の結果リソース。

rownumber

どの行番号を取得するのかを任意で指定する。

返り値

取得した行に対応する配列を返します。もう行がない場合には **FALSE** を返します。

変更履歴

バージョン	説明
4.3.3	この関数は、IBM DB2 あるいは UnixODBC のサポートを有効にして コンパイルした場合に使用可能です。
4.3.2	この関数は、Windows 版で使用可能です。
4.0.2	この関数は DBMaker のサポートを有効にしてコンパイルした場合に 使用可能です。

参考

- *odbc_fetch_row()*
- *odbc_fetch_object()*
- *odbc_num_rows()*

---

odbc_fetch_into

(PHP 4, PHP 5)

odbc_fetch_into — 一行ぶんの結果を配列に取り込む

説明

```
int odbc_fetch_into ( resource $result_id , array &$result_array [, int $rownumber ] )
bool odbc_fetch_into ( resource $result_id [, int $rownumber ], array &$result_array )
```

結果におけるカラム番号を返します。また、エラーの場合に **FALSE** を返します。result_array は参照渡しとする必要があります。 しかし、このポインタは配列型に変換されるので、どんな型であっても構いません。 この配列には、添字 0 から始まるカラム番号が代入されます。

PHP 4.0.5 以降、result_array は参照渡しとする必要がなくなりました。

PHP 4.0.6 以降、rownumber は定数として渡すことはできず、変数として渡します。

PHP 4.2.0 以降、result_array と rownumber が交換されています。これにより、 再び rownumber を定数とすることができるようになりました。この変更は、 この関数の変更としては最後のものになるでしょう。

Example#1 odbc_fetch_into() 4.0.6 以前の例

```
<?php
$rc = odbc_fetch_into($res_id, $my_array);
?>
```

または

```
<?php
$rc = odbc_fetch_into($res_id, $row, $my_array);

$rc = odbc_fetch_into($res_id, 1, $my_array);
?>
```

Example#2 odbc_fetch_into() 4.0.6 での例

```
<?php
$rc = odbc_fetch_into($res_id, $my_array);
?>
```

または

```
<?php
$row = 1;
$rc = odbc_fetch_into($res_id, $row, $my_array);
?>
```

Example#3 odbc_fetch_into() 4.2.0 での例

```
<?php
$rc = odbc_fetch_into($res_id, $my_array);
?>
```

または

```
<?php
$rc = odbc_fetch_into($res_id, $my_array, 2);
?>
```

---

odbc_fetch_object

(PHP 4 >= 4.0.2, PHP 5)

odbc_fetch_object — オブジェクトとして結果の行を取得する

説明

object **odbc_fetch_object** (resource $result [, int $rownumber])

ODBC クエリから、オブジェクトを取得します。この関数が使用可能なバージョンに ついては、以下の更新履歴を参照ください。

パラメータ

result

odbc_exec() の結果リソース。

rownumber

どの行番号を取得するのかを任意で指定する。

返り値

取得した行に対応するオブジェクトを返します。もう行がない場合には **FALSE** を返します。

変更履歴

バージョン	説明
4.3.3	この関数は、IBM DB2 あるいは UnixODBC のサポートを有効にして コンパイルした場合に使用可能です。
4.3.2	この関数は、Windows 版で使用可能です。
4.0.2	この関数は DBMaker のサポートを有効にしてコンパイルした場合に 使用可能です。

参考

- odbc_fetch_row()
- odbc_fetch_array()
- odbc_num_rows()

---

odbc_fetch_row

(PHP 4, PHP 5)

odbc_fetch_row — 行を取り込む

説明

bool **odbc_fetch_row** (resource $result_id [, int $row_number])

odbc_fetch_row() が成功した場合（行があった場合）、 **TRUE** が返されます。もう行がない場合、**FALSE** が返されます。

odbc_fetch_row() は、odbc_do() や odbc_exec() から返された行データを取得します。 **odbc_fetch_row()** がコールされた後、この行のフィールドは、 odbc_result() でアクセス可能となります。

row_number が指定されない場合、 **odbc_fetch_row()** は、 結果セットにおける次の行を取り込もうと試みます。 row_number を指定した **odbc_fetch_row()** のコールと指定しないコールを混用することができます。

結果を複数回走査したい場合、**odbc_fetch_row()** を row_number に 1 を指定してコールし、 続いて結果を再度見るために row_number を指定せずに **odbc_fetch_row()** を実行しつづけます。 行を番号で取り込むことをドライバがサポートしていない場合、 row_number パラメータは無視されます。

---

odbc_field_len

(PHP 4, PHP 5)

odbc_field_len — フィールドの長さ（精度）を得る

説明

int **odbc_field_len** (resource $result_id , int $field_number)

odbc_field_len() は、指定した ODBC 結果 ID の番号で参照されるフィールドの長さを返します。 フィールド番号は 1 から始まります。

浮動小数点の精度を得るには、odbc_field_scale() も参照ください。

---

odbc_field_name

(PHP 4, PHP 5)

odbc_field_name — カラム名を得る

説明

string **odbc_field_name** (resource $result_id , int $field_number)

odbc_field_name() は、指定された ODBC 結果 ID において指定されたカラム番号にあるフィールドの名前を返します。 フィールド番号は、1 から始まります。エラーの場合、 **FALSE** を返します。

---

odbc_field_num

(PHP 4, PHP 5)

odbc_field_num — カラム番号を返す

説明

int **odbc_field_num** (resource $result_id , string $field_name)

odbc_field_num() は、指定したODBC 結果 ID におけるフィールド名に対応するカラムスロットの数を返します。 フィールド番号は、1 から始まります。エラーの場合、 **FALSE** を返します。

---

odbc_field_precision

(PHP 4, PHP 5)

odbc_field_precision — odbc_field_len() の同義語

説明

int **odbc_field_precision** (resource $result_id , int $field_number)

odbc_field_precision() は、指定した ODBC 結果 ID において番号で指定されたフィールドの精度を返します。

浮動小数点の精度を得るには、odbc_field_scale() も参照ください。

---

odbc_field_scale

(PHP 4, PHP 5)

odbc_field_scale — フィールドの精度を得る

説明

int **odbc_field_scale** (resource $result_id , int $field_number)

odbc_field_scale() は、指定した ODBC 結果 ID の番号で指定したフィールドの精度を返します。

---

odbc_field_type

(PHP 4, PHP 5)

odbc_field_type — フィールドのデータ型を返す

説明

string **odbc_field_type** (resource $result_id , int $field_number)

odbc_field_type() は、指定された ODBC 結果 ID において指定番号で参照されるフィールドの SQL 型を返します。 フィールド番号は 1 から始まります。

---

odbc_foreignkeys

(PHP 4, PHP 5)

odbc_foreignkeys — 指定したテーブルの外部キーのリストまたは指定したテーブルの主キーを 参照する他のテーブルの外部キーのリストを返す

説明

resource **odbc_foreignkeys** (resource $connection_id , string $pk_qualifier , string $pk_owner , string $pk_table , string $fk_qualifier , string $fk_owner , string $fk_table)

odbc_foreignkeys() は外部キーに関する情報を取得します。 ODBC 結果 ID または失敗した場合に **FALSE** を返します。

結果は以下のカラムを有します。

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE
- FK_NAME
- PK_NAME

pk_table がテーブル名を有している場合、 **odbc_foreignkeys()** は指定したテーブルの主キー およびそのキーを参照する全ての外部キーのリストを結果として返します。

fk_table がテーブル名を有している場合、 **odbc_foreignkeys()** は指定したテーブルにある全ての 外部キーおよびそのキーが参照する(他のテーブルの)主キーのリストを 結果として返します。

pk_table および fk_table が共にテーブル名を有している場合、 **odbc_foreignkeys()** は pk_table で指定されたテーブルの主キーを参照する fk_table で指定されたテーブルの外部キーを返します。 返されるキーは最大でも一つだけです。

---

odbc_free_result

(PHP 4, PHP 5)

odbc_free_result — 結果を保持するリソースを開放する

説明

bool **odbc_free_result** (resource $result_id)

常に **TRUE** を返します。

odbc_free_result()は、スクリプトのメモリ消費量が 多すぎることが懸念される場合にのみコールする必要があります。 全ての結果保持用メモリは、スクリプト実行が終了した際に自動的に 開放されます。しかし、結果データをスクリプトでもはや必要としないことが 確実な場合、**odbc_free_result()** をコールして result_id が指すメモリを開放することが できます。

> **注意**: 自動コミットが無効（odbc_autocommit() を 参照ください）な時にコミットの前に **odbc_free_result()** をコールすると、全ての 未解決のトランザクションは、ロールバックされます。

---

odbc_gettypeinfo

(PHP 4, PHP 5)

odbc_gettypeinfo — データソースによりサポートされるデータ型に関する情報を有する結果 ID を返す

説明

resource **odbc_gettypeinfo** (resource $connection_id [, int $data_type])

データソースによりサポートされるデータ型に関する情報を取得します。 ODBC 結果 ID または失敗した場合に **FALSE** を返します。 オプションの引数 data_type は単一のデータ型に情報を制限するために使用することが可能です。

結果は以下のカラムを有します。

- TYPE_NAME
- DATA_TYPE
- PRECISION
- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY
- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE
- MAXIMUM_SCALE

結果は、DATA_TYPE および TYPE_NAME でソートされます。

---

odbc_longreadlen

(PHP 4, PHP 5)

odbc_longreadlen — LONG カラムを処理する

説明

bool **odbc_longreadlen** (resource $result_id , int $length)

(影響する ODBC SQL 型: LONG, LONGVARBINARY) PHP に返されるバイト数は、パラメータ length により制御されます。 これを 0 をセットした場合、ロングカラムデータは、 クライアントにそのまま渡されます。

注意: LONGVARBINARY カラムの処理は、odbc_binmode() にも影響を受けます。

---

odbc_next_result

(PHP 4 >= 4.0.5, PHP 5)

odbc_next_result — 複数の結果が利用可能などうか確認する

説明

bool **odbc_next_result** (resource $result_id)

odbc_next_result() は、まだ結果セットが存在して odbc_fetch_array()、odbc_fetch_row()、 odbc_result() などで次の結果セットにアクセスできる場合に **TRUE** を返します。

この関数は、エラーが発生した場合に **FALSE** を返します。

引数に指定するのは、odbc_exec() が返す結果 ID です。

Example#1 odbc_next_result()

```
<?php
$r_Connection = odbc_connect($dsn, $username, $password);

$s_SQL = <<<END_SQL
SELECT 'A'
SELECT 'B'
SELECT 'C'
END_SQL;

$r_Results = odbc_exec($r_Connection, $s_SQL);

$a_Row1 = odbc_fetch_array($r_Results);
$a_Row2 = odbc_fetch_array($r_Results);
echo "最初の結果セットを出力します";
var_dump($a_Row1, $a_Row2);

echo "二番目の結果セットを取得します ";
var_dump(odbc_next_result($r_Results));

$a_Row1 = odbc_fetch_array($r_Results);
$a_Row2 = odbc_fetch_array($r_Results);
echo "二番目の結果セットを出力します ";
var_dump($a_Row1, $a_Row2);

echo "三番目の結果セットを取得します ";
var_dump(odbc_next_result($r_Results));

$a_Row1 = odbc_fetch_array($r_Results);
$a_Row2 = odbc_fetch_array($r_Results);
echo "三番目の結果セットを出力します ";
var_dump($a_Row1, $a_Row2);

echo "四番目の結果セットを調べてみます ";
var_dump(odbc_next_result($r_Results));
?>
```

上の例の出力は以下となります。

```
最初の結果セットを出力します array(1) {
   ["A"]=>
   string(1) "A"
}
bool(false)
二番目の結果セットを取得します bool(true)
二番目の結果セットを出力します array(1) {
   ["B"]=>
   string(1) "B"
}
bool(false)
三番目の結果セットを取得します bool(true)
三番目の結果セットを出力します array(1) {
   ["C"]=>
   string(1) "C"
}
bool(false)
四番目の結果セットを調べてみます bool(false)
```

---

odbc_num_fields

(PHP 4, PHP 5)

odbc_num_fields — 結果のカラム数を返す

説明

int **odbc_num_fields** (resource $result_id)

odbc_num_fields()は、ODBC 結果におけるフィールド（カラム）の数を返します。この関数は、エラー時に-1を返します。 引数は、odbc_exec() より返された有効な結果 ID です。

---

odbc_num_rows

(PHP 4, PHP 5)

odbc_num_rows — 結果における行数を返す

説明

int **odbc_num_rows** (resource $result_id)

odbc_num_rows()は、ODBC 結果における行の数を返します。 この関数は、エラー時に-1を返します。 INSERT、UPDATE および DELETE 命令に関しては、 **odbc_num_rows()** は、変更された行の数を返します。 SELECT 文の場合、この関数は利用可能な行の数を返します。

注意: SELECT の後に利用可能な行の数を定義するために **odbc_num_rows()** を使用した場合、 多くのドライバにおいて、-1 が返されます。

---

odbc_pconnect

(PHP 4, PHP 5)

odbc_pconnect — 持続的なデータベース接続を開く

説明

resource **odbc_pconnect** (string $dsn , string $user , string $password [, int $cursor_type])

ODBC 接続 ID またはエラー時に 0 (**FALSE**) を返します。 この関数は、スクリプトが終了した時に接続が閉じられないということ以外は odbc_connect() に非常に似ています。 同じ dsn 、user 、 password の組み合わせ (odbc_connect() および **odbc_pconnect()** による) 接続の場合は、 持続的な接続を再利用することが可能です。

> **注意**: 持続的な接続 は、PHP が CGI プログラムとして使用される場合は使用できません。

オプションの cursor_type パラメータに関する情報については、 odbc_connect() 関数を参照ください。 持続的接続に関する更に詳細な情報については、PHP FAQ を参照ください。

---

odbc_prepare

(PHP 4, PHP 5)

odbc_prepare — 実行用に文を準備する

説明

resource **odbc_prepare** (resource $connection_id , string $query_string)

エラー時には、**FALSE** を返します。

SQL コマンドの準備に成功した場合は、ODBC 結果 ID を返します。 結果 ID は、この後 odbc_execute() で命令を実行する際に使用することができます。

(IBM DB2、MS SQL Server および Oracle のように) ストアドプロシージャが利用可能で、ODBC 仕様で定義されている IN、INOUT および OUT が利用できるものもあります。 しかし、Unified ODBC ドライバでは現在 IN 型のパラメータしかサポートしていません。

以下のコードでは、$res が有効な値になるのは myproc のすべてのパラメータが IN パラメータの場合のみです。

```
<?php
$a = 1;
$b = 2;
$c = 3;
$stmt = odbc_prepare($conn, 'CALL myproc(?,?,?)');
$res = odbc_execute($stmt, array($a, $b, $c));
?>
```

INOUT や OUT 型のパラメータを持つストアドプロシージャをコールする必要がある場合に推奨される方法は、 データベースのネイティブ拡張モジュール (例: MS SQL Server なら mssql、 あるいは Oracle なら oci8) を使用することです。

---

odbc_primarykeys

(PHP 4, PHP 5)

odbc_primarykeys — テーブルの主キーを有するカラムの名前を取得する際に使用可能な結果 ID を返す

説明

resource **odbc_primarykeys** (resource $connection_id , string $qualifier , string $owner , string $table)

テーブルの主キーを有するカラムの名前を返します。ODBC 結果 ID または 失敗した場合に **FALSE** を返します。

結果は以下のカラムを有します。

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

---

odbc_procedurecolumns

(PHP 4, PHP 5)

odbc_procedurecolumns — プロシージャへのパラメータに関する情報を取得する

説明

resource **odbc_procedurecolumns** (resource $connection_id [, string $qualifier], string $owner , string $proc , string $column)

指定したプロシージャに関して入出力パラメータのリストとその結果を構成するカラムを返します。 ODBC 結果 ID または失敗した場合に **FALSE** を返します。

結果は以下のカラムを有します。

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

結果セットは、PROCEDURE_QUALIFIER、PROCEDURE_OWNER、 PROCEDURE_NAME、COLUMN_TYPE でソートされます。

引数 owner 、proc 、 column には検索パターン（ゼロ以上の文字にマッチする '%' や単一の文字にマッチする '_'）を使用可能です。

---

odbc_procedures

(PHP 4, PHP 5)

odbc_procedures — 指定したデータソースに保存されているプロシージャのリストを取得する

説明

resource **odbc_procedures** (resource $connection_id [, string $qualifier], string $owner , string $name)

指定した範囲の全てのプロシージャのリストを取得します。ODBC 結果 ID または失敗した場合に **FALSE** を返します。

結果は以下のカラムを有します。

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS

- REMARKS
- PROCEDURE_TYPE

引数 owner および name には検索パターン（ゼロ以上の文字にマッチする '%' や単一の文字にマッチする '_'）を使用可能です。

---

odbc_result_all

(PHP 4, PHP 5)

odbc_result_all — HTML テーブルとして結果を出力する

説明

int **odbc_result_all** (resource $result_id [, string $format])

成功の場合に結果の行数、エラーの場合に **FALSE** を返します。

odbc_result_all() は、 odbc_exec() により作成された結果 ID から全ての行を出力します。結果は、HTML テーブル形式で出力されます。 オプションのフォーマット 文字列により、 テーブルのあらゆるフォーマット指定が可能です。

---

odbc_result

(PHP 4, PHP 5)

odbc_result — 結果データを得る

説明

mixed **odbc_result** (resource $result_id , mixed $field)

フィールドの内容を表す文字列を返します。エラー時には **FALSE** 、NULL データの場合は **NULL** 、そしてバイナリデータの場合は **TRUE** を返します。

field には、取得するフィールドのカラム番号を 表す整数またはフィールド名を表す文字列のどちらを指定できます。 例えば次のようにします。

```
<?php
$item_3  = odbc_result($Query_ID, 3);
$item_val  = odbc_result($Query_ID, "val");
?>
```

最初の **odbc_result()** コールにより、クエリー結果の 現在のレコードにおける 3 番目のフィールドの値が返されます。 2 番目の **odbc_result()** コールにより、クエリーの 結果の現在のレコードにおける "val" というフィールド名のフィールドの 値を返します。あるフィールドのカラム番号パラメータが 1 より小さいか 現在のレコードのカラム(またはフィールド)の数を超える場合、 エラーを生じます。同様に、クエリーを行ったテーブルのフィールド名には ない名前をフィールドとして指定した場合にもエラーを生じます。

フィールドインデックスは 1 から始まります。 バイナリまたはロングカラムデータの返し方に関しては、 odbc_binmode() および odbc_longreadlen() を参照ください。

---

odbc_rollback

(PHP 4, PHP 5)

odbc_rollback — トランザクションをロールバックする

説明

bool **odbc_rollback** (resource $connection_id)

connection_id における全ての未解決の命令をロールバックします。 成功した場合に **TRUE** 、失敗した場合に **FALSE** を返します。

---

odbc_setoption

(PHP 4, PHP 5)

odbc_setoption — ODBC の設定を変更する

説明

bool **odbc_setoption** (resource $id , int $function , int $option , int $param)

この関数により特定の接続またはクエリー結果に関する ODBC オプションを変更することが可能となります。この関数は、気まぐれな ODBC ドライバの問題に対処する際の道具として作成されたものです。自分が ODBC プログラマであり、様々なオプションが有する効果を理解している場合にのみ この関数を使用するべきでしょう。 使用可能な全てのオプションおよび値を理解するためには、良い ODBC リファレンスが必要です。ドライバーのバージョンが違うと、 サポートされるオプションも異なります。

この関数の効果は ODBC ドライバに依存する可能性があるので、 この関数を一般に公開するスクリプトで使用することは、 必ず避けるべきです。 また、いくつかの ODBC オプションはこの関数では利用できません。 それは、これらを接続の確立またはクエリの準備の前に設定する必要があるからです。 しかし、特定の業務において自分の上司が商用製品の使用を指示しなかった ために PHP を使用する場合、このことは実際的な問題となります。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

id は設定を変更する接続 ID または 結果 ID です。SQLSetConnectOption() の場合、これは接続 ID です。 SQLSetStmtOption() の場合、これは結果 ID です。

Function は使用する ODBC 関数です。 値は SQLSetConnectOption() の場合 1、 SQLSetStmtOption() の場合 2 である必要があります。

パラメータ option は設定するオプションです。

パラメータ param は指定した option の値です。

Example#1 ODBC Setoption の例

```
<?php
// 1. SQLSetConnectOption() のオプション 102 は SQL_AUTOCOMMIT です。
//      SQL_AUTOCOMMIT の値 1 は SQL_AUTOCOMMIT_ON です。
//      この例は odbc_autocommit($conn, true); と同じ結果
//    となります。

odbc_setoption($conn, 1, 102, 1);

// 2. SQLSetStmtOption() のオプション 0 は SQL_QUERY_TIMEOUT です。
//    この例は 30 秒後に時間切れとなるクエリーを設定します。

$result = odbc_prepare($conn, $sql);
odbc_setoption($result, 2, 0, 30);
odbc_execute($result);
?>
```

---

odbc_specialcolumns

(PHP 4, PHP 5)

odbc_specialcolumns — テーブルのレコードを特定する最適なカラムの組合せ、 またはレコードの値がトランザクションにより更新される際に自動的に更新されるカラムを返す

説明

*resource **odbc_specialcolumns** (resource $connection_id , int $type , string $qualifier , string $owner , string $table , int $scope , int $nullable)*

*引数 type が SQL_BEST_ROWID の場合、 **odbc_specialcolumns()** はテーブルの各レコードを特定するカラムを返します。*

*引数 type が SQL_ROWVER の場合、 **odbc_specialcolumns()** はカラムから値を取得する ことにより指定したテーブルでレコードを特定できる最適なカラムまた はカラムの組を返します。*

*ODBC 結果 ID または失敗した場合に **FALSE** を返します。*

結果は以下のカラムを有します。

- *SCOPE*
- *COLUMN_NAME*
- *DATA_TYPE*
- *TYPE_NAME*
- *PRECISION*
- *LENGTH*
- *SCALE*
- *PSEUDO_COLUMN*

結果セットは SCOPE でソートされます。

---

odbc_statistics

(PHP 4, PHP 5)

odbc_statistics — テーブルに関する統計情報を取得する

説明

*resource **odbc_statistics** (resource $connection_id , string $qualifier , string $owner , string $table_name , int $unique , int $accuracy)*

*テーブルおよびインデックスに関する統計情報を取得します。ODBC 結果 ID または失敗した場合に **FALSE** を返します。*

結果は以下のカラムを有します。

- *TABLE_QUALIFIER*
- *TABLE_OWNER*
- *TABLE_NAME*
- *NON_UNIQUE*
- *INDEX_QUALIFIER*
- *INDEX_NAME*
- *TYPE*
- *SEQ_IN_INDEX*

- COLUMN_NAME
- COLLATION
- CARDINALITY
- PAGES
- FILTER_CONDITION

結果セットは、NON_UNIQUE、TYPE、INDEX_QUALIFIER、 INDEX_NAME、SEQ_IN_INDEX でソートされます。

---

odbc_tableprivileges

(PHP 4, PHP 5)

odbc_tableprivileges — 各テーブルのリストおよび関連する権限のリストを取得する

説明

resource **odbc_tableprivileges** (resource $connection_id , string $qualifier , string $owner , string $name)

指定した範囲にあるテーブルおよび各テーブルに関連する権限のリストを取得します。 ODBC 結果 ID または、失敗した場合に **FALSE** を返します。

結果は次のカラムを有します。

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

結果セットは TABLE_QUALIFIER、TABLE_OWNER、TABLE_NAME の順番になります。

引数 owner および name には検索パターン（ゼロ以上の文字にマッチする '%' や単一の文字にマッチする '_'）を使用可能です。

---

odbc_tables

(PHP 4, PHP 5)

odbc_tables — 指定したデータソースに保存されたテーブルの名前のリストを取得する

説明

resource **odbc_tables** (resource $connection_id [, string $qualifier [, string $owner [, string $name [, string $types]]]])

指定した範囲の全てのテーブルリストを得ます。情報を含んでいる ODBC 結果 ID または失敗した際に **FALSE** を返します。

結果は以下のカラムを有します。

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

結果セットは TABLE_TYPE、TABLE_QUALIFIER、TABLE_OWNER、TABLE_NAME の順番になります。

引数 owner および name には、検索パターン（ゼロ以上の文字にマッチする '%' や単一の文字にマッチする '_'）を指定可能です。

限定子、所有者、テーブル名を数えるために、 qualifier 、owner 、 name 、table_type について以下のような特別な記号が使用可能です。

- qualifier がパーセント記号（%）のみで、 owner および name が空文字列であった場合、結果にはそのデータソースに関する 有効な限定子のリスト（**NULL** を有する TABLE_QUALIFIER カラム以外の 全てのカラム)が含まれます。
- owner がパーセント記号（%）のみで、 qualifier および name が空文字列の場合、結果にはその データソースに関する有効な所有者のリスト（**NULL** を有する TABLE_OWNER カラム以外の全てのカラム）が含まれます。
- table_type がパーセント記号（%）のみで、 qualifier 、owner 、 name が空文字列の場合、結果にはそのデータソースに 関する有効なテーブル型のリスト（**NULL** を有する TABLE_TYPE カラム以外の全てのカラム）が含まれます。

table_type が空の文字列ではない場合、検索する型に ついてカンマで区切った値のリストを指定する必要があります。 各値は、シングルクオート(')で括るかまたは括らない形で指定可能です。 例えば、"'TABLE','VIEW'" または "TABLE, VIEW" となります。データソースが が指定したテーブル型をサポートしていない場合、 **odbc_tables()** はその型についていかなる結果も 返しません。

関連する権限の取得については、 odbc_tableprivileges() も参照ください。

目次

- odbc_autocommit — 自動コミットの動作をオンまたはオフにする
- odbc_binmode — バイナリカラムデータを処理する
- odbc_close_all — 全ての ODBC 接続を閉じる
- odbc_close — ODBC 接続を閉じる
- odbc_columnprivileges — カラムおよび付随する権限のリストを取得する際に使用する結果 ID を返す
- odbc_columns — 指定したテーブルにあるカラム名のリストを取得する
- odbc_commit — ODBC トランザクションをコミットする
- odbc_connect — データソースに接続する
- odbc_cursor — カーソル名を得る
- odbc_data_source — 現在の接続についての情報を返す
- odbc_do — odbc_exec の同義語
- odbc_error — 直近のエラーコードを得る
- odbc_errormsg — 直近のエラーメッセージを得る
- odbc_exec — SQL文を準備し、実行する
- odbc_execute — プリペアドステートメントを実行する
- odbc_fetch_array — 連想配列として結果の行を取得する
- odbc_fetch_into — 一行ぶんの結果を配列に取り込む
- odbc_fetch_object — オブジェクトとして結果の行を取得する
- odbc_fetch_row — 行を取り込む
- odbc_field_len — フィールドの長さ (精度) を得る
- odbc_field_name — カラム名を得る
- odbc_field_num — カラム番号を返す
- odbc_field_precision — odbc_field_len の同義語
- odbc_field_scale — フィールドの精度を得る
- odbc_field_type — フィールドのデータ型を返す
- odbc_foreignkeys — 指定したテーブルの外部キーのリストまたは指定したテーブルの主キーを 参照する他のテーブルの外部キーのリストを返す
- odbc_free_result — 結果を保持するリソースを開放する
- odbc_gettypeinfo — データソースによりサポートされるデータ型に関する情報を有する結果 ID を返す
- odbc_longreadlen — LONG カラムを処理する
- odbc_next_result — 複数の結果が利用可能などうか確認する
- odbc_num_fields — 結果のカラム数を返す
- odbc_num_rows — 結果における行数を返す
- odbc_pconnect — 持続的なデータベース接続を開く
- odbc_prepare — 実行用に文を準備する
- odbc_primarykeys — テーブルの主キーを有するカラムの名前を取得する際に使用可能な結果 ID を返す
- odbc_procedurecolumns — プロシージャへのパラメータに関する情報を取得する
- odbc_procedures — 指定したデータソースに保存されているプロシージャのリストを取得する
- odbc_result_all — HTML テーブルとして結果を出力する
- odbc_result — 結果データを得る
- odbc_rollback — トランザクションをロールバックする
- odbc_setoption — ODBC の設定を変更する
- odbc_specialcolumns — テーブルのレコードを特定する最適なカラムの組合せ、 またはレコードの値がトランザクションにより更新される際に自動的に更新されるカラムを返す
- odbc_statistics — テーブルに関する統計情報を取得する
- odbc_tableprivileges — 各テーブルのリストおよび関連する権限のリストを取得する
- odbc_tables — 指定したデータソースに保存されたテーブルの名前のリストを取得する

ODBC および DB2 関数 (PDO_ODBC)

導入

PDO_ODBC は PHP Data Objects (PDO) インターフェイスを実装したドライバで、 PHP から ODBC ドライバあるいは IBM DB2 Call Level Interface (DB2 CLI) ライブラリを使用した データベースへのアクセスが可能となります。 PDO_ODBC は、現在 3 種類のデータベースドライバをサポートしています。

ibm-db2

DB2 クライアントを使用した、 IBM DB2 Universal Database、Cloudscape および Apache Derby サーバへのアクセスをサポートします。

unixODBC

unixODBC ドライバマネージャおよびデータベースごとの ODBC ドライバを使用した、データベースサーバへのアクセスをサポートします。

generic

PDO_ODBC が明示的にサポートしていない ODBC ドライバマネージャのためのコンパイルオプションを提供します。

Windows では、PDO_ODBC はデフォルトで PHP コアに組み込まれています。 これは Windows ODBC ドライバマネージャに対してリンクされてお

り、 システムの DSN に登録されているあらゆるデータベースに対して PHP から接続することができます。Microsoft SQL Server データベースに接続する際には、このドライバの使用を推奨します。

インストール手順

UNIX システムでの PDO_ODBC

1. PHP 5.1 では、PDO_ODBC は PHP ソースの中に含まれています。 PDO_ODBC 拡張モジュールを静的モジュールあるいは共有モジュールとしてコンパイルするには次のような **configure** コマンドを実行します。

   ibm_db2

   ```
   ./configure  --with-pdo-odbc=ibm-db2,/opt/IBM/db2/V8.1/
   ```

   PDO_ODBC を ibm-db2 形式でビルドするには、PDO_ODBC をコンパイルするのと同じマシンに DB2 アプリケーション開発用ヘッダが インストールされていなければなりません。DB2 アプリケーション開発ヘッダは DB2 サーバのインストールオプションに 含まれており、また IBM DB2 Universal Database の » サポートサイト からフリーでダウンロードできる DB2 Application Development Client にも含まれています。

   configure コマンドに DB2 ライブラリおよび ヘッダの場所を指定しなかった場合は、PDO_ODBC はデフォルトとして /home/db2inst1/sqllib を使用します。

   unixODBC

   ```
   ./configure  --with-pdo-odbc=unixODBC,/usr/local
   ```

   configure コマンドに unixODBC ライブラリおよび ヘッダの場所を指定しなかった場合は、PDO_ODBC はデフォルトとして /usr/local を使用します。

   generic

   ```
   ./configure  --with-pdo-odbc=generic,/usr/local,libname,ldflags,cflags
   ```

実行時設定

php.ini の設定により動作が変化します。

PDO_ODBC 設定オプション

名前	デフォルト	変更の可否	変更履歴
pdo_odbc.connection_pooling	"strict"	PHP_INI_ALL	PHP 5.1.0 以降で使用可能です。
pdo_odbc.db2_instance_name	NULL	PHP_INI_SYSTEM	PHP 5.1.1 以降で使用可能です。PHP 6.0.0 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

pdo_odbc.connection_pooling string

ODBC 接続のプール方法を指定します。"strict"、"relaxed" あるいは "off" ("" と同じ) のいずれかです。このパラメータは、 プールされている既存の接続との比較をどの程度厳密に行うのかを 接続マネージャに指定します。 **strict** は推奨されているデフォルト設定で、 接続パラメータが完全に一致する場合にのみ既存の接続を使用します。 **relaxed** は、接続パラメータが似ている場合に 既存の接続を使用します。これはキャッシュの再利用率を高めますが、 (例えば) 仮想ホスト間での接続情報がおかしくなってしまう恐れがあります。

この設定は php.ini ファイルでのみ変更可能で、 その内容はプロセス全体に影響します。同じ ODBC ライブラリを使用している すべてのモジュール、たとえば Unified ODBC 拡張モジュール などが影響を受けます。

警告

relaxed を共有サーバで使用してはいけません。 これはセキュリティの問題があるためです。

ヒント

どうしても変更する必要がない限り、この設定はデフォルトの **strict** のままにしておきましょう。

pdo_odbc.db2_instance_name string

db2 方式で PDO_ODBC をコンパイルした場合、 Linux および UNIX 上で DB2 インスタンスを指定するための環境変数 DB2INSTANCE の値を設定します。これによって PDO_ODBC が DB2 ライブラリの場所を見つけられるようになり、 DB2 データベースへのカタログ接続が可能になります。

この設定は php.ini ファイルでのみ変更可能で、 その内容はプロセス全体に影響します。同じ ODBC ライブラリを使用している すべてのモジュール、たとえば Unified ODBC 拡張モジュール などが影響を受けます。

この設定は、Windows 上では何の意味も持ちません。

PDO_ODBC DSN

(No version information available, might be only in CVS)

PDO_ODBC DSN — ODBC あるいは DB2 データベースに接続する

説明

PDO_ODBC データソース名 (DSN) は以下の要素で構成されます。

DSN 接頭辞

DSN 接頭辞は **odbc:** です。 ODBC ドライバマネージャや DB2 カタログに登録されている データベースに接続する際には、登録されている名前を DSN に追加することができます。

DSN

ODBC ドライバマネージャあるいは DB2 カタログに登録されている データベース名を指定します。それ以外の方法として、 » http://www.connectionstrings.com/ で説明されている完全な ODBC 接続文字列を使用して データベースに接続することもできます。

UID

接続に使用するユーザの名前を指定します。DSN と同じユーザ名を指定した場合は、PDO はコンストラクタの引数で指定された ユーザ名を無視します。

PWD

接続するユーザのパスワードを指定します。DSN と同じパスワードを指定した場合は、PDO はコンストラクタの引数で指定された パスワードを無視します。

例

Example#1 PDO_ODBC DSN の例 (ODBC ドライバマネージャ)

以下の例は、ODBC ドライバマネージャで testdb として登録されている ODBC データベースに接続するための PDO_ODBC DSN を表します。

```
odbc:testdb
```

Example#2 PDO_ODBC DSN の例 (IBM DB2 非カタログ接続)

以下の例は、完全な ODBC DSN 構文を使用して **SAMPLE** という名前の IBM DB2 データベースに接続するための PDO_ODBC DSN を表します。

```
odbc:DRIVER={IBM DB2 ODBC DRIVER};HOSTNAME=localhost;PORT=50000;DATABASE=SAMPLE;PROTOCOL=TCPIP;UID=db2inst1;PWD=ibmdb2;
```

Example#3 PDO_ODBC DSN の例 (Microsoft Access 非カタログ接続)

以下の例は、完全な ODBC DSN 構文を使用して **C:¥db.mdb** にある Microsoft Access データベースに接続するための PDO_ODBC DSN を表します。

```
odbc:Driver={Microsoft Access Driver (*.mdb)};Dbq=C:¥¥db.mdb;Uid=Admin
```

---

oggvorbis

導入

OGG/Vorbis ファイルフォーマットは » http://www.vorbis.com/ で定義されており、 音質の劣化を最小限に抑えた圧縮オーディオストリームを提供する仕組みです。 この拡張モジュールは、PHP の URL ラッパ を Ogg Vorbis に対応させます。読み込みモードで使用した場合は、以下の表の 6 つの PCM エンコーディングフォーマットのうちひとつを用いて OGG/Vorbis 圧縮データを PCM オーディオに展開します。

要件

この拡張モジュールは、PHP >= 4.3.0、 » libogg >= 1.0 および » libvorbis >= 1.0 を必要とします。

実行時設定

設定ディレクティブは定義されていません。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/oggvorbis

Context options

OGG/Vorbis のチューニングオプション

オプション	定義	モード	デフォルト
pcm_mode	使用する PCM バイトエンコーディング形式。以下の定数を参照ください。	Read / Write	**OGGVORBIS_PCM_S16_LE**
rate	PCM サンプリングレート。Hz 単位。	Write only	**44100**
bitrate	Vorbis 平均ビットレート（ABR）/ 可変ビットレート（VBR）。 bps 単位（ABR）あるいは品質レベル（VBR: 0.0 から 1.0）。 128000 ABR と 0.4 VBR がほぼ同じレベルです。	Write only	**128000**
channels	PCM チャネル数。1 == モノラル、2 == ステレオ。	Write only	**2**
serialno	ファイル内のストリームのシリアル番号。ファイル内で一意である必要があります。連結したファイル内でシリアル番号が重複してしまう可能性があるため、エンコードの際には手動で一意の番号を割り当てるよう注意して ください。	Write only	*Random*
comments	ファイルについてのコメントの連想配列。 *strtoupper($name) . "=$value"* 形式に変換されます。 注意: このオプションは、oggvorbis-0.1 では使用できません。	Write only	**array('ENCODER' => 'PHP/OggVorbis, http://pear.php.net/oggvorbis')**

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

OGG/Vorbis は、以下のフォーマットの PCM エンコーディングをサポートします

定数	定義
OGGVORBIS_PCM_U8	符号なし 8 ビット PCM。
OGGVORBIS_PCM_S8	符号付き 8 ビット PCM。
OGGVORBIS_PCM_U16_LE	符号なし 16 ビット PCM。リトルエンディアンバイトオーダー。
OGGVORBIS_PCM_U16_BE	符号なし 16 ビット PCM。ビッグエンディアンバイトオーダー。
OGGVORBIS_PCM_S16_LE	符号付き 16 ビット PCM。リトルエンディアンバイトオーダー。
OGGVORBIS_PCM_S16_BE	符号付き 16 ビット PCM。ビッグエンディアンバイトオーダー。

使用方法

(No version information available, might be only in CVS)

使用方法 — ogg:// ラッパの使用例

例

Example#1 OGG/Vorbis ファイルを読み込む

```
<?php
dl("oggvorbis.so");

/* デフォルトでは、ogg:// は符号付き 16 ビット リトルエンディアンにデコードします。*/
$fp = fopen('ogg://myaudio.ogg', 'r');

/* ファイルの情報を取得します。*/
$metadata = stream_get_meta_data($fp);

/* 最初の曲を調べます（たいていは 1 曲だけですが、
   OGG/Vorbis ファイルは連結が可能です）。*/
$songdata = $metadata['wrapper_data'][0];

echo "OGG/Vorbis file encoded by: {$songdata['vendor']}¥n.";
echo "   {$songdata['channels']} channels of {$songdata['rate']}Hz sampling encoded at {$songdata['bitrate_nominal']}bps.¥
foreach($songdata['comments'] as $comment) {
    echo "   $comment¥n";
}

while ($audio_data = fread($fp, 8192)) {
  /* OGG から展開した PCM オーディオについて、なんらかの処理を行います。
     linux システムなら、たとえば /dev/dsp にコピーするなどがよい例でしょう。
     まず最初にデバイスの設定をすることを忘れないようにしましょう。*/
}

fclose($fp);

?>
```

Example#2 オーディオファイルを OGG/Vorbis にエンコードする

```
<?php
dl('oggvorbis.so');

$context  = stream_context_create(array('ogg'=>array(
              'pcm_mode'  => OGGVORBIS_PCM_S8,  /* 符号付き 8 ビットオーディオ */
              'rate'  => 44100,                  /* 44kHz CD クオリティ */
              'bitrate'  => 0.5,                /* 中品質の VBR（可変ビットレート） */
              'channels'  => 1,                  /* モノラル */
              'serialno'  => 12345)));          /* ストリーム内で一意の番号 */

/* ファイルを追記モードでオープンします。これは、
2 番目の OGG ストリームを最初のストリームの後に連結します。*/
$ogg  = fopen('ogg://mysong.ogg', 'a', false, $context);

$pcm  = fopen('mysample.pcm', 'r');

/* PCM オーディオデータ mysample.pcm を mysong.ogg に圧縮します。*/
stream_copy_to_stream($pcm, $ogg);

fclose($pcm);
fclose($ogg);
?>
```

OpenAL 音声バインディング

導入

プラットフォームに依存しない音声バインディングです。 » OpenAL ライブラリを必要とします。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/openal.

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールでは、 Open AL(Device) - openal_device_open() が返すもの、 Open AL(Context) - openal_context_create() が返すもの、 Open AL(Buffer) - openal_buffer_create() が返すもの および Open AL(Source) - openal_source_create() が返すものの 4 種類のリソース型が定義されています。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

ALC_FREQUENCY (integer)
コンテキスト属性
ALC_REFRESH (integer)
コンテキスト属性
ALC_SYNC (integer)
コンテキスト属性
AL_FREQUENCY (integer)
バッファ設定
AL_BITS (integer)
バッファ設定
AL_CHANNELS (integer)
バッファ設定
AL_SIZE (integer)
バッファ設定
AL_BUFFER (integer)
ソース/リスナー 設定 (Integer)
AL_SOURCE_RELATIVE (integer)
ソース/リスナー 設定 (Integer)
AL_SOURCE_STATE (integer)
ソース/リスナー 設定 (Integer)
AL_PITCH (integer)
ソース/リスナー 設定 (Float)
AL_GAIN (integer)
ソース/リスナー 設定 (Float)
AL_MIN_GAIN (integer)
ソース/リスナー 設定 (Float)
AL_MAX_GAIN (integer)
ソース/リスナー 設定 (Float)
AL_MAX_DISTANCE (integer)
ソース/リスナー 設定 (Float)
AL_ROLLOFF_FACTOR (integer)
ソース/リスナー 設定 (Float)
AL_CONE_OUTER_GAIN (integer)
ソース/リスナー 設定 (Float)
AL_CONE_INNER_ANGLE (integer)
ソース/リスナー 設定 (Float)

AL_CONE_OUTER_ANGLE (integer)
ソース/リスナー 設定 (Float)
AL_REFERENCE_DISTANCE (integer)
ソース/リスナー 設定 (Float)
AL_POSITION (integer)
ソース/リスナー 設定 (Float Vector)
AL_VELOCITY (integer)
ソース/リスナー 設定 (Float Vector)
AL_DIRECTION (integer)
ソース/リスナー 設定 (Float Vector)
AL_ORIENTATION (integer)
ソース/リスナー 設定 (Float Vector)
AL_FORMAT_MONO8 (integer)
PCM フォーマット
AL_FORMAT_MONO16 (integer)
PCM フォーマット
AL_FORMAT_STEREO8 (integer)
PCM フォーマット
AL_FORMAT_STEREO16 (integer)
PCM フォーマット
AL_INITIAL (integer)
ソースの状態
AL_PLAYING (integer)
ソースの状態
AL_PAUSED (integer)
ソースの状態
AL_STOPPED (integer)
ソースの状態
AL_LOOPING (integer)
ソースの状態
AL_TRUE (integer)
OpenAL が理解する boolean 値
AL_FALSE (integer)
OpenAL が理解する boolean 値

---

openal_buffer_create

(PECL openal:0.1)

openal_buffer_create — OpenAL バッファを生成する

説明

resource **openal_buffer_create** (void)

返り値

成功した場合に Open AL (バッファ) リソース、失敗した場合に **FALSE** を返します。

参考

- openal_buffer_loadwav()
- openal_buffer_data()

---

openal_buffer_data

(PECL openal:0.1)

openal_buffer_data — バッファのデータを読み込む

説明

bool **openal_buffer_data** (resource $buffer , int $format , string $data , int $freq)

パラメータ

buffer

Open AL (バッファ) リソース (事前に openal_buffer_create() で作成したもの)。

format

data のフォーマット。 **AL_FORMAT_MONO8**、 **AL_FORMAT_MONO16**、 **AL_FORMAT_STEREO8** そして **AL_FORMAT_STEREO16** のいずれか。

data

format および freq で指定した、バイナリ音声データブロック。

freq

Hz 単位で指定した data の周波数。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_buffer_loadwav()
- openal_stream()

---

openal_buffer_destroy

(PECL openal:0.1)

openal_buffer_destroy — OpenAL バッファを削除する

説明

bool **openal_buffer_destroy** (resource $buffer)

パラメータ

buffer

Open AL (バッファ) リソース (事前に openal_buffer_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_buffer_create()

---

openal_buffer_get

(PECL openal:0.1)

openal_buffer_get — OpenAL バッファのプロパティを取得する

説明

int **openal_buffer_get** (resource $buffer , int $property)

パラメータ

buffer

Open AL (バッファ) リソース (事前に openal_buffer_create() で作成したもの)。

property

特定のプロパティ。 **AL_FREQUENCY**、 **AL_BITS**、 **AL_CHANNELS** そして **AL_SIZE** のいずれか。

返り値

要求された property の値を整数で返します。 失敗した場合には **FALSE** を返します。

参考

- openal_buffer_create()

---

openal_buffer_loadwav

(PECL openal:0.1)

openal_buffer_loadwav — .wav ファイルをバッファに読み込む

説明

bool **openal_buffer_loadwav** (resource $buffer , string $wavfile)

パラメータ

buffer

Open AL (バッファ) リソース (事前に openal_buffer_create() で作成したもの)。

wavfile

ローカルファイルシステム上の .WAV ファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_buffer_data()
- openal_stream()

---

openal_context_create

(PECL openal:0.1)

openal_context_create — 音声処理コンテキストを作成する

説明

resource **openal_context_create** (resource $device)

パラメータ

device

Open AL (デバイス) リソース (事前に openal_device_open() で作成したもの)。

返り値

成功した場合に Open AL (コンテキスト) リソース、 失敗した場合に **FALSE** を返します。

参考

- openal_device_open()
- openal_context_destroy()

---

openal_context_current

(PECL openal:0.1)

openal_context_current — 指定したコンテキストを現在のコンテキストにする

説明

bool **openal_context_current** (resource $context)

パラメータ

context

Open AL (コンテキスト) リソース (事前に openal_context_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_context_create()

---

openal_context_destroy

(PECL openal:0.1)

openal_context_destroy — コンテキストを削除する

説明

bool **openal_context_destroy** (resource $context)

パラメータ

context

Open AL (コンテキスト) リソース (事前に openal_context_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_context_create()

---

openal_context_process

(PECL openal:0.1)

openal_context_process — 指定したコンテキストを処理する

説明

bool **openal_context_process** (resource $context)

パラメータ

context

Open AL (コンテキスト) リソース (事前に openal_context_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_context_create()
- openal_context_current()
- openal_context_suspend()

---

openal_context_suspend

(PECL openal:0.1)

openal_context_suspend — 指定したコンテキストをサスペンドする

説明

bool **openal_context_suspend** (resource $context)

パラメータ

context

Open AL (コンテキスト) リソース (事前に openal_context_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_context_create()
- openal_context_current()
- openal_context_process()

---

openal_device_close

(PECL openal:0.1)

openal_device_close — OpenAL デバイスを閉じる

説明

bool **openal_device_close** (resource $device)

パラメータ

device

閉じようとしている Open AL (デバイス) リソース (事前に openal_device_open() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_device_open()

---

openal_device_open

(PECL openal:0.1)

openal_device_open — OpenAL 音声レイヤを初期化する

説明

resource **openal_device_open** ([string $device_desc])

パラメータ

device_desc

device_desc で指定した音声デバイスをオープンします。 device_desc が指定されなかった場合は、 最初に見つかった音声デバイスが使用されます。

返り値

成功した場合に Open AL (デバイス) リソース、 失敗した場合に **FALSE** を返します。

参考

- openal_device_close()
- openal_context_create()

---

openal_listener_get

(PECL openal:0.1)

openal_listener_get — リスナーのプロパティを取得する

説明

mixed **openal_listener_get** (int $property)

パラメータ

property

取得するプロパティ。 **AL_GAIN** (float)、 **AL_POSITION** (array(float,float,float))、 **AL_VELOCITY** (array(float,float,float)) そして **AL_ORIENTATION** (array(float,float,float)) のいずれか。

返り値

float あるいは float の配列 (どちらか適切なほう)、 あるいは失敗した場合に, **FALSE** を返します。

参考

- openal_listener_set()

---

openal_listener_set

(PECL openal:0.1)

openal_listener_set — リスナーのプロパティを設定する

説明

bool **openal_listener_set** (int $property , mixed $setting)

パラメータ

property

設定するプロパティ。 **AL_GAIN** (float)、 **AL_POSITION** (array(float,float,float))、 **AL_VELOCITY** (array(float,float,float)) そして **AL_ORIENTATION** (array(float,float,float)) のいずれか。

setting

設定する値。float あるいは float の配列のうち適切なもの。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_listener_get()

---

openal_source_create

(PECL openal:0.1)

openal_source_create — ソースリソースを生成する

説明

resource **openal_source_create** (void)

返り値

成功した場合に Open AL (ソース) リソース、 失敗した場合に **FALSE** を返します。

参考

- openal_source_set()
- openal_source_play()
- openal_source_destroy()

---

openal_source_destroy

(PECL openal:0.1)

openal_source_destroy — ソースリソースを削除する

説明

bool **openal_source_destroy** (resource $source)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_source_create()

---

openal_source_get

(PECL openal:0.1)

openal_source_get — OpenAL ソースのプロパティを取得する

説明

mixed **openal_source_get** (resource $source , int $property)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

property

取得するプロパティ。 **AL_SOURCE_RELATIVE** (int)、 **AL_SOURCE_STATE** (int)、 **AL_PITCH** (float)、 **AL_GAIN** (float)、 **AL_MIN_GAIN** (float)、 **AL_MAX_GAIN** (float)、 **AL_MAX_DISTANCE** (float)、 **AL_ROLLOFF_FACTOR** (float)、 **AL_CONE_OUTER_GAIN** (float)、 **AL_CONE_INNER_ANGLE** (float)、 **AL_CONE_OUTER_ANGLE** (float)、 **AL_REFERENCE_DISTANCE** (float)、 **AL_POSITION** (array(float, float,float))、 **AL_VELOCITY** (array(float,float,float))、 **AL_DIRECTION** (array(float,float,float)) のいずれか。

返り値

取得したプロパティに関連付けられた型、あるいは失敗した場合に **FALSE** を返します。

参考

- openal_source_create()
- openal_source_set()
- openal_source_play()

---

openal_source_pause

(PECL openal:0.1)

openal_source_pause — ソースを一時停止する

説明

bool **openal_source_pause** (resource $source)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_source_stop()
- openal_source_play()
- openal_source_rewind()

---

openal_source_play

(PECL openal:0.1)

openal_source_play — ソースの再生を開始する

説明

bool **openal_source_play** (resource $source)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_source_stop()
- openal_source_pause()
- openal_source_rewind()

---

openal_source_rewind

(PECL openal:0.1)

openal_source_rewind — ソースを巻き戻す

説明

bool **openal_source_rewind** (resource $source)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_source_stop()
- openal_source_pause()
- openal_source_play()

---

openal_source_set

(PECL openal:0.1)

openal_source_set — ソースのプロパティを設定する

説明

bool **openal_source_set** (resource $source , int $property , mixed $setting)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

property

設定するプロパティ。 **AL_BUFFER** (OpenAL (ソース))、 **AL_LOOPING** (bool)、 **AL_SOURCE_RELATIVE** (int)、 **AL_SOURCE_STATE** (int)、 **AL_PITCH** (float)、 **AL_GAIN** (float)、 **AL_MIN_GAIN** (float)、 **AL_MAX_GAIN** (float)、 **AL_MAX_DISTANCE** (float)、 **AL_ROLLOFF_FACTOR** (float)、 **AL_CONE_OUTER_GAIN** (float)、 **AL_CONE_INNER_ANGLE** (float)、 **AL_CONE_OUTER_ANGLE** (float)、 **AL_REFERENCE_DISTANCE** (float)、 **AL_POSITION** (array(float,float,float))、 **AL_VELOCITY** (array(float,float,float))、 **AL_DIRECTION** (array(float,float,float)) のいずれか。

setting

指定した property に代入する値。 代入できる値については、property の説明を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_source_create()
- openal_source_get()
- openal_source_play()

---

openal_source_stop

(PECL openal:0.1)

openal_source_stop — ソースの再生を停止する

説明

bool **openal_source_stop** (resource $source)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openal_source_play()
- openal_source_pause()
- openal_source_rewind()

---

openal_stream

(PECL openal:0.1)

openal_stream — ソースのストリーム再生を開始する

説明

resource **openal_stream** (resource $source , int $format , int $rate)

パラメータ

source

Open AL (ソース) リソース (事前に openal_source_create() で作成したもの)。

format

data のフォーマット。 **AL_FORMAT_MONO8**、 **AL_FORMAT_MONO16**、 **AL_FORMAT_STEREO8** そして **AL_FORMAT_STEREO16** のいずれか。

rate

データをストリームに流す際の周波数を Hz 単位で指定します。

返り値

成功した場合にストリームリソース、失敗した場合に **FALSE** を返します。

参考

- openal_source_create()
- fwrite()

目次

OpenSSL 関数

導入

このモジュールは、署名の生成および認証、そして、データのシール (暗号化)およびオープン(復号化)を行うために、 » OpenSSL の関数を使用します。 OpenSSL は多くの機能を提供しますが、これらはまだこのモジュールでは サポートされていません。これらのいくつかは将来的に追加される可能性が あります。

要件

OpenSSL 関数を使用するためには、» OpenSSL パッケージがインストール されていることを要します。 PHP のバージョン 4.0.5 から 4.3.1 までは、OpenSSL >= 0.9.5 で動作します。他のバージョン (PHP <=4.0.4 および >= 4.3.2) では OpenSSL >= 0.9.6 を必要とします。

警告

最新のバージョンの OpenSSL を使用するようにしてください。 さもないと、Web サーバへの攻撃に対しての脆弱性をかかえてしまいます。

インストール手順

PHP の OpenSSL サポートを使用するには、--with-openssl[=DIR] を指定して PHP を コンパイルする必要があります。

> **注意**: **Win32 ユーザへの注意** この拡張モジュールを動作させるには、 Windows システムの PATH が通った場所に DLL ファイルが存在する必要があります。 FAQ の "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" で、その方法

を説明しています。 DLL ファイルを PHP のフォルダから Windows のシステムディレクトリにコピーしても動作します（システムディレクトリは、デフォルトで PATH に含まれるからです）が、これは推奨しません。 この拡張モジュールを使用するには、以下のファイルが PATH の通った場所にある必要があります。 libeay32.dll
加えてキー生成およびサイン認証関数を使用する計画がある場合、 システムに 有効な openssl.cnf をインストールする 必要があります。PHP 4.3.0 以降、Win32 バイナリ配布版の openssl フォルダに サンプル設定ファイルが含まれています。 PHP 4.2.0 以降を使用しておりこのファイルがない場合、 » OpenSSLのホームページから入手するか PHP 4.3.0 のリリース版をダウンロードし、それらに含まれる 設定ファイルを使用することができます。 PHP は、 以下のロジックにより openssl.cnf を探します。

- 環境変数 OPENSSL_CONF が設定された場合、 設定ファイルの（ファイル名を含む）パスとして使用されます。
- 環境変数 SSLEAY_CONF が設定された場合、 設定ファイルの（ファイル名を含む）パスとして使用されます。
- ファイル openssl.cnf はデフォルトの認証エリアに あることが仮定され、openssl DLL がコンパイルされた時間で設定されます。 通常、デフォルトのファイル名が c:¥usr¥local¥ssl¥openssl.cnf であることを 意味します。

インストール時に、設定ファイルを c:¥usr¥local¥ssl¥openssl.cnf または 他の場所にインストールし、(例えば仮想ホスト毎に)環境変数に設定ファ イルの場所を指定するかを選ぶ必要があります。 設定ファイルを必要とする関数の configargs に より、デフォルトのパスを上書きすることが可能であることに注意してください。

実行時設定

設定ディレクティブは定義されていません。

リソース型

キー/証明書パラメータ

OpenSSL 関数のうち、キーまたは証明書パラメータを必要とするものは ごく一部です。PHP 4.0.5 より以前では、openssl_get_xxx 関数により返されたキーまたは証明書リソースを使用する必要がありました。 これより後のバージョンでは、次の方法のどれかを使用することが 可能となる予定です。

- 証明書
  1. openssl_x509_read() から返された X.509 リソース。
  2. file://path/to/cert.pem 形式の文字列。 このファイルは、PEM エンコードされた証明書である必要があります。
  3. PEMエンコードされた証明書の内容を含む文字列。
- 公開鍵/秘密鍵
  1. openssl_get_publickey() あるいは openssl_get_privatekey() から返された キーリソース。
  2. 公開鍵のみ: X.509リソース。
  3. file://path/to/file.pem 形式の文字列。- このファイルは、PEM エンコードされた証明書/秘密鍵である 必要があります（両方を含むことも可能です）。
  4. PEM エンコードされた証明書/キーの内容を含む文字列
  5. 秘密鍵については array($key, $passphrase) という構文を使用することも可能です。 ただし、$key は file:// または上記のテキスト表現形式を使用 して指定したキー、$passphrase はその秘密鍵に関するパスワードを 有する文字列を表します。

証明書の認証

サイン/証明書を認証する関数をコールする際、 cainfo パラメータは、ファイルと認証済みの CA ファイルの場所を指定するファイルディレクトリ名を含む配列です。 ディレクトリが指定された場合、**openssl** コマンドが 使用できるような正しい形式にハッシュされたディレクトリである必要があります。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

目的を調べるフラグ

X509_PURPOSE_SSL_CLIENT (integer)
X509_PURPOSE_SSL_SERVER (integer)
X509_PURPOSE_NS_SSL_SERVER (integer)
X509_PURPOSE_SMIME_SIGN (integer)
X509_PURPOSE_SMIME_ENCRYPT (integer)
X509_PURPOSE_CRL_SIGN (integer)
X509_PURPOSE_ANY (integer)

パディングフラグ

OPENSSL_PKCS1_PADDING (integer)
OPENSSL_SSLV23_PADDING (integer)
OPENSSL_NO_PADDING (integer)
OPENSSL_PKCS1_OAEP_PADDING (integer)

キーの型

OPENSSL_KEYTYPE_RSA (integer)
OPENSSL_KEYTYPE_DSA (integer)
OPENSSL_KEYTYPE_DH (integer)

PKCS7 フラグ/定数

S/MIME 関数はビットフィールドを使用して指定したフラグを使用します。 このビットフィールドには、以下の値を一つ以上含むことが可能です。

PKCS7 定数

定数	説明
PKCS7_TEXT	text/plain content type ヘッダを暗号化/署名されたメッセージに 追加します。復号化または認証を行う際には、このヘッダは出力から 取り除かれます。復号化または認証されたメッセージがMIME 型 text/plain でない場合、エラーとなります。
PKCS7_BINARY	通常は、入力されたメッセージは CR および LF を行端として使用した 「正規化」された形式に変換されます。こらは、S/MIME の規格に 基づくものです。このオプションが指定された場合、変換は行われません。 この機能は、MIME 形式でないバイナリデータを処理する際に 便利です。
PKCS7_NOINTERN	メッセージを認証する際に、通常、メッセージに含まれる証明書が 証明書にサインする際に検索されます。 このオプションでは、 openssl_pkcs7_verify() の *extracerts* パラメータで指定した証明書 のみが使用されます。しかし、指定された証明書を信頼されていな い CA として使用することも可能です。
PKCS7_NOVERIFY	サインつきメッセージをサインした証明書の署名について 検証しません。
PKCS7_NOCHAIN	サインを行った側の証明書の認証の連鎖を行いません。 この場合、サイン付きのメッセージにある証明書を未認証の CA として使用しません。
PKCS7_NOCERTS	メッセージにサインする際、通常はサインをする人の証明書が挿入 されますが、このオプションを指定した場合はそうなりません。これに よりサイン付きのメッセージのサイズは小さくなりますが、認証 側が（例えば、openssl_pkcs7_verify() の *extracerts* を用いて渡すことにより） サインをした人の証明書のコピーをローカルに用意する必要があります。
PKCS7_NOATTR	通常、メッセージがサインされる時、サインした時間やサポートされる 対象アルゴリズムを含む一連の属性が付加されます。このオプションを 指定した場合、それらの属性は付加されません。
PKCS7_DETACHED	メッセージにサインをする際、MIME型 multipart/signed を指定して クリアテキストでサインを行います。これは、openssl_pkcs7_sign() において フラグを指定しなかった場合の *flags* パラメータのデフォルトです。このオプションをオフにした場合、 メッセージは不透明なサインによりサインされます。これは、 メールリレイによる変換に対してより耐性がありますが、S/MIME を サポートしないメールエージェントでは読むことはできません。
PKCS7_NOSIGS	メッセージにサインや認証を試みません。

***注意**: これらの定数は、4.0.6 で追加されました。*

署名アルゴリズム

OPENSSL_ALGO_SHA1 *(integer)*
openssl_sign() および openssl_verify() のデフォルトアルゴリズムとして用いられます。
OPENSSL_ALGO_MD5 *(integer)*
OPENSSL_ALGO_MD4 *(integer)*
OPENSSL_ALGO_MD2 *(integer)*

***注意**: これらの定数は、5.0.0 で追加されました。*

暗号化方式

OPENSSL_CIPHER_RC2_40 *(integer)*
OPENSSL_CIPHER_RC2_128 *(integer)*
OPENSSL_CIPHER_RC2_64 *(integer)*
OPENSSL_CIPHER_DES *(integer)*
OPENSSL_CIPHER_3DES *(integer)*

***注意**: これらの定数は、4.3.0 で追加されました。*

バージョン定数

OPENSSL_VERSION_TEXT *(string)*
OPENSSL_VERSION_NUMBER *(integer)*

***注意**: これらの定数は、5.2.0 で追加されました。*

---

openssl_csr_export_to_file

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_export_to_file — CSR をファイルにエクスポートする

説明

bool ***openssl_csr_export_to_file*** *(resource $csr , string $outfilename [, bool $notext])*

openssl_csr_export_to_file() *は、Certificate Signing Request csr を受け取り、 それを outfilename という名前のファイルに ASCII テキストとして保存します。*

パラメータ

csr

outfilename

出力ファイルへのパス。

notext

*オプションのパラメータ notext は、 出力内容の冗長性に影響します。**FALSE** を設定すると、人間が理解できる形式の 追加情報が出力に付加されます。notext の デフォルト値は **TRUE** です。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *openssl_csr_export()*
- *openssl_csr_new()*
- *openssl_csr_sign()*

---

openssl_csr_export

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_export — CSR を文字列としてエクスポートする

説明

*bool **openssl_csr_export** (resource $csr , string &$out [, bool $notext])*

***openssl_csr_export()** は、Certificate Signing Request csr を受け取り、 それを out に ASCII テキストとして保存します。 このパラメータは参照で渡されます。*

パラメータ

csr

out

notext

*オプションのパラメータ notext は、 出力内容の冗長性に影響します。**FALSE** を設定すると、人間が理解できる形式の 追加情報が出力に付加されます。notext の デフォルト値は **TRUE** です。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *openssl_csr_export_to_file()*
- *openssl_csr_new()*
- *openssl_csr_sign()*

---

openssl_csr_get_public_key

(PHP 5 >= 5.2.0)

openssl_csr_get_public_key — CERT の公開鍵を返す

説明

*resource **openssl_csr_get_public_key** (mixed $csr [, bool $use_shortnames])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

openssl_csr_get_subject

(PHP 5 >= 5.2.0)

openssl_csr_get_subject — CERT の subject を返す

説明

*array **openssl_csr_get_subject** (mixed $csr [, bool $use_shortnames])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

openssl_csr_new

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_new — CSR を作成する

説明

mixed **openssl_csr_new** (array $dn , resource &$privkey [, array $configargs [, array $extraattribs]])

openssl_csr_new() は、新しい CSR (Certificate Signing Request) を dn の情報に基づいて作成します。dn は、証明書で使用される識別名です。

> **注意**: この関数を正しく機能させるには、有効な openssl.cnf がインストールされている必要があります。詳細な情報については インストールセクション の下にある注記を参照ください。

パラメータ

dn

証明書で使用される識別名。

privkey

privkey には、事前に openssl_pkey_new() （あるいはその他の openssl_pkey 系の関数）で作成した秘密鍵を設定します。 これに対応する公開鍵が、CSR への署名に使用されます。

configargs

デフォルトでは、システムの openssl.conf の設定に したがってリクエストが初期化されます。configargs のキー config_section_section を設定することで、この デフォルト項目を変更することが可能です。また、キー config に別の openssl 設定ファイルを指定することで別の設定を使用することも可能です。 もし configargs に以下の表のキーが存在すれば、それらは openssl.conf の対応する項目と同じ働きをします。

設定の上書き

configargs のキー	型	openssl.conf で同等の意味を持つ項目	説明
digest_alg	string	default_md	使用するダイジェストメソッドを選択します
x509_extensions	string	x509_extensions	x509 証明書を作成する際に使用する拡張モジュールを選択します
req_extensions	string	req_extensions	CSR を作成する際に使用する拡張モジュールを選択します
private_key_bits	integer	default_bits	秘密鍵を作成する際に使用するビット数を指定します
private_key_type	integer	none	作成する秘密鍵の型を指定します。以下の定数 OPENSSL_KEYTYPE_DSA、OPENSSL_KEYTYPE_DH あるいは OPENSSL_KEYTYPE_RSA からひとつ選択します。 デフォルト値は OPENSSL_KEYTYPE_RSA で、 現時点でサポートしているのはこの型のみです。
encrypt_key	boolean	encrypt_key	（パスフレーズとともに）エクスポートされるキーを暗号化するか?

extraattribs

extraattribs は、CSR に関する追加の設定情報を 設定するために使用します。dn および extraattribs はどちらも連想配列で、それらの キーが OID に変換されたうえでリクエストの関連する部分に適用されます。

返り値

CSR を返します。

例

Example#1 自己署名証明書の作成

```php
<?php
// 証明書で使用する識別名を指定します。
// ご自分の環境に応じて、名前や会社名を変更する必要があります。
// より厳密に言うなら、あなたが証明書を作成しようとしている個人あるいは
// 組織の名前と会社名です。
// SSL 証明書では、通常は証明書を使用するドメイン名を commonName に
// 指定します。しかし S/MIME 証明書では、証明書を使用する個人の名前を
// commonName に指定します。
$dn = array(
    "countryName" => "UK",
    "stateOrProvinceName" => "Somerset",
    "localityName" => "Glastonbury",
    "organizationName" => "The Brain Room Limited",
    "organizationalUnitName" => "PHP Documentation Team",
    "commonName" => "Wez Furlong",
    "emailAddress" => "wez@example.com"
);

// 新しい秘密鍵（および公開鍵）のペアを作成します
$privkey = openssl_pkey_new();

// 証明書への署名要求を作成します
$csr = openssl_csr_new($dn, $privkey);

// 認証局があなたの申請を受け付けてくれるまでは、自己署名の証明書を
// 作成したくなるでしょう。
// これは、365 日間有効な自己署名の証明書を作成します。
$sscert = openssl_csr_sign($csr, null, $privkey, 365);

// ここで、Web サーバやメールサーバ、メールクライアント（証明書の使用方法に
// 依存します）にインストールするために、
// 作成した秘密鍵・CSR および自己署名の証明書を保存したくなることでしょう。
// この例では、これらを変数に格納する方法を示します。とはいえ、もちろん
// それを直接ファイルに保存することも可能です。
// 一般的には、CSR を認証局に送り、「正式な」証明書として証明してもらう
// ことになります。
openssl_csr_export($csr, $csrout) and var_dump($csrout);
openssl_x509_export($sscert, $certout) and var_dump($certout);
```

```
openssl_pkey_export($privkey, $pkeyout, "mypassword") and var_dump($pkeyout);

// 発生したエラーを表示します。
while (($e = openssl_error_string()) !== false) {
    echo $e . "¥n";
}
?>
```

---

openssl_csr_sign

(PHP 4 >= 4.2.0, PHP 5)

openssl_csr_sign — 他の CERT（あるいは自分自身）で証明書をサインする

説明

resource **openssl_csr_sign** (mixed $csr , mixed $cacert , mixed $priv_key , int $days [, array $configargs [, int $serial]])

openssl_csr_sign() は、 指定した CSR を用いて x509 証明書リソースを作成します。

> **注意**: この関数を正しく機能させるには、有効な openssl.cnf がインストールされている必要があります。詳細な情報については インストールセクション の下にある注記を参照ください。

パラメータ

csr

openssl_csr_new() で作成した CSR。 file://path/to/csr 、あるいは openssl_csr_export() で生成した文字列で指定した場合は PEM エンコードされた CSR も使用可能です。

cacert

作成された証明書は cacert で署名されます。 cacert が **NULL** の場合は、自己署名の証明書となります。

priv_key

priv_key は cacert に対応する秘密鍵です。

days

days は、作成された証明書の有効期限を日数で指定します。

configargs

configargs で証明書の詳細設定が可能です。 configargs についての詳細な情報は openssl_csr_new() を参照ください。

serial

発行される証明書のシリアル番号を、オプションで指定します。 省略した場合のデフォルトは 0 です。

返り値

成功した場合に x509 証明書リソース、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.3	serial パラメータが追加されました。

例

Example#1 openssl_csr_sign() の例 - CSR に署名する（あなた自身の CA を作成する）

```
<?php
// このスクリプトでは、前のページのテキストエリアから受け取った
// CSR を利用すると仮定します。
$csrdata = $_POST["CSR"];

// これから、私たち自身 "certificate authority" 証明書を使用して
// 署名を行います。どんな証明書でも署名は可能ですが、署名された
// 証明書がソフトウェアやユーザに信頼されない限り、その手続きは
// 無意味です。

// CA cert および秘密鍵が必要です。
$cacert = "file://path/to/ca.crt";
$privkey = array("file://path/to/ca.key", "your_ca_key_passphrase");

$userscert = openssl_csr_sign($csrdata, $cacert, $privkey, 365);

// 作成された証明書を表示します。これをコピーして、
// ローカル設定（たとえば SSL サーバへの接続設定ファイルなど）
// に貼り付けます。
openssl_x509_export($usercert, $certout);
echo $certout;

// 発生したエラーをすべて表示します。
while (($e = openssl_error_string()) !== false) {
    echo $e . "¥n";
}
?>
```

openssl_error_string

(PHP 4 >= 4.0.6, PHP 5)

openssl_error_string — OpenSSL エラーメッセージを返す

説明

string **openssl_error_string** (void)

openssl_error_string() は、 OpenSSL ライブラリから直近のエラーを返します。 エラーメッセージはスタックにつまれており、 全ての情報を集めるにはこの関数を複数回コールする必要があります。

返り値

エラーメッセージ文字列を返します。 エラーメッセージがもうない場合は、**FALSE** を返します。

例

Example#1 openssl_error_string() の例

```
<?php
// ここで、処理に失敗する OpenSSL 関数をコールしたと仮定します
while ($msg = openssl_error_string())
    echo $msg . "<br />¥n";
?>
```

---

openssl_free_key

(PHP 4 >= 4.0.4, PHP 5)

openssl_free_key — キーリソースを開放する

説明

void **openssl_free_key** (resource $key_identifier)

openssl_free_key() は、指定した key_identifier が指すキーをメモリから開放します。

パラメータ

key_identifier

返り値

値を返しません。

---

openssl_get_privatekey

(PHP 4 >= 4.0.4, PHP 5)

openssl_get_privatekey — openssl_pkey_get_private() のエイリアス

説明

この関数は次の関数のエイリアスです。 openssl_pkey_get_private().

---

openssl_get_publickey

(PHP 4 >= 4.0.4, PHP 5)

openssl_get_publickey — openssl_pkey_get_public() のエイリアス

説明

この関数は次の関数のエイリアスです。 openssl_pkey_get_public().

---

openssl_open

(PHP 4 >= 4.0.4, PHP 5)

openssl_open — シール(暗号化)されたデータをオープン(復号)する

説明

bool **openssl_open** (string $sealed_data , string &$open_data , string $env_key , mixed $priv_key_id)

openssl_open() は、キー ID priv_key_id およびエンベロープキー env_key に関連する公開鍵を使用して、 sealed_data をオープン(復号)します。 エンベロープキーは、データがシール(暗号化)された際に生成され、特定の 一つの公開鍵でのみ使用することが可能です。詳細な情報については、 openssl_seal() を参照ください。

パラメータ

sealed_data

open_data

成功した場合、オープンしたデータをここに返します。

env_key

priv_key_id

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 openssl_open() の例

```
<?php
// $sealed および $env_key に暗号化されたデータおよびエンベロープキー
// が含まれていると仮定。共にシール元(暗号化側)から与えられる。

// ファイルから公開鍵を取得し、使用可能とする
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// データを復号化し、$open に保存
if (openssl_open($sealed, $open, $env_key, $pkeyid)) {
    echo "here is the opened data: ", $open;
} else {
    echo "failed to open data";
}

// 公開鍵をメモリから開放
openssl_free_key($pkeyid);
?>
```

参考

- openssl_seal()

---

openssl_pkcs12_export_to_file

(PHP 5 >= 5.2.2)

openssl_pkcs12_export_to_file — PKCS#12 互換の証明書保存ファイルをエクスポートする

説明

bool **openssl_pkcs12_export_to_file** (mixed $x509 , string $filename , mixed $priv_key , string $pass [, array $args])

openssl_pkcs12_export_to_file() は、 x509 をファイル filename に PKCS#12 フォーマットで保存します。

パラメータ

x509

filename

出力ファイルへのパス。

priv_key

PKCS#12 ファイルのプライベートキーコンポーネント。

pass

PKCS#12 ファイルのロックを解除するためのパスワード。

args

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

openssl_pkcs12_export

(PHP 5 >= 5.2.2)

openssl_pkcs12_export — PKCS#12 互換の証明書保存ファイルを変数にエクスポートする

説明

bool **openssl_pkcs12_export** (mixed $x509 , string &$out , mixed $priv_key , string $pass [, array $args])

openssl_pkcs12_export() は、 x509 を PKCS#12 ファイルフォーマットにしたものを out で指定した文字列に格納します。

パラメータ

x509

out

成功した場合に、ここに PKCS#12 が保存されます。

priv_key

PKCS#12 ファイルのプライベートキーコンポーネント。

pass

PKCS#12 ファイルのロックを解除するためのパスワード。

args

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

openssl_pkcs12_read

(PHP 5 >= 5.2.2)

openssl_pkcs12_read — PKCS#12 認証ストアをパースして配列形式にする

説明

bool **openssl_pkcs12_read** (mixed $PKCS12 , array &$certs , string $pass)

openssl_pkcs12_read() は、 PKCS12 で指定した PKCS#12 認証ストアをパースして certs で指定した配列に格納します。

パラメータ

PKCS12

certs

成功した場合に、ここに認証ストアデータが格納されます。

pass

PKCS#12 ファイルのロックを解除するためのパスワード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

openssl_pkcs7_decrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_decrypt — S/MIME 暗号化されたメッセージを復号する

説明

bool **openssl_pkcs7_decrypt** (string $infilename , string $outfilename , mixed $recipcert [, mixed $recipkey])

infilename で指定したファイル中の S/MIME 暗号化されたメッセージを、recipcert および recipkey で指定した証明書と公開鍵を用いて復号します。

パラメータ

infilename

outfilename

復号したメッセージは、outfilename で指定したファイルに出力されます。

recipcert

recipkey

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 openssl_pkcs7_decrypt() の例

```
<?php
// $cert および $key にはあなたの個人証明書と公開鍵が含まれており、
// あなたはS/MIMEメッセージの受信者であると仮定します。
$infilename = "encrypted.msg";  //  暗号化されたメッセージを含むファイル
$outfilename = "decrypted.msg"; //  このファイルへの書き込み権限が必要

if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key)) {
    echo "復号しました!";
} else {
    echo "復号に失敗しました!";
}
?>
```

---

openssl_pkcs7_encrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_encrypt — S/MIME メッセージを暗号化する

説明

bool **openssl_pkcs7_encrypt** (string $infile , string $outfile , mixed $recipcerts , array $headers [, int $flags [, int $cipherid]])

openssl_pkcs7_encrypt() は、 infile という名前のファイルの内容を RC2 40 ビット暗号により暗号化します。この内容は、 recipcerts で指定した意図する受信者によってのみ読むことが可能です。

パラメータ

infile

outfile

recipcerts

X.509 証明書または X.509 証明書の配列。

headers

headers は、 暗号化された後にデータの前に付加されるヘッダの配列です。

headers はヘッダ名をキーとする連想配列または添字配列であり、 各要素には、各ヘッダ行が一行ずつ含まれています。

flags

flags は オプションとして使用可能であり、エンコード処理を変更するために指定します。 PKCS7 定数 を参照ください。

cipherid

cipherid で暗号を選択可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	cipherid パラメータが追加されました。

例

Example#1 openssl_pkcs7_encrypt() の例

```
<?php
// 暗号化するメッセージを nighthawk という名前の外部の秘密の
// エージェントに送信します。送信先の証明書をファイル nighthawk.pem に
// 有しています。
$data = <<<EOD
Nighthawk,

Top secret, for your eyes only!

The enemy is closing in! Meet me at the cafe at 8.30am
to collect your forged passport!

HQ
EOD;

// キーを読み込む
$key = file_get_contents("nighthawk.pem");

// ファイルにメッセージを保存
$fp = fopen("msg.txt", "w");
```

```
fwrite($fp, $data);
fclose($fp);

// メッセージを暗号化
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", $key,
     array("To" => "nighthawk@example.com", //  連想配列の構文
           "From:  HQ  <hq@example.com>", //  添字配列の構文
           "Subject" => "Eyes  only"))) {
    // メッセージを暗号化し、送信します!
    exec(ini_get("sendmail_path") . " < enc.txt");
}
?>
```

---

openssl_pkcs7_sign

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_sign — S/MIME メッセージにサインする

説明

bool **openssl_pkcs7_sign** (string $infilename , string $outfilename , mixed $signcert , mixed $privkey , array $headers [, int $flags [, string $extracerts]])

openssl_pkcs7_sign() は、 infilename という名前のファイルの内容について パラメータ signcert および privkey で指定した証明書と公開鍵を用いてサインをします。

パラメータ

infilename

outfilename

signcert

privkey

headers

headers は、ヘッダの配列です。このヘッダは、 サインされた後でデータの前に付加されます (このパラメータの形式の詳細については openssl_pkcs7_encrypt() を参照ください)。

flags

flags を出力を変更するために使用することが可能です。 PKCS7 定数 を参照ください。これが指定されない場合、デフォルトの PKCS7_DETACHED になります。

extracerts

extracerts には、 サインに含めるための他の一連の証明書を記述したファイル名を指定します。 これは、例えば送信者が使用した証明書を受信者が検証しやすくするために使用することが可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 openssl_pkcs7_sign() の例

```
<?php
// 受信者が送信者を確認できるようにサインしたいメッセージ
$data = <<<EOD

You have my authorization to spend $10,000 on dinner expenses.

The CEO
EOD;
// ファイルにメッセージを保存
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// 暗号化
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
     array("file://mycert.pem", "mypassphrase"),
     array("To" => "joes@example.com", //  連想配列の構文
           "From:  HQ  <ceo@example.com>", //  添字配列の構文
           "Subject" => "Eyes  only")
     )) {
    // メッセージはサインされました。送信しましょう!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
?>
```

---

openssl_pkcs7_verify

(PHP 4 >= 4.0.6, PHP 5)

openssl_pkcs7_verify — S/MIME でサインされたメッセージの署名を検証する

説明

mixed ***openssl_pkcs7_verify*** *(string $filename , int $flags [, string $outfilename [, array $cainfo [, string $extracerts [, string $content]]]])*

openssl_pkcs7_verify() *は、 指定したファイルの S/MIME メッセージを読み込み、デジタル署名を評価します。*

パラメータ

filename

メッセージへのパス。

flags

flags により署名の検証方法を指定することが可能です。 詳細については、PKCS7 定数 を参照ください。

outfilename

outfilename を指定する場合、 メッセージに署名した人の証明書が PEM 形式で保存されたファイルの名前をこの変数に指定する必要があります。

cainfo

cainfo が指定された場合、 検証処理で使用するために認証済みの CA 証明書に関する情報を保持する必要があります。 このパラメータに関するより詳細な情報については、 証明書の検証 を参照ください。

extracerts

extracerts が指定された場合、 これは未認証の CA として一連の証明書を使用するためのファイルの名前となります。

content

ファイル名とともに content を指定すると、検証したデータがここに格納されます。 格納する際に、署名情報は除去されます。

返り値

署名が検証された場合は ***TRUE****、正しくない場合（メッセージが改暫されたか署名に用いられた証明書が無効）は* ***FALSE****、 エラーの場合に -1 を返します。*

変更履歴

バージョン	説明
5.1.0	*content* パラメータが追加されました。

---

openssl_pkey_export_to_file

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_export_to_file — エクスポート可能な形式で、キーをファイルに取得する

説明

bool ***openssl_pkey_export_to_file*** *(mixed $key , string $outfilename [, string $passphrase [, array $configargs]])*

openssl_pkey_export_to_file() *は、ASCII 変換された (PEM エンコードされた) key を outfilename のファイルに保存します。*

> ***注意****: この関数を正しく機能させるには、有効な openssl.cnf がインストールされている必要があります。詳細な情報については インストールセクション の下にある注記を参照ください。*

パラメータ

key

outfilename

出力ファイルへのパス。

passphrase

オプションで passphrase を使用してキーを保護することが可能です。

configargs

configargs により openssl 設定ファイルの設定を 追加したり上書きしたりすることで、エクスポート処理の詳細設定が可能です。 configargs についての詳細な情報は openssl_csr_new() を参照ください。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

---

openssl_pkey_export

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_export — エクスポート可能な形式で、キーを文字列に取得する

説明

bool **openssl_pkey_export** (mixed $key , string &$out [, string $passphrase [, array $configargs]])

openssl_pkey_export() は、 key を PEM エンコードした文字列として取得し、 out （参照渡し）に格納します。

> **注意**: この関数を正しく機能させるには、有効な openssl.cnf がインストールされている必要があります。詳細な情報については インストールセクション の下にある注記を参照ください。

パラメータ

key

out

passphrase

オプションで passphrase を使用してキーを保護することが可能です。

configargs

configargs により openssl 設定ファイルの設定を 追加したり上書きしたりすることで、エクスポート処理の詳細設定が可能です。 configargs についての詳細な情報は openssl_csr_new() を参照ください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

openssl_pkey_free

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_free — 秘密鍵を開放する

説明

void **openssl_pkey_free** (resource $key)

この関数は、openssl_pkey_new() で作成した秘密鍵を開放します。

パラメータ

key

キーを保持するリソース。

返り値

値を返しません。

---

openssl_pkey_get_details

(PHP 5 >= 5.2.0)

openssl_pkey_get_details — キーの詳細の配列を返す

説明

array **openssl_pkey_get_details** (resource $key)

この関数は、キーの詳細 (bits, key, type) を返します。

パラメータ

key

キーを保持する配列。

返り値

成功した場合にキーの詳細を含む配列、失敗した場合に **FALSE** を返します。 返される配列のキーは bits (ビット数)、 key (公開鍵を表す文字列) および type (キーの種類。 **OPENSSL_KEYTYPE_RSA**、 **OPENSSL_KEYTYPE_DSA**、 **OPENSSL_KEYTYPE_DH**、 **OPENSSL_KEYTYPE_EC** のいずれか。あるいは不明な場合は -1) となります。

---

openssl_pkey_get_private

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_get_private — 秘密鍵を取得する

説明

resource **openssl_pkey_get_private** (mixed $key [, string $passphrase])

openssl_get_privatekey() は key をパースし、他の関数で使用できるよう準備します。

パラメータ

key

key は以下のいずれかです。

1. file://path/to/file.pem 形式の文字列。 このファイルは、PEM エンコードされた証明書/秘密鍵である必要が あります（両方を含むことも可能です）。
2. PEM フォーマットの秘密鍵。

passphrase

指定されたキーが（パスフレーズを用いて）暗号化されている場合は、 オプションのパラメータ passphrase を使用する必要があります。

返り値

成功した場合に正のキーリソース ID、エラー時に **FALSE** を返します。

---

openssl_pkey_get_public

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_get_public — 証明書から公開鍵を抽出し、使用できるようにする

説明

resource **openssl_pkey_get_public** (mixed $certificate)

openssl_get_publickey() は公開鍵を certificate から抽出し、 他の関数で使用できるよう準備します。

パラメータ

certificate

certificate は以下のいずれかです。

1. an X.509 証明書リソース
2. file://path/to/file.pem 形式の文字列。 このファイルは、PEM エンコードされた証明書/秘密鍵である必要が あります（両方を含むことも可能です）。
3. PEM フォーマットの秘密鍵。

返り値

成功した場合に正のキーリソース ID、エラー時に **FALSE** を返します。

---

openssl_pkey_new

(PHP 4 >= 4.2.0, PHP 5)

openssl_pkey_new — 新規に秘密鍵を生成する

説明

resource **openssl_pkey_new** ([array $configargs])

openssl_pkey_new() は、 新しい秘密鍵と公開鍵の鍵ペアを作成します。鍵の公開部は、 openssl_pkey_get_public() を使用して取得可能です。

> **注意**: この関数を正しく機能させるには、有効な openssl.cnf がインストールされている必要があります。詳細な情報については インストールセクション の下にある注記を参照ください。

パラメータ

configargs

鍵の作成方法の詳細（ビット数など）を指定するには、 configargs を使用します。 configargs の詳細な情報については openssl_csr_new() を参照ください。

返り値

成功した場合に秘密鍵のリソース ID、 エラー時に **FALSE** を返します。

---

openssl_private_decrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_private_decrypt — 秘密鍵でデータを復号する

説明

bool ***openssl_private_decrypt*** *(string $data , string &$decrypted , mixed $key [, int $padding])*

openssl_private_decrypt() *は、事前に openssl_public_encrypt() で暗号化された data を復号し、それを decrypted に格納します。*

これを使用するのは、例えばあなたにのみ送られてきたデータを復号する場合です。

パラメータ

data

decrypted

key

key は、 データの暗号化に使用したものに対応する秘密鍵である必要があります。

padding

padding のデフォルトは ***OPENSSL_PKCS1_PADDING*** *ですが、 これ以外にも* ***OPENSSL_SSLV23_PADDING***、 ***OPENSSL_PKCS1_OAEP_PADDING***、 ***OPENSSL_NO_PADDING*** *が指定可能です。*

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *openssl_public_encrypt()*
- *openssl_public_decrypt()*

---

openssl_private_encrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_private_encrypt — 秘密鍵でデータを暗号化する

説明

bool ***openssl_private_encrypt*** *(string $data , string &$crypted , mixed $key [, int $padding])*

openssl_private_encrypt() *は、data を秘密鍵 key で暗号化し、それを crypted に格納します。暗号化されたデータは、openssl_public_decrypt() を用いて復号可能です。*

この関数を使用するのは、例えばデータ（あるいはその一部）に署名をし、 それが他人によって書かれたものでないことを証明する場合です。

パラメータ

data

crypted

key

padding

padding のデフォルトは ***OPENSSL_PKCS1_PADDING*** *ですが、これ以外に* ***OPENSSL_NO_PADDING*** *が指定可能です。*

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *openssl_public_encrypt()*
- *openssl_public_decrypt()*

---

openssl_public_decrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_public_decrypt — 公開鍵でデータを復号する

説明

bool ***openssl_public_decrypt*** *(string $data , string &$decrypted , mixed $key [, int $padding])*

openssl_public_decrypt() *は、事前に openssl_private_encrypt() で暗号化された data を復号し、それを decrypted に格納します。*

これを使用するのは、例えばメッセージの作者が秘密鍵の所有者であるかどうかを調べる場合です。

パラメータ

data

decrypted

key

key は、 データの暗号化に使用したものに対応する公開鍵である必要があります。

padding

padding のデフォルトは **OPENSSL_PKCS1_PADDING** ですが、 これ以外にも **OPENSSL_NO_PADDING** が指定可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openssl_private_encrypt()
- openssl_private_decrypt()

---

openssl_public_encrypt

(PHP 4 >= 4.0.6, PHP 5)

openssl_public_encrypt — 公開鍵でデータを暗号化する

説明

bool **openssl_public_encrypt** (*string $data , string &$crypted , mixed $key [, int $padding]*)

openssl_public_encrypt() は、*data* を公開鍵 *key* で暗号化し、それを *crypted* に格納します。暗号化されたデータは openssl_private_decrypt() を用いて復号可能です。

この関数を使用するのは、例えば秘密鍵の所有者にのみ読めるようにメッセージを暗号化する場合です。 また、データベースに格納するデータを安全な形式にするためにも使用されます。

パラメータ

data

crypted

暗号化した結果がここに格納されます。

key

公開鍵。

padding

padding のデフォルトは **OPENSSL_PKCS1_PADDING** ですが、これ以外に **OPENSSL_SSLV23_PADDING**、 **OPENSSL_PKCS1_OAEP_PADDING**、 **OPENSSL_NO_PADDING** が指定可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- openssl_private_encrypt()
- openssl_private_decrypt()

---

openssl_seal

(PHP 4 >= 4.0.4, PHP 5)

openssl_seal — データをシール(暗号化)する

説明

int **openssl_seal** (*string $data , string &$sealed_data , array &$env_keys , array $pub_key_ids*)

openssl_seal() は、ランダムに生成された秘密鍵により *RC4* を使用して *data* をシール(暗号化) します。このキーは、*pub_key_ids* を *ID* とする 公開鍵で暗号化されます。これは、暗号化されたデータを複数の受信者に 送信できることを意味します（この際、各受信者は送信側に公開鍵を提供します）。各受信者は、暗号化されたデータとその受信者の 公開鍵で暗号化されたエンベロープキーを受け取る必要があります。

パラメータ

data

sealed_data

env_keys

pub_key_ids

返り値

成功時にシール(暗号化)されたデータの長さ、エラー時に ***FALSE*** *を返します。成功時には、暗号化されたデータが* *sealed_data* *に、エンベロープキーが* *env_keys* *に返されます。*

例

Example#1 openssl_seal() の例

```
<?php
// $data には、暗号化されるデータが含まれていると仮定

// 受信者の公開鍵を取得し、使用可能にする
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// 2 番目の受信者についても同様
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// メッセージを暗号化。$pk1 および $pk2 の所有者のみが、$sealed を
// データをそれぞれ $ekeys[0] および $ekeys[1] で復号化することが
// 可能
openssl_seal($data, $sealed, $ekeys, array($pk1, $pk2));

// キーをメモリから開放する
openssl_free_key($pk1);
openssl_free_key($pk2);
?>
```

参考

- openssl_open()

---

openssl_sign

(PHP 4 >= 4.0.4, PHP 5)

openssl_sign — *署名を生成する*

説明

bool ***openssl_sign*** *(string $data , string &$signature , mixed $priv_key_id [, int $signature_alg])*

openssl_sign() *は、指定した* *data* *に関してハッシュ計算に* *SHA1* *を使用して 署名を計算し、その後* *priv_key_id* *で 指定した秘密鍵を使用して暗号化を行います。*data* *自体は 暗号化されないことに注意してください。*

パラメータ

data

signature

成功した場合、署名が *signature* *に格納されます。*

priv_key_id

signature_alg

デフォルト値は ***OPENSSL_ALGO_SHA1*** *です。 詳細な情報は、* 署名アルゴリズム *の一覧を参照ください。*

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

変更履歴

バージョン	説明
5.0.0	*signature_alg* パラメータが追加されました。

例

Example#1 openssl_sign() の例

```
<?php
// $data に署名するデータが含まれていると仮定

// ファイルから公開鍵を取得し、使用可能とする
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);
```

```
// 署名を計算
openssl_sign($data, $signature, $pkeyid);

// メモリからキーを開放する
openssl_free_key($pkeyid);
?>
```

参考

- openssl_verify()

---

openssl_verify

(PHP 4 >= 4.0.4, PHP 5)

openssl_verify — 署名を検証する

説明

int **openssl_verify** (string $data , string $signature , mixed $pub_key_id [, int $signature_alg])

openssl_verify() は、 pub_key_id が指す公開鍵を使用し、 指定した data に関して signature が正しいことを確認します。 署名が正しいと判定されるためには、 その公開鍵が署名の際に使用した秘密鍵に対応していることを必要とします。

パラメータ

data

signature

pub_key_id

signature_alg

デフォルト値は **OPENSSL_ALGO_SHA1** です。 詳細な情報は、 署名アルゴリズム の一覧を参照ください。

返り値

署名 (signature) が正しい場合に 1、正しくない場合に 0、 エラーの場合に -1 を返します。

変更履歴

バージョン	説明
5.0.0	signature_alg パラメータが追加されました。

例

Example#1 openssl_verify() の例

```
<?php
// $data および $signature はデータおよび署名が含まれていると仮定

// 認証局から公開鍵を取得し、使用可能にする
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// 署名が正しいかどうかを確認
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1) {
    echo "正しいです";
} elseif ($ok == 0) {
    echo "正しくありません";
} else {
    echo "署名を確認する際にエラーが発生しました";
}
// メモリからキーを開放
openssl_free_key($pubkeyid);
?>
```

参考

- openssl_sign()

---

openssl_x509_check_private_key

(PHP 4 >= 4.2.0, PHP 5)

openssl_x509_check_private_key — 秘密鍵が証明書に対応するかを確認する

説明

bool **openssl_x509_check_private_key** (mixed $cert , mixed $key)

key が cert に対応する秘密鍵かどうかを調べます。

パラメータ

cert

証明書。

key

秘密鍵。

返り値

key が cert に対応する秘密鍵の場合に TRUE、それ以外の場合に FALSE を返します。

---

openssl_x509_checkpurpose

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_checkpurpose — 証明書が特定の目的に使用可能かどうか確認する

説明

int **openssl_x509_checkpurpose** (mixed $x509cert , int $purpose [, array $cainfo [, string $untrustedfile]])

openssl_x509_checkpurpose() は証明書を調べ、 purpose で指定した目的に使用可能であるかどうかを確認します。

パラメータ

x509cert

調べたい証明書。

purpose

openssl_x509_checkpurpose() の目的

定数	説明
X509_PURPOSE_SSL_CLIENT	この証明書を SSL 接続のクライアント側で使用できるか?
X509_PURPOSE_SSL_SERVER	この証明書を SSL 接続のサーバ側で使用できるか?
X509_PURPOSE_NS_SSL_SERVER	この証明書を Netscape SSL サーバで使用できるか?
X509_PURPOSE_SMIME_SIGN	この証明書を S/MIME email で使用できるか?
X509_PURPOSE_SMIME_ENCRYPT	この証明書を S/MIME email の暗号化で使用できるか?
X509_PURPOSE_CRL_SIGN	この証明書を証明書取消リスト(CRL)にサインをする際に使用できるか?
X509_PURPOSE_ANY	この証明書をあらゆる用途に使用できるか?

これらのオプションはビットフィールドではありません。 指定できるのは一つだけです!

cainfo

cainfo は、証明書の認証 で説明したような信頼できる CA ファイル/ディレクトリの配列です。 デフォルトは空の配列です。

untrustedfile

指定した場合は、これが証明書を含むPEMエンコードされたファイルの名前になります。 この証明書は、検証と証明を簡単化するために使用されます。 そのファイル内にある証明書は、信頼されない(untrusted)証明書とみなされます。

返り値

証明書が意図した目的に使用可能である場合に TRUE、使用できない場合に FALSE 、エラーの場合に -1 を返します。

---

openssl_x509_export_to_file

(PHP 4 >= 4.2.0, PHP 5)

openssl_x509_export_to_file — 証明書をファイルにエクスポートする

説明

bool **openssl_x509_export_to_file** (mixed $x509 , string $outfilename [, bool $notext])

openssl_x509_export_to_file() は、 PEM エンコード形式の x509 をファイル outfilename に保存します。

パラメータ

x509

outfilename

出力ファイルへのパス。

notext

オプションのパラメータ notext は、 出力内容の冗長性に影響します。FALSE を設定すると、人間が理解できる形式の 追加情報が出力に付加されま

す。notext の デフォルト値は **TRUE** です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

openssl_x509_export

(PHP 4 >= 4.2.0, PHP 5)

openssl_x509_export — 証明書を文字列としてエクスポートする

説明

bool **openssl_x509_export** (mixed $x509 , string &$output [, bool $notext])

openssl_x509_export() は、 PEM エンコード形式の x509 を文字列 output に保存します。

パラメータ

x509

output

成功した場合に、ここに PEM が格納されます。

notext

オプションのパラメータ notext は、 出力内容の冗長性に影響します。**FALSE** を設定すると、人間が理解できる形式の 追加情報が出力に付加されます。notext の デフォルト値は **TRUE** です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

openssl_x509_free

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_free — 証明書リソースを開放する

説明

void **openssl_x509_free** (resource $x509cert)

openssl_x509_free() は、メモリから指定した x509cert リソースに関連した証明書を開放します。

パラメータ

x509cert

返り値

値を返しません。

---

openssl_x509_parse

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_parse — X509 証明書をパースし、配列として情報を返す

説明

array **openssl_x509_parse** (mixed $x509cert [, bool $shortnames])

openssl_x509_parse() は、指定した x509cert に関する情報を返します。この情報には 題名、発行者名、目的、発効日および有効期限等が含まれます。

パラメータ

x509cert

shortnames

shortnames は、配列中でのデータの添字付けの 方法を設定します。shortnames が **TRUE** の場合 (デフォルト)、フィールドは短縮型で添字が付けられます。そうでない場合、 長い名前が使用されます。例えば、CN は commonName の短縮型です。

返り値

返されるデータの構造については（わざと）まだ文書化していません。 これは、今後もデータの構造が変更される可能性があるためです。

---

openssl_x509_read

(PHP 4 >= 4.0.6, PHP 5)

openssl_x509_read — X.509 証明書をパースし、リソース ID を返す

説明

resource **openssl_x509_read** (mixed $x509certdata)

openssl_x509_read() は、 x509certdata で指定した証明書をパースし、 その証明書用のリソースIDを返します。

パラメータ

x509certdata

返り値

成功した場合にリソース ID、失敗した場合に **FALSE** を返します。

目次

Oracle 関数 [推奨されません]

導入

警告

This extension is deprecated. Instead, use the improved oci8 extension. Functions documented here do list recommended oci8 alternatives. この拡張モジュールは推奨されません。代わりに、改善された oci8 拡張モジュールを使用してください。 ここでドキュメント化されている関数群には、代替として推奨される oci8 関数も一覧しています。

この拡張モジュールにより、Oracle データベースサーバへのアクセスがサポートされます。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.0.

インストール手順

オプション--with-oracle[=DIR]を指 定してPHPをコンパイルする必要があります。ただし、DIRのデフォルトは、 環境変数ORACLE_HOMEの値です。

リソース型

この拡張モジュールでは、Oracle 接続 ID と Oracle カーソルインデックスのふたつのリソースを定義しています。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

ORA_BIND_INOUT (integer)
ORA_BIND_IN (integer)
ORA_BIND_OUT (integer)
ORA_FETCHINTO_ASSOC (integer)
ORA_FETCHINTO_NULLS (integer)

---

ora_bind

(PHP 4, PHP 5 <= 5.0.5)

ora_bind — PHP 変数を Oracle パラメータにバインドする

説明

bool **ora_bind** (resource $cursor , string $phpvar , string $sqlparam , int $length [, int $type])

指定した名前の PHP 変数を SQL パラメータとバインドします。

ora_bind() は ora_parse() の後でかつ ora_exec() の前に呼び出されなくてはなりません。 入力値はバインドされた PHP 変数に代入することにより指定することが可能です。 ora_exec() のコールによりバインド先の SQL パラメータに値が出力される場合には、バインドされた PHP 変数に出力値が代入されます。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

phpvar

バインドする PHP 変数。

sqlparam

SQL パラメータ。:name 形式でなければなりません。

length

type

パラメータの型を指定します。デフォルトは **ORA_BIND_INOUT** です。とりうる値を以下にまとめます。

定数	値
ORA_BIND_INOUT	0
ORA_BIND_IN	1
ORA_BIND_OUT	2

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。
3.0.1	type 用の定数が追加されました。 以前のバージョンでは、数値を指定する必要がありました。

例

Example#1 ora_bind() の例

```
<?php
   ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
```

```
  ora_bind($curs, "result", ":x", $len, 2);
  ora_bind($curs, "input", ":in", 5, 1);
  ora_bind($curs, "output", ":out", 5, 2);
  $input = 765;
  ora_exec($curs);
  echo "Result: $result<br />Out: $output<br />In: $input";
?>
```

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_bind_by_name()
- oci_bind_array_by_name()

---

ora_close

(PHP 4, PHP 5 <= 5.0.5)

ora_close — Oracle カーソルをクローズする

説明

bool ***ora_close*** *(resource $cursor)*

データカーソル cursor をクローズします。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。*

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_free_statement()

---

ora_columnname

(PHP 4, PHP 5 <= 5.0.5)

ora_columnname — Oracle 結果カラムの名前を取得する

説明

string ***ora_columnname*** *(resource $cursor , int $column)*

カーソル cursor 上のカラムの名前を返します。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

column

カラム番号。最初は 0 です。

返り値

名前を表す文字列を返します。この値はすべて大文字となります。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_field_name()*

---

ora_columnsize

(PHP 4, PHP 5 <= 5.0.5)

ora_columnsize — Oracle の結果カラムのサイズを返す

説明

*int **ora_columnsize** (resource $cursor , int $column)*

カーソル cursor 上の Oracle カラム column のサイズを返します。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

column

カラム番号。最初は 0 です。

返り値

*サイズを表す整数値、あるいはエラー時に **FALSE** を返します。*

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_field_size()*

---

ora_columntype

(PHP 4, PHP 5 <= 5.0.5)

ora_columntype — Oracle のカラムの型を取得する

説明

*string **ora_columntype** (resource $cursor , int $column)*

カーソル cursor 上の、フィールド/カラム column の Oracle データ型名を返します。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

column

カラム番号。最初は 0 です。

返り値

返された型名は、次のどれかになります。

- *"VARCHAR2"*
- *"VARCHAR"*
- *"CHAR"*
- *"NUMBER"*
- *"LONG"*
- *"LONG RAW"*
- *"ROWID"*
- *"DATE"*
- *"CURSOR"*

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_field_type()
- oci_field_type_raw()

---

ora_commit

(PHP 4, PHP 5 <= 5.0.5)

ora_commit — Oracle トランザクションをコミットする

説明

bool **ora_commit** (resource $connection)

Oracle のトランザクションをコミットします。

トランザクションは、直近のコミット/ロールバック、またはオートコミットがオフになった時点、 または接続が確立された時点から指定した接続に加えられた全ての変更として定義されます。

パラメータ

connection

ora_logon() でオープンした接続 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーについての詳細は、ora_error() および **ora_errorcode ()** 関数を用いて調べられます。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_commit()

参考

- ora_commiton()
- ora_commitoff()

---

ora_commitoff

(PHP 4, PHP 5 <= 5.0.5)

ora_commitoff — オートコミットをオフにする

説明

bool **ora_commitoff** (resource $connection)

各 ora_exec() の後のオートコミットをオフにします。

パラメータ

connection

ora_logon() でオープンした接続 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーに関する詳細な情報は、関数 ora_error() および ora_errorcode() により取得することが可能です。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- ***OCI_DEFAULT** フラグを oci_execute() に渡す*

参考

- *ora_commiton()*
- *ora_commit()*

---

ora_commiton

(PHP 4, PHP 5 <= 5.0.5)

ora_commiton — オートコミットを有効にする

説明

*bool **ora_commiton** (resource $connection)*

指定した接続において ora_exec() 実行後のオートコミットをオンにします。

パラメータ

connection

ora_logon() でオープンした接続 ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。*

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_execute()*

参考

- *ora_commitoff()*
- *ora_commit()*

---

ora_do

(PHP 4, PHP 5 <= 5.0.5)

ora_do — パース、実行、フェッチを行う

説明

*resource **ora_do** (resource $connection , string $query)*

クエリをパース、実行し、結果の最初のレコードを取得します。

この関数は、ora_parse()、 ora_exec() および ora_fetch() を組み合わせたものです。

パラメータ

connection

ora_logon() でオープンした接続 ID。

query

SQL 文。

返り値

この関数はカーソルインデックスを返し、エラー時に **FALSE** を返します。 エラーに関する詳細は、 ora_error() および ora_errorcode() 関数を用いて取得することが可能です。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

ora_do() を直接置き換えるような関数は oci8 にはありません。 そのかわりに、oci_parse()、**oci_excute()** そして oci_fetch_*() 関数のいずれかを使用します。

参考

- ora_parse()
- ora_exec()
- ora_fetch()

---

ora_error

(PHP 4, PHP 5 <= 5.0.5)

ora_error — Oracle エラーメッセージを取得する

説明

string **ora_error** ([resource $cursor_or_connection])

指定したカーソルあるいは接続上で直近に実行された文のエラーメッセージを返します。

パラメータ

cursor_or_connection

Oracle カーソルあるいは接続 ID。

返り値

XXX-NNNNN という形式のエラーメッセージが返されます。ここで、 XXX は何処からエラーが発生したか、 NNNNN はエラーメッセージを識別します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。
3.0.4	接続 ID のサポートが追加されました。

例

Unix 版の Oracle 上では、 以下のようにエラーメッセージに関する詳細を得ることができます。

```
$> oerr ora 00001
```

上の例の出力は、たとえば以下のようになります。

```
00001, 00000,
"unique constraint (%s.%s) violated" // *Cause: An update or insert
statement attempted to insert a duplicate key //           For Trusted
ORACLE configured in DBMS MAC mode, you may see //           this message
if a duplicate entry exists at a different level.  // *Action: Either
remove the unique restriction or do not insert the key
```

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_error()

---

ora_errorcode

(PHP 4, PHP 5 <= 5.0.5)

ora_errorcode — Oracle エラーコードを取得する

説明

int ***ora_errorcode*** *([resource $cursor_or_connection])*

指定したカーソルまたは接続において最後に実行したステートメントの エラーコード番号を返します。

パラメータ

cursor_or_connection

Oracle カーソルあるいは接続 ID。

返り値

エラーコードを整数で返します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。
3.0.4	接続 ID のサポートが追加されました。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_error() (結果の配列の "code" フィールド)*

---

ora_exec

(PHP 4, PHP 5 <= 5.0.5)

ora_exec — Oracle カーソル上でのパースしたステートメントを実行する

説明

bool ***ora_exec*** *(resource $cursor)*

あらかじめパースされた文 cursor を実行します。

パラメータ

cursor

すでに ora_parse() でパース済みの Oracle カーソル。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。*

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_execute()*

参考

- *ora_parse()*
- *ora_fetch()*
- *ora_do()*

---

ora_fetch_into

(PHP 4, PHP 5 <= 5.0.5)

ora_fetch_into — 指定した配列 result にレコードを取得する

説明

int ***ora_fetch_into*** *(resource $cursor , array &$result [, int $flags])*

データの行を取得し、配列に格納します。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

result

参照渡しの配列。ここに取得した値が格納されます。

flags

二種類のフラグがあります。 **ORA_FETCHINTO_NULLS** が指定された場合、 **NULL** 値を持つカラムは配列にセットされますが、 **ORA_FETCHINTO_ASSOC** が指定された場合、 連想配列が生成されます。

返り値

取得したカラム数、あるいはエラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

例

Example#1 ora_fetch_into()

```
<?php
$results = array();
ora_fetch_into($cursor, $results);
echo $results[0];
echo $results[1];
$results = array();
ora_fetch_into($cursor, $results, ORA_FETCHINTO_NULLS|ORA_FETCHINTO_ASSOC);
echo $results['MyColumn'];
?>
```

注意

廃止予定の *oracle* のかわりに *oci8* 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_fetch_array()* のような取得用関数

参考

- *ora_parse()*
- *ora_exec()*
- *ora_fetch()*
- *ora_do()*

---

ora_fetch

(PHP 4, PHP 5 <= 5.0.5)

ora_fetch — カーソルから一行分のデータを取得する

説明

bool **ora_fetch** (*resource $cursor*)

指定した *cursor* から一行分のデータを取得します。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

返り値

(列が取り出されれば) **TRUE** もしくは (取り出す列がもうないか、 エラーが発生した場合) **FALSE** を返します。エラーが発生した場合、 詳細は *ora_error()* および *ora_errorcode()*関数を用いて調べることが できます。エラーがない場合、*ora_errorcode()* は *0* を返します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の *oracle* のかわりに *oci8* 拡張モジュールを用いる場合は、以下を使用してください。

- oci_fetch()

参考

- ora_parse()
- ora_exec()
- ora_do()

---

ora_getcolumn

(PHP 4, PHP 5 <= 5.0.5)

ora_getcolumn — 取得した行からデータを得る

説明

string **ora_getcolumn** (resource $cursor , int $column)

カラムのデータもしくは関数の結果を取り出します。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

column

カラム番号を表す整数値。

返り値

カラムデータを返します。エラーが発生した場合には **FALSE** を返し、ora_errorcode() は 0 以外の値を返します。 しかしながら、この関数の返り値が**FALSE**となるかどうかを調べる際には、 エラーでない場合(返り値がヌル、空白文字列、数字の0、文字列の"0")にも 判定式の値が真となる可能性があることに注意してください。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_result()

---

ora_logoff

(PHP 4, PHP 5 <= 5.0.5)

ora_logoff — Oracle 接続を閉じる

説明

bool **ora_logoff** (resource $connection)

ユーザのログアウトを行い、サーバとの接続を閉じます。

パラメータ

connection

ora_logon() でオープンした接続 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_close()

参考

- ora_logon()

---

ora_logon

(PHP 4, PHP 5 <= 5.0.5)

ora_logon — Oracle 接続をオープンする

説明

resource **ora_logon** (string $user , string $password)

指定したユーザ名とパスワードにより PHP と Oracle データベースの接続を確立します。

次のように TNS 名を user に与えることにより、接続は SQL*Net を用いて作成されます。

```
<?php
$conn = Ora_Logon("user@TNSNAME", "pass");
?>
```

非アスキー文字を含む文字データがある場合、環境変数 NLS_LANG を設定する必要があります。サーバモジュールの場合、 サーバの起動前にサーバの環境変数として設定する必要があります。

パラメータ

user

ユーザ名。

password

user に対応するパスワード。

返り値

成功した場合に接続 ID、失敗した場合に **FALSE** を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_connect()
- oci_new_connect()

---

ora_numcols

(PHP 4, PHP 5 <= 5.0.5)

ora_numcols — カラム数を返す

説明

int **ora_numcols** (resource $cursor)

結果のカラム数を返します。 parse/exec/fetch シーケンスの後でのみ意味がある値が返されます。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

返り値

カラムの数を表す整数値、あるいはエラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_num_fields()*

参考

- *ora_parse()*
- *ora_exec()*
- *ora_fetch()*
- *ora_do()*

---

ora_numrows

(PHP 4, PHP 5 <= 5.0.5)

ora_numrows — レコード数を返す

説明

```
int ora_numrows ( resource $cursor )
```

結果の行数を返します。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

返り値

行の数を表す整数値、あるいはエラー時に FALSE を返します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_num_rows()*

---

ora_open

(PHP 4, PHP 5 <= 5.0.5)

ora_open — Oracle カーソルをオープンする

説明

```
resource ora_open ( resource $connection )
```

指定した接続に関して Oracle カーソルをオープンします。

パラメータ

connection

ora_logon() でオープンした接続 ID。

返り値

カーソルインデックスもしくは失敗した場合に FALSE を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_new_cursor()

---

ora_parse

(PHP 4, PHP 5 <= 5.0.5)

ora_parse — Oracle の SQL ステートメントをパースする

説明

bool **ora_parse** (resource $cursor , string $sql_statement [, int $defer])

SQL 文または PL/SQL ブロックをパースし、 指定したカーソル cursor に関連付けます。

パラメータ

cursor

ora_open() でオープンした Oracle カーソル。

sql_statement

SQL 文あるいは PL/SQL ブロック。

defer

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- oci_parse()

参考

- ora_exec()
- ora_fetch()
- ora_do()

---

ora_plogon

(PHP 4, PHP 5 <= 5.0.5)

ora_plogon — 持続的な Oracle 接続をオープンする

説明

resource **ora_plogon** (string $user , string $password)

PHP と Oracle データベースの間に持続的な接続を確立します。

パラメータ

user

ユーザ名。

password

user に対応するパスワード。

返り値

持続的な接続 ID、あるいはエラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_pconnect()*

参考

- *ora_logon()*

---

ora_rollback

(PHP 4, PHP 5 <= 5.0.5)

ora_rollback — トランザクションをロールバックする

説明

*bool **ora_rollback** (resource $connection)*

Oracle のトランザクションをロールバックします。

トランザクションの定義については ora_commit() を参照ください。

パラメータ

connection

ora_logon() でオープンした接続 ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーについての詳細は、ora_error() および ora_errorcode() 関数を用いて調べられます。*

変更履歴

バージョン	説明
5.1.0	oracle 拡張モジュールは非推奨です。かわりに oci8 を使用してください。

注意

廃止予定の oracle のかわりに oci8 拡張モジュールを用いる場合は、以下を使用してください。

- *oci_rollback()*

目次

- ora_plogon — 持続的な Oracle 接続をオープンする
- ora_rollback — トランザクションをロールバックする

Oracle 関数 (PDO_OCI)

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

PDO_OCI は、OCI ライブラリを通じて PHP から Oracle データベースへのアクセスを可能にするための PHP Data Objects (PDO) インターフェース を実装したドライバです。

PDO_OCI DSN

(No version information available, might be only in CVS)

PDO_OCI DSN — Oracle データベースに接続する

説明

PDO_OCI データソース名 (DSN) は以下の要素で構成されます:

DSN 接頭辞

DSN 接頭辞は **oci:** です。

dbname (Oracle Instant Client)

Oracle Instant Client 接続に対する URI は dbname=//hostname:port-number/database のような形式です。 もし tnsnames.ora で定義されたデータベースに接続する場合は、データベース名のみを使用して dbname=database とします。

charset

現在の環境ハンドルにおける、クライアント側の文字セットです。

例

Example#1 PDO_OCI DSN の例

以下の例は、Oracle データベースに接続するための PDO_OCI DSN を表します:

```
// tnsnames.ora で定義したデータベースに接続します
oci:dbname=mydb

// Oracle Instant Client を使用して接続します
oci:dbname=//localhost:1521/mydb
```

出力制御関数(output control)

導入

出力制御関数により、スクリプトから送信される出力を制御することが可 能になります。この機能は、複数の異なった場面、特にスクリプトがデー タ出力を開始した後にヘッダをブラウザに送信する必要がある場合に有用 です。出力制御関数は、header() または setcookie()を使用して送信されたヘッダには作用せ ず、echo() のような関数とPHPコードのブロック間 のデータにのみ作用します。

> **注意**: PHP 4.1.x (および 4.2.x) から 4.3.x に更新する際、前のバージョンのバグのせいで、 php.ini の implict_flush を OFF にする必要があります。さもないと、 ob_start() を使用する全ての出力は、 出力を抑制することができなくなります。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

出力制御設定オプション

名前	デフォルト	変更の可否	変更履歴
output_buffering	"0"	PHP_INI_PERDIR	
output_handler	NULL	PHP_INI_PERDIR	PHP 4.0.4 以降で使用可能です。
implicit_flush	"0"	PHP_INI_ALL	PHP <= 4.2.3 では PHP_INI_PERDIR です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

output_buffering boolean/integer

このディレクティブを 'On' と設定することにより、全てのファイルに 関して出力バッファリングを有効にすることができます。 特定の大きさにバッファの大きさを制限したい場合、このディレクティブの 値として 'On' の代わりに最大バイト数(例:output_buffering=4096) を使用することができます。 PHP 4.3.5 以降、PHP-CLI ではこのディレクティブが常に Off となります。

output_handler string

スクリプトの全ての出力を関数にリダイレクトすることができます。 例えば、output_handler に mb_output_handler() を指定した場合、文字エンコーディングは透過的に指定したエンコーディングに 変換されます。出力ハンドラを指定することにより自動的に出力 バッファリングを on にします。

> ***注意**: ob_iconv_handler() と mb_output_handler() の両方で使用することは できません。また、 ob_gzhandler() と zlib.output_compression の両方を使用することはできません。*
>
> ***注意**: このディレクティブには、組み込み関数のみが使用可能です。ユーザ定義の 関数については、ob_start() を使用してください。*

implicit_flush boolean

*デフォルトは **FALSE** です。これを **TRUE** に変更すると、PHP が 各出力ブロックの後で自動的に出力レイヤをフラッシュするよう 指定します。これは、各 print() および HTML ブロックの後で flush() 関数をコールすることと等価です。*

*Web 環境の中で PHP を使用している時に このオプションを on に変更すると、著しい性能低下が生じるため、 通常はデバッグ目的のみにすることが推奨されます。CLI SAPI のもとで実行される時、この値はデフォルトで **TRUE** になっています。*

ob_implicit_flush() も参照ください。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

例

Example#1 出力制御の例

```
<?php

ob_start();
echo "Hello¥n";

setcookie("cookiename", "cookiedata");

ob_end_flush();

?>
```

上記の例では、echo()からの出力は、 ob_end_flush() がコールされるまで出力バッファに 保存されます。この際、 setcookie()をコールするとエラーを発生することな くクッキーが保存されます。(通常、データの送信後はブラウザにヘッダ を送信することはできません。)

参考

header()およびsetcookie() も参照ください。

---

flush

(PHP 4, PHP 5)

flush — 出力バッファをフラッシュする

説明

*void **flush** (void)*

PHP および PHP が使っている(CGI, Web サーバなどの)バックエンドの 出力バッファをフラッシュします。これにより、それまでの全ての 出力がユーザのブラウザに対して効率的に出力されます。

***flush()** は、Webサーバまたはクライアント側の ブラウザのバッファリングの手法に影響を与えません。 そのため、出力バッファをフラッシュするには ob_flush() および **flush()** の両方をコールする必要があります。*

いくつかのサーバ、特に Win32 上ではスクリプトからの出力をブラウザに 結果を送信する前にスクリプトが終了するまでバッファに溜めることがあります。

*mod_gzip のような Apache 用のサーバモジュールはそれ自体がバッファリングを行います。 そのため、**flush()** をコールしても 即時にデータを*

クライアントに送信しないという結果につながります。

ブラウザ側で表示前に入力をバッファリングすることもあり得ます。 Netscape では例えば改行または開始タグを受信するまでテキストは バッファリングされ、最も外側のテーブルの </table> タグが現れるまでテーブルは描画されません。

いくつかのバージョンの Microsoft Internet Explorer は、256 バイトの 出力を受けてからページを表示し始めます。このため、これらのブラウザに ページを表示させるには、フラッシュする前に余分な空白を送信する 必要があるかもしれません。

返り値

値を返しません。

---

ob_clean

(PHP 4 >= 4.2.0, PHP 5)

ob_clean — 出力バッファをクリア(消去)する

説明

void **ob_clean** (void)

この関数は、出力バッファの内容を消去します。

この関数は、ob_end_clean() のように出力バッファを破棄するわけではありません。

返り値

値を返しません。

参考

- ob_flush()
- ob_end_flush()
- ob_end_clean()

---

ob_end_clean

(PHP 4, PHP 5)

ob_end_clean — 出力用バッファをクリア(消去)し、出力のバッファリングをオフにする

説明

bool **ob_end_clean** (void)

出力用バッファの内容を消去し、出力のバッファリングをオフにします。 バッファの内容について更に何らかの処理を行いたい場合には、 バッファの内容は **ob_end_clean()** がコールされると 破棄されるため、**ob_end_clean()** の前に ob_get_contents() をコールしなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 失敗する原因は、まず、アクティブなバッファ以外に対して この関数をコールしたか、あるいは何らかの理由により バッファを消去することができなかった場合です（特殊なバッファ等）。

エラー / 例外

この関数は失敗すると **E_NOTICE** レベルのエラーを発行します。

変更履歴

バージョン	説明
4.2.0	boolean 型の戻り値が追加されました。

例

以下は全ての出力バッファを消去する簡単な方法の例です。

Example#1 ob_end_clean() の例

```
<?php
ob_start();
echo 'Text that won¥'t get displayed.';
ob_end_clean();
?>
```

参考

- ob_start()
- ob_get_contents()
- ob_flush()

---

ob_end_flush

(PHP 4, PHP 5)

ob_end_flush — 出力用バッファをフラッシュ(送信)し、出力のバッファリングをオフにする

説明

bool **ob_end_flush** (void)

この関数は、出力用バッファの内容を出力し、出力のバッファリングをオフにします。 出力用バッファの内容を更に処理したい場合には、 バッファの内容は **ob_end_flush()** がコールされた後に消去されるため、 **ob_end_flush()** の前に ob_get_contents() をコールする必要があります。

注意: この関数は ob_get_flush() に似ていますが、 ob_get_flush() はバッファを文字列として返すという点が違います。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 失敗する原因は、まず、アクティブなバッファ以外に対してこの 関数をコールしたか、あるいは何らかの理由により バッファを消去することができなかった場合です（特殊なバッファ等）。

エラー / 例外

この関数は失敗すると **E_NOTICE** レベルのエラーを発行します。

変更履歴

バージョン	説明
4.2.0	boolean 型の戻り値が追加されました。

例

Example#1 ob_end_flush() example

以下は全ての出力バッファをフラッシュし終了する簡単な方法の例です。

```
<?php
  while (@ob_end_flush());
?>
```

参考

- ob_start()
- ob_get_contents()
- ob_get_flush()
- ob_flush()
- ob_end_clean()

---

ob_flush

(PHP 4 >= 4.2.0, PHP 5)

ob_flush — 出力バッファをフラッシュ(送信)する

説明

void **ob_flush** (void)

この関数は、(ある場合に)出力バッファの内容を送信します。 更にバッファの内容を処理したい場合には、**ob_flush()** がコールされた後にバッファの内容が破棄されるので、 **ob_flush()** の前に ob_get_contents() をコールする必要があります。

この関数は、ob_end_flush() のように出力バッファを破棄しません。

返り値

値を返しません。

参考

- ob_get_contents()
- ob_clean()
- ob_end_flush()
- ob_end_clean()

---

ob_get_clean

(PHP 4 >= 4.3.0, PHP 5)

ob_get_clean — 現在のバッファの内容を取得し、出力バッファを削除する

説明

string **ob_get_clean** (void)

現在のバッファの中身を取得し、出力バッファを削除します。

ob_get_clean() は、基本的に ob_get_contents() および ob_end_clean() を同時に実行するのと同じです。

返り値

出力バッファの内容を返した後で出力のバッファリングを終了します。 出力バッファリングが開始されていない場合は **FALSE** が返されます。

例

Example#1 単純な ob_get_clean() の例

```
<?php

ob_start();

echo "Hello World";

$out = ob_get_clean();
$out = strtolower($out);

var_dump($out);
?>
```

上の例の出力は以下となります。

```
string(11) "hello world"
```

参考

- ob_get_contents()
- ob_start()

---

ob_get_contents

(PHP 4, PHP 5)

ob_get_contents — 出力用バッファの内容を返す

説明

string **ob_get_contents** (void)

出力用バッファの内容を取得します。バッファの内容はクリアしません。

返り値

出力用バッファの内容を返します。 出力のバッファリングがアクティブでない場合には **FALSE** を返します。

例

Example#1 単純な ob_get_contents() の例

```
<?php

ob_start();

echo "Hello ";

$out1 = ob_get_contents();

echo "World";

$out2 = ob_get_contents();

ob_end_clean();

var_dump($out1, $out2);
?>
```

上の例の出力は以下となります。

```
string(6) "Hello "
string(11) "Hello World"
```

参考

- ob_start()
- ob_get_length()

---

ob_get_flush

(PHP 4 >= 4.3.0, PHP 5)

ob_get_flush — 出力バッファをフラッシュし、その内容を文字列として返した後で出力バッファリングを終了する

説明

string **ob_get_flush** (void)

ob_get_flush() は、 出力バッファをフラッシュしてその内容を文字列として返した後、 出力バッファリングを終了します。

> **注意**: この関数は ob_end_flush() と似ていますが、 この関数はバッファの内容を文字列として返します。

返り値

出力バッファを返します。バッファリングが開始されていない場合は **FALSE** を返します。

例

Example#1 ob_get_flush() の例

```
<?php
// 出力バッファリングを On にします
print_r(ob_list_handlers());

// バッファをファイルに保存します
$buffer = ob_get_flush();
file_put_contents('buffer.txt', $buffer);

print_r(ob_list_handlers());
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => default output handler
)
Array
(
)
```

参考

- ob_end_clean()
- ob_end_flush()
- ob_list_handlers()

---

ob_get_length

(PHP 4 >= 4.0.2, PHP 5)

ob_get_length — 出力バッファの長さを返す

説明

int **ob_get_length** (void)

この関数は、出力バッファの内容の長さを返します。

返り値

出力バッファの内容の長さを返します。 出力のバッファリングがアクティブでない場合には、**FALSE** を返します。

例

Example#1 単純な ob_get_length() の例

```
<?php

ob_start();

echo "Hello ";

$len1 = ob_get_length();
```

```
echo "World";

$len2 = ob_get_length();

ob_end_clean();

echo $len1 . ", " . $len2;
?>
```

上の例の出力は以下となります。

```
6, 11
```

参考

- ob_start()
- ob_get_contents()

---

ob_get_level

(PHP 4 >= 4.2.0, PHP 5)

ob_get_level — 出力バッファリング機構のネストレベルを返す

説明

int **ob_get_level** (void)

出力バッファリングハンドラのネストレベルを返します。

返り値

出力バッファリングハンドラのネストレベルを返します。 バッファリングがアクティブでない場合はゼロを返します。

参考

- ob_start()
- ob_get_contents()

---

ob_get_status

(PHP 4 >= 4.2.0, PHP 5)

ob_get_status — 出力バッファのステータスを取得する

説明

array **ob_get_status** ([bool $full_status =FALSE])

ob_get_status() は、トップレベルの出力バッファの ステータス情報を返します。full_status が **TRUE** に設定されている場合は、すべてのアクティブな出力バッファの ステータス情報を返します。

パラメータ

full_status

TRUE を指定すると、すべてのアクティブな出力バッファを返します。 **FALSE** を指定した場合、あるいは省略した場合は、 トップレベルの出力バッファのみを返します。

返り値

パラメータ full_status を指定していなかったり full_status = **FALSE** としていた場合は、 以下の要素を保持する配列が返されます。

```
Array
(
     [level] => 2
     [type] => 0
     [status] => 0
     [name] => URL-Rewriter
     [del] => 1
)
```

単純な ob_get_status() の出力結果
キー:level
値:出力階層レベル
キー:type
値:PHP_OUTPUT_HANDLER_INTERNAL (0) あるいは PHP_OUTPUT_HANDLER_USER (1)
キー:status

値:PHP_OUTPUT_HANDLER_START (0)、PHP_OUTPUT_HANDLER_CONT (1) あるいは PHP_OUTPUT_HANDLER_END (2) のいずれか
キー:name
値:アクティブな出力ハンドラの名前、あるいは設定されていない場合は ' default output handler'
キー:del
値:ob_start() が設定した削除フラグ

full_status = **TRUE** を指定してコールした場合、 出力バッファごとにひとつの要素を保持する配列が返されます。 出力レベルが配列のキーとして使用され、対応する値として 各出力レベルのステータス情報を配列として保持します。

```
Array
(
     [0] => Array
         (
              [chunk_size] => 0
              [size] => 40960
              [block_size] => 10240
             [type] => 1
             [status] => 0
              [name] => default output handler
             [del] => 1
         )

     [1] => Array
         (
              [chunk_size] => 0
              [size] => 40960
              [block_size] => 10240
             [type] => 0
              [buffer_size] => 0
             [status] => 0
              [name] => URL-Rewriter
             [del] => 1
         )

)
```

完全な出力には、追加項目として以下の要素が含まれます。

完全な ob_get_status() の出力結果
キー:chunk_size
値:ob_start() で設定したチャンクの大きさ
キー:size
値:...
キー:blocksize
値:...

参考

- ob_get_level()
- ob_list_handlers()

---

ob_gzhandler

(PHP 4 >= 4.0.4, PHP 5)

ob_gzhandler — 出力バッファを gzip 圧縮するための ob_start コールバック関数

説明

```
string ob_gzhandler ( string $buffer , int $mode )
```

ob_gzhandler() は ob_start() 用のコールバック関数として使用されることを意図したもので、 圧縮されたページをサポートしている web ブラウザに対して gz エンコードされたデータを送信することを容易にします。 **ob_gzhandler()** は、実際に圧縮されたデータを送信する前にブラウザがサポートする content encoding の種類("gzip"、"deflate" またはなし)を調べ、それに基づいて 出力を返します。すべてのブラウザがサポートされています。 というのも、ブラウザは、 自分が圧縮されたページをサポートするかどうかを表す 適切なヘッダを送信することになっているからです。 圧縮されたページをブラウザがサポートしていない場合、 この関数は **FALSE** を返します。

パラメータ

buffer

mode

返り値

変更履歴

バージョン	説明
4.0.5	mode パラメータが追加されました。

例

Example#1 ob_gzhandler() の例

```
<?php

ob_start("ob_gzhandler");
```

```
?>
<html>
<body>
<p>このページは圧縮されます。</p>
</html>
<body>
```

注意

注意: **ob_gzhandler()** は、zlib 拡張モジュールを必要とします。

注意: **ob_gzhandler()** と zlib.output_compression の両方を使用することはできません。 また、 zlib.output_compression を使用すると、それは **ob_gzhandler()** よりも優先されることに注意してください。

参考

- ob_start()
- ob_end_flush()

---

ob_implicit_flush

(PHP 4, PHP 5)

ob_implicit_flush — 自動フラッシュをオンまたはオフにする

説明

void **ob_implicit_flush** ([int $flag])

ob_implicit_flush() は、 自動フラッシュをオンまたはオフに切替えます。 自動フラッシュにより、出力関数のコールが行われるたびに フラッシュ操作が行われるようになります。このため、flush() を明示的にコールする必要はなくなります。

パラメータ

flag

TRUE で自動フラッシュをオンに、**FALSE** でオフにします。 デフォルトは **TRUE** です。

返り値

値を返しません。

参考

- flush()
- ob_start()
- ob_end_flush()

---

ob_list_handlers

(PHP 4 >= 4.3.0, PHP 5)

ob_list_handlers — 使用中の出力ハンドラの一覧を取得する

説明

array **ob_list_handlers** (void)

使用中の出力ハンドラの一覧を返します。

返り値

これは、使用中の出力ハンドラを（もし存在すれば）配列で返します。もし output_buffering が 有効になっているか、あるいは ob_start() で無名関数が使用されている場合、**ob_list_handlers()** は "default output handler" を返します。

例

Example#1 ob_list_handlers() の例

```
<?php
// 出力バッファリングを On にします
print_r(ob_list_handlers());
ob_end_flush();

ob_start("ob_gzhandler");
print_r(ob_list_handlers());
ob_end_flush();

// 無名関数
ob_start(create_function('$string', 'return $string;'));
print_r(ob_list_handlers());
ob_end_flush();
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => default output handler
)

Array
(
    [0] => ob_gzhandler
)

Array
(
    [0] => default output handler
)
```

参考

- ob_end_clean()
- ob_end_flush()
- ob_get_flush()
- ob_start()

---

ob_start

(PHP 4, PHP 5)

ob_start — 出力のバッファリングを有効にする

説明

bool **ob_start** ([callback $output_callback [, int $chunk_size [, bool $erase]]])

この関数は出力のバッファリングをオンにします。 出力のバッファリングを有効にすると、 （ヘッダ以外の）スクリプトからの出力は実際には行われず、 代わりに内部バッファに保存されます。

この内部バッファの内容は、ob_get_contents() を用いて文字列変数にコピーされます。 内部バッファの内容を出力するには ob_end_flush() を使用します。 ob_end_clean() は、バッファの内容を出力せずに消去します。

警告

web サーバによっては（例: Apache）、コールバック関数からコールされた際に、 スクリプトの実行ディレクトリを変更するものがあります。 コールバック関数の内部で chdir(dirname($_SERVER['SCRIPT_FILENAME'])) などと指定することで、これをもとに戻すことが可能です。

出力バッファはスタッカブルであり、このため、他の **ob_start()** がアクティブの間に **ob_start()** をコールすることが可能です。この場合、ob_end_flush() を適切な回数コールするようにしてください。 複数の出力コールバック関数がアクティブの場合、 ネストした順番で逐次連続的に出力がフィルタ処理されます。

パラメータ

output_callback

オプションの引数 output_callback 関数を指定することが可能です。この関数は、パラメータとして文字列をとり、 文字列を返します。このコールバック関数は、 ob_end_flush() がコールされた際、 またはリクエストの最後にブラウザに出力をフラッシュする際にコールされます。 output_callback がコールされた際に、 この関数は出力バッファの内容をパラメータとして受け取ります。このコールバック関数は、 新規の出力バッファを実際に出力される結果として返す必要があり、 この結果はブラウザに送信されます。 output_callback がコール可能な関数ではない場合は **FALSE** を返します。

コールバック関数が 2 つの引数をとる場合、 2 番目のパラメータは以下のビットフィールド **PHP_OUTPUT_HANDLER_START**、**PHP_OUTPUT_HANDLER_CONT** および **PHP_OUTPUT_HANDLER_END** を含みます。

output_callback が **FALSE** を返すと、元の入力がそのままブラウザに送信されます。

output_callback パラメータに **NULL** 値を渡すと、 これをバイパスすることができます。

ob_end_clean()、 ob_end_flush()、 ob_clean()、 ob_flush() および **ob_start()** をコールバック関数の内部からコールすることはできません。 実際にコールした際の挙動は未定義です。バッファの内容を消去したい際には、 コールバック関数から ""（空文字列）を返してください。 同じく、print_r($expression, true) や highlight_file($filename, true) のような 出力バッファリング関数も、 コールバック関数の内部からコールすることはできません。

> **注意**: PHP 4.0.4 において、Web ページの圧縮をサポートする圧縮 gz エンコード されたデータの Web ブラウザへの送信を容易にするために ob_gzhandler() がサポートされています。 ob_gzhandler() は、ブラウザが受け入れる content encoding の型を調べ、それに基づいて出力を返します。

chunk_size

オプションのパラメータ chunk_size が渡された場合、 バッファの長さが chunk_size バイトを超えるたびに、 出力の後でバッファがフラッシュされます。 デフォルト値は 0 で、これはこの関数が最後に一度だけコールされることを意味します。 また、もうひとつの特別な値は 1 で、これを指定すると chunk_size が 4096 となります。

erase

オプションのパラメータ erase に **FALSE** を指定すると、 スクリプトが終了するまでバッファは削除されません (PHP 4.3.0 以降で対応)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.2	output_callback を実行することができない場合に FALSE を返すように変更されました。

例

Example#1 ユーザ定義のコールバック関数の例

```
<?php

function callback($buffer)
{
  // apples を全て oranges に置換する
  return (str_replace("apples", "oranges", $buffer));
}

ob_start("callback");

?>
<html>
<body>
<p>It's like comparing apples to oranges.</p>
</body>
</html>
<?php

ob_end_flush();

?>
```

上の例の出力は以下となります。

```
<html>
<body>
<p>It's like comparing oranges to oranges.</p>
</body>
</html>
```

参考

- ob_get_contents()
- ob_end_clean()
- ob_end_flush()
- ob_implicit_flush()
- ob_gzhandler()
- ob_iconv_handler()
- mb_output_handler()
- ob_tidyhandler()

---

output_add_rewrite_var

(PHP 4 >= 4.3.0, PHP 5)

output_add_rewrite_var — URL リライタの値を追加する

説明

bool **output_add_rewrite_var** (string $name , string $value)

この関数は、URL リライト機構に新しい名前/値の組を追加します。 名前および値は、URL (GET パラメータとして) およびフォーム (hidden フィールドとして) で追加されます。これは、session.use_trans_sid で透過的 URL リライティングが有効になっている場合に セッション ID が渡される方法と同じです。 絶対 URL (http://example.com/..) はリライトされないことに注意しましょう。

この関数の挙動は、php.ini パラメータ url_rewriter.tags によって制御されます。

注意: もし出力バッファリングが有効になっていない場合、この関数を コールすると出力バッファリングが暗黙的に開始されます。

パラメータ

name

変数名。

value

変数の値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 output_add_rewrite_var() の例

```
<?php
output_add_rewrite_var('var', 'value');

// リンク
echo '<a href="file.php">リンク</a>
<a href="http://example.com">リンク2</a>';

// フォーム
echo '<form action="script.php" method="post">
<input type="text" name="var2" />
</form>';

print_r(ob_list_handlers());
?>
```

上の例の出力は以下となります。

```
<a href="file.php?var=value">リンク</a>
<a href="http://example.com">リンク2</a>

<form action="script.php" method="post">
<input type="hidden" name="var" value="value" />
<input type="text" name="var2" />
</form>

Array
(
    [0] => URL-Rewriter
)
```

参考

- output_reset_rewrite_vars()
- ob_flush()
- ob_list_handlers()

---

output_reset_rewrite_vars

(PHP 4 >= 4.3.0, PHP 5)

output_reset_rewrite_vars — URL リライタの値をリセットする

説明

bool **output_reset_rewrite_vars** (void)

この関数は URL リライタをリセットし、 output_add_rewrite_var() 関数や セッション管理機構 (session.use_trans_sid が session_start() に設定されている場合) によって事前に設定されたリライト変数を削除します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 output_reset_rewrite_vars() の例

```
<?php
session_start();
output_add_rewrite_var('var', 'value');

echo '<a href="file.php">link</a>';
ob_flush();

output_reset_rewrite_vars();
echo '<a href="file.php">link</a>';
?>
```

上の例の出力は以下となります。

```
<a href="file.php?PHPSESSID=xxx&var=value">link</a>
<a href="file.php">link</a>
```

参考

- output_add_rewrite_var()
- ob_flush()
- ob_list_handlers()
- session_start()

目次

Ovrimos SQL 関数

導入

Ovrimos SQL サーバはクライアント/サーバ形式で トランザクション機能を持つ RDBMS で、Web に対応しており 高速なトランザクション処理が可能です。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 4.4.5 および PHP 5.1.0。

> **注意**: この拡張モジュールは Windows 環境では利用できません。

要件

Ovrimos SQL サーバの配布ファイルから入手可能な sqlcli ライブラリを インストールする必要があります。

インストール手順

PHP で Ovrimos サポートを有効にするには、パラメータ --with-ovrimos[=DIR] を configure スクリプト に指定して PHP をコンパイルします。ただし、DIR は Ovrimos の libsqlcli をインストールしたディレクトリです。

実行時設定

設定ディレクティブは定義されていません。

リソース型

例

Example#1 Ovrimos SQL サーバに接続し、システムテーブルからデータを読み込む

```
<?php
$conn = ovrimos_connect("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all($res);
        ovrimos_free_result($res);
    }
    ovrimos_close($conn);
}
?>
```

この例は、Ovrimos SQL サーバに接続するだけのものです。

---

ovrimos_close

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_close — ovrimos への接続を閉じる

説明

void ***ovrimos_close*** *(int $connection)*

Ovrimos への指定した接続 connection を閉じます。コミットしていないトランザクションはロールバックされます。

パラメータ

connection

ovrimos_connect() が返す Ovrimos 接続 ID。

返り値

値を返しません。

参考

- *ovrimos_connect()*

---

ovrimos_commit

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_commit — トランザクションをコミットする

説明

bool ***ovrimos_commit*** *(int $connection_id)*

トランザクションをコミットします。

パラメータ

connection_id

ovrimos_connect() が返す Ovrimos 接続 ID。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *ovrimos_rollback()*

---

ovrimos_connect

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_connect — 指定したデータベースに接続する

説明

int ***ovrimos_connect*** *(string $host , string $dborport , string $user , string $password)*

指定したデータベースに接続します。

パラメータ

host

ホスト名あるいは IP アドレス。

dborport

データベース名、あるいは接続するポート番号。

user

ユーザ名。

password

user のパスワード。

返り値

成功した場合に接続 ID (0 より大きい値)、 失敗した場合に 0 を返します。

例

Example#1 ovrimos_connect() の例

```
<?php
```

```
$conn = ovrimos_connect("server.example.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all($res);
        ovrimos_free_result($res);
    }
    ovrimos_close($conn);
}
?>
```

参考

- ovrimos_close()

---

ovrimos_cursor

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_cursor — カーソルの名前を返す

説明

string **ovrimos_cursor** (int $result_id)

カーソルの名前を返します。 位置を指定した update または delete を実行する際に便利です。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

返り値

名前を表す文字列、あるいはエラー時に **FALSE** を返します。

---

ovrimos_exec

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_exec — SQL 文を実行する

説明

int **ovrimos_exec** (int $connection_id , string $query)

SQL 文（クエリまたは更新）を実行し、結果 ID を返します。

パラメータ

connection_id

ovrimos_connect() が返す Ovrimos 接続 ID。

query

SQL 命令。パラメータを含めてはいけません。プリペアドステートメントの場合は ovrimos_prepare() を使用します。

返り値

結果 ID を表す文字列、あるいはエラー時に **FALSE** を返します。

参考

- ovrimos_execute()
- ovrimos_prepare()

---

ovrimos_execute

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_execute — 準備された SQL 命令を実行する

説明

bool **ovrimos_execute** (int $result_id [, array $parameters_array])

プリペアドステートメントを実行します。

パラメータ

result_id

ovrimos_prepare() が返す Ovrimos 結果 ID。

parameters_array

プリペアドステートメントにパラメータが含まれる（文の中に クエスチョンマークがある）場合に、 正しい数のパラメータを配列で渡します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ovrimos_prepare()*

---

ovrimos_fetch_into

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_fetch_into — 結果セットから行を取得する

説明

bool **ovrimos_fetch_into** *(int $result_id , array &$result_array [, string $how [, int $rownumber]])*

指定した結果セットから行を取得し、 *result_array* に格納します。

パラメータ

result_id

ovrimos_execute() あるいは *ovrimos_exec()* が返す結果 ID。

result_array

配列は参照として渡さなければなりません。 取得した結果がそこに格納されます。

how

行の取得方法を指定します。以下の文字列のいずれかひとつとなります（大文字小文字を区別しません）。

オプション	説明
Next	現在位置から後方へ。 これはデフォルト値です。
Prev	現在位置から前方へ。
First	先頭から順方向へ。
Last	終端から逆方向へ。
Absolute	先頭からの絶対位置。 *rownumber* を指定する必要があります。

rownumber

行番号。先頭は *0* です。 *how* が *Absolute* の場合にのみ必要となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 取得の例

```
<?php
$conn=ovrimos_connect("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into($res, &$row)) {
            list($table_id, $table_name) = $row;
            echo "table_id=" . $table_id . ", table_name=" . $table_name . "¥n";
            if (ovrimos_fetch_into($res, &$row)) {
                list($table_id, $table_name) = $row;
                echo "table_id=" . $table_id . ", table_name=" . $table_name . "¥n";
            } else {
                echo "Next: error¥n";
            }
        } else {
            echo "First: error¥n";
        }
        ovrimos_free_result($res);
    }
    ovrimos_close($conn);
}
?>
```

参考

- ovrimos_fetch_row()

---

ovrimos_fetch_row

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_fetch_row — 結果セットからレコードを取得する

説明

bool **ovrimos_fetch_row** (int $result_id [, int $how [, int $row_number]])

結果セットから行を取得します。カラムの値は、 別のコールで取得する必要があります。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

how

行を取得する方法を指定します。以下の文字列のいずれかです（大文字小文字は区別しません）。

オプション	説明
Next	現在位置から後方へ。 これはデフォルト値です。
Prev	現在位置から前方へ。
First	先頭から順方向へ。
Last	終端から逆方向へ。
Absolute	先頭からの絶対位置。 rownumber を指定する必要があります。

rownumber

行番号。先頭は 0 です。 how が Absolute の場合にのみ必要となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 取得の例

```
<?php
$conn = ovrimos_connect("remote.host", "8001", "admin", "password");
if ($conn != 0) {
     echo "Connection ok!";
     $res=ovrimos_exec($conn, "select table_id, table_name from sys.tables");
     if ($res != 0) {
          echo "Statement ok!";
          if (ovrimos_fetch_row($res, "First")) {
              $table_id = ovrimos_result($res, 1);
              $table_name = ovrimos_result($res, 2);
              echo "table_id=" . $table_id . ", table_name=" . $table_name . "¥n";
              if (ovrimos_fetch_row($res, "Next")) {
                  $table_id = ovrimos_result($res, "table_id");
                  $table_name = ovrimos_result($res, "table_name");
                  echo "table_id=" . $table_id . ", table_name=" . $table_name . "¥n";
              } else {
                  echo "Next: error¥n";
              }
          } else {
              echo "First: error¥n";
          }
          ovrimos_free_result($res);
     }
     ovrimos_close($conn);
}
?>
```

この例は、行を取得し、結果を出力します。

参考

- ovrimos_fetch_into()

---

ovrimos_field_len

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_field_len — 出力カラムの長さを返す

説明

int **ovrimos_field_len** (int $result_id , int $field_number)

指定した出力カラムの長さを取得します。

パラメータ

result_id

ovrimos_execute() あるいは *ovrimos_exec()* が返す結果 ID。

field_number

フィールド番号。1 から始まります。

返り値

フィールドの長さを表す整数値、あるいはエラー時に **FALSE** を返します。

参考

- *ovrimos_field_name()*
- *ovrimos_field_num()*
- *ovrimos_field_type()*

---

ovrimos_field_name

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_field_name — 出力のカラム名を返す

説明

string **ovrimos_field_name** (int $result_id , int $field_number)

指定したインデックスに対応する出力カラム名を返します。

パラメータ

result_id

ovrimos_execute() あるいは *ovrimos_exec()* が返す結果 ID。

field_number

フィールド番号。1 から始まります。

返り値

フィールド名を表す文字列、あるいはエラー時に **FALSE** を返します。

参考

- *ovrimos_field_len()*
- *ovrimos_field_num()*
- *ovrimos_field_type()*

---

ovrimos_field_num

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_field_num — 出力カラムの (1 から始まる) インデックスを返す

説明

int **ovrimos_field_num** (int $result_id , string $field_name)

指定した出力カラムの、1 からはじまるインデックスを返します。

パラメータ

result_id

ovrimos_execute() あるいは *ovrimos_exec()* が返す結果 ID。

field_name

フィールド名。

返り値

1 からはじまるインデックス、あるいはエラー時に **FALSE** を返します。

参考

- *ovrimos_field_len()*

- ovrimos_field_name()
- ovrimos_field_type()

---

ovrimos_field_type

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_field_type — 出力カラムの型を返す

説明

int **ovrimos_field_type** (int $result_id , int $field_number)

出力カラムの型を返します。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

field_number

1 からはじまるインデックス。

返り値

フィールドの型を表す整数値、あるいはエラー時に **FALSE** を返します。

参考

- ovrimos_field_len()
- ovrimos_field_name()
- ovrimos_field_num()

---

ovrimos_free_result

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_free_result — 指定した result_id を開放する

説明

bool **ovrimos_free_result** (int $result_id)

指定した結果 ID を開放します。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

返り値

TRUE を返します。

---

ovrimos_longreadlen

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_longreadlen — long データ型から取得されるバイト数を指定する

説明

bool **ovrimos_longreadlen** (int $result_id , int $length)

long データ型 (long varchar とlong varbinary) から取得するバイト数を指定します。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

length

取得するバイト数。デフォルトはゼロ。

返り値

TRUE を返します。

---

ovrimos_num_fields

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_num_fields — カラム数を返す

説明

int **ovrimos_num_fields** (int $result_id)

指定した結果 ID のカラムの数を返します。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

返り値

カラムの数を表す整数値、あるいはエラー時に *FALSE* を返します。

参考

- ovrimos_num_rows()

---

ovrimos_num_rows

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_num_rows — update 命令により変更された行の数を返す

説明

int **ovrimos_num_rows** (int $result_id)

update 操作で変更された行の数を取得します。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

返り値

行の数を表す整数値、あるいはエラー時 *FALSE* を返します。

参考

- ovrimos_num_fields()

---

ovrimos_prepare

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_prepare — SQL 命令を準備する

説明

int **ovrimos_prepare** (int $connection_id , string $query)

SQL 文を準備します。

パラメータ

connection_id

ovrimos_connect() が返す Ovrimos 接続 ID。

query

SQL 文。

返り値

成功した場合に結果 ID、エラー時に **FALSE** を返します。

例

Example#1 ovrimos_prepare() の例

```
<?php
$conn=ovrimos_connect("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";

    // Prepare the statement
    $res=ovrimos_prepare($conn, "select table_id, table_name
                                    from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        // Execute the prepared statement
        if (ovrimos_execute($res)) {
            echo "Execute ok!¥n";
            ovrimos_result_all($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result($res);
    } else {
        echo "Prepare not ok!¥n";
    }
    ovrimos_close($conn);
}
?>
```

---

ovrimos_result_all

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_result_all — 結果全体を HTML テーブルとして出力する

説明

int **ovrimos_result_all** (int $result_id [, string $format])

結果全体を HTML の表として出力します。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

format

作成される table 要素にオプションで指定する、HTML 属性。

返り値

作成された表の行数を返します。

例

これにより SQL 命令が実行され、結果が HTML テーブルとして出力されます。

Example#1 命令を準備、実行、結果を表示する

```
<?php
$conn = ovrimos_connect("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute($res, array(3))) {
            echo "Execute ok!¥n";
            ovrimos_result_all($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result($res);
    } else {
        echo "Prepare not ok!¥n";
    }
    ovrimos_close($conn);
}
?>
```

Example#2 メタ情報つきの ovrimos_result_all()

```
<?php
$conn = ovrimos_connect("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec($conn, "select table_id, table_name
                                    from sys.tables where table_id = 1");
```

```
    if ($res != 0) {
        echo "Statement ok! cursor=" . ovrimos_cursor($res) . "¥n";
        $colnb = ovrimos_num_fields($res);
        echo "Output columns=" . $colnb . "¥n";
        for ($i=1; $i <= $colnb; $i++) {
            $name = ovrimos_field_name($res, $i);
            $type = ovrimos_field_type($res, $i);
            $len = ovrimos_field_len($res, $i);
            echo "Column " . $i . " name=" . $name . " type=" . $type . " len=" . $len . "¥n";
        }
        ovrimos_result_all($res);
        ovrimos_free_result($res);
    }
    ovrimos_close($conn);
}
?>
```

---

ovrimos_result

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_result — 出力カラムを取得する

説明

string **ovrimos_result** (int $result_id , mixed $field)

field で指定した出力カラムを取得します。

パラメータ

result_id

ovrimos_execute() あるいは ovrimos_exec() が返す結果 ID。

field

フィールド名を表す文字列、あるいは 1 から始まるインデックスとなります。

返り値

カラムを表す文字列、あるいは失敗した場合に **FALSE** を返します。

---

ovrimos_rollback

(PHP 4 >= 4.0.3, PHP 5 <= 5.0.5)

ovrimos_rollback — トランザクションをロールバックする

説明

bool **ovrimos_rollback** (int $connection_id)

トランザクションをロールバックします。

パラメータ

connection_id

ovrimos_connect() が返す Ovrimos 接続 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ovrimos_commit()

目次

- ovrimos_field_name — 出力のカラム名を返す
- ovrimos_field_num — 出力カラムの（1 から始まる）インデックスを返す
- ovrimos_field_type — 出力カラムの型を返す
- ovrimos_free_result — 指定した result_id を開放する
- ovrimos_longreadlen — long データ型から取得されるバイト数を指定する
- ovrimos_num_fields — カラム数を返す
- ovrimos_num_rows — update 命令により変更された行の数を返す
- ovrimos_prepare — SQL 命令を準備する
- ovrimos_result_all — 結果全体を HTML テーブルとして出力する
- ovrimos_result — 出力カラムを取得する
- ovrimos_rollback — トランザクションをロールバックする

Paradox ファイルアクセス

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

このモジュールにより、Paradox データベースおよび プライマリインデックスファイル、blob ファイルへの読み書きが可能となります。 書き込み機能はきわめて信頼性の高いものとなっていますが、 出来上がったファイルが他のアプリケーションから読み込めない可能性もあります。 というのも、Paradox についての資料があまりないからです。 pxlib >= 0.5.0 を使用すると、パスワードを指定しなくても 暗号化されたデータベースを読み込めます。

> **注意**: このモジュールは開発途中であり、今後変更される可能性があります。 とはいえ、API に大きな変更を加えるつもりはありません。

要件

関数の基本セットを使用するには、少なくとも PHP 5.0.0 および pxlib >= 0.4.4 が必要です。いくつかの新しい関数は、 pxlib >= 0.6.0 でのみ使用可能となります。 暗号化されたデータベースの読み書きには、少なくとも pxlib >= 0.5.0 が必要です。 paradox ライブラリ (pxlib) は、 » http://pxlib.sourceforge.net で取得できます。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 »
http://pecl.php.net/package/paradox

事前に pxlib をインストールしておいてください。 rpm や debian パッケージで pxlib をインストールした場合は、 開発者向けパッケージも忘れずにインストールしてください。

実行時設定

設定ディレクティブは定義されていません。

オブジェクト指向の API

paradox 拡張モジュールは、オブジェクト指向の API も提供しています。 この中に含まれるクラスは paradox_db だけです。 そのメソッドと関数の違いは、まず名前、 そしてもちろん最初のパラメータがないことです。 次の表に、すべてのメソッドとそれに対応する関数を示します。

paradox_db クラスのメソッド

メソッド名	同等な関数
コンストラクタ	px_new()
デストラクタ	px_delete()
open_fp()	px_open_fp()
create_fp()	px_create_fp()
close()	px_close()
numrecords()	px_numrecords()
numfields()	px_numfields()
get_record()	px_get_record()
put_record()	px_put_record()
retrieve_record()	px_retrieve_record()
delete_record()	px_delete_record()
insert_record()	px_insert_record()
update_record()	px_update_record()
get_field()	px_get_field()
get_schema()	px_get_schema()
get_info()	px_get_info()
set_parameter()	px_set_parameter()
get_parameter()	px_get_parameter()
set_value()	px_set_value()
get_value()	px_get_value()
get_info()	px_get_info()
set_targetencoding()	px_set_targetencoding()
set_tablename()	px_set_tablename()
set_blob_file()	px_set_blob_file()
date2string()	px_date2string()
timestamp2string()	px_timestamp2string()

リソース型

px_new() は新しい Paradox オブジェクトを作成します。 これは、すべての Paradox 関数で必要となります。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

以下のふたつの表で、paradox 拡張モジュールで定義されている全ての定数を示します。

フィールド型の定数

名前	意味
PX_FIELD_ALPHA	固定長の文字データ
PX_FIELD_DATE	日付。0000 年 1 月 1 日からの経過日数
PX_FIELD_SHORT	Short 型の整数 (2 バイト)
PX_FIELD_LONG	Long 型の整数 (4 バイト)
PX_FIELD_CURRENCY	PX_FIELD_NUMBER と同じ
PX_FIELD_NUMBER	Double
PX_FIELD_LOGICAL	Boolean
PX_FIELD_MEMOBLOB	バイナリラージオブジェクト
PX_FIELD_BLOB	バイナリラージオブジェクト (サポートされません)
PX_FIELD_FMTMEMOBLOB	バイナリラージオブジェクト
PX_FIELD_OLE	OLE オブジェクト (基本的に blob です。サポートされません)
PX_FIELD_GRAPHIC	画像 (基本的に blob です。サポートされません)
PX_FIELD_TIME	時刻。深夜 0 時からの経過ミリ秒数
PX_FIELD_TIMESTAMP	タイムスタンプ。0000 年 1 月 1 日からの経過ミリ秒数
PX_FIELD_AUTOINC	自動インクリメントの整数 (PX_FIELD_LONG と似ています)
PX_FIELD_BCD	bcd フォーマットで保存された十進数 (サポートされません)
PX_FIELD_BYTES	255 バイトをこえないバイト配列 (サポートされません)
PX_KEYTOLOWER	すべてのフィールド名を小文字にします
PX_KEYTOUPPER	すべてのフィールド名を大文字にします

ファイル型の定数

名前	意味
PX_FILE_INDEX_DB	インデックス化されたデータベース
PX_FILE_PRIM_INDEX	プライマリインデックス
PX_FILE_NON_INDEX_DB	インデックス化されていないデータベース
PX_FILE_NON_INC_SEC_INDEX	インクリメンタルでないセカンダリインデックス
PX_FILE_SEC_INDEX	セカンダリインデックス
PX_FILE_INC_SEC_INDEX	インクリメンタルなセカンダリインデックス
PX_FILE_NON_INC_SEC_INDEX_G	インクリメンタルでないセカンダリインデックス
PX_FILE_SEC_INDEX_G	セカンダリインデックス
PX_FILE_INC_SEC_INDEX_G	インクリメンタルなセカンダリインデックス

---

px_close

(PECL paradox:1.0-1.4.1)

px_close — paradox データベースを閉じる

説明

bool ***px_close*** *(resource $pxdoc)*

paradox データベースを閉じます。この関数は、ファイルは閉じません。 この後で fclose() をコールする必要があります。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *px_open_fp()*
- *px_new() の例*

---

px_create_fp

(PECL paradox:1.0-1.4.1)

px_create_fp — 新しい paradox データベースを作成する

説明

bool **px_create_fp** (resource $pxdoc , resource $file , array $fielddesc)

新しい paradox データベースファイルを作成します。 実際のファイルは、事前に fopen() でオープンしておく必要があります。 ファイルが書き込み可能であることを確認してください。

> **注意**: この関数をコールすると「テーブル名が空である」という警告が発生しますが、 これは無視できます。単に、 px_set_parameter() を使用して後からテーブル名を設定すればよいのです。
>
> **注意**: paradox のファイルフォーマットについての資料が不足しているため、 この関数は、きわめて実験的なものです。 この関数で作成したデータベースファイルは px_open_fp() や Paradox ソフトウェアでオープンできます。 しかしそこから先の道のりは大変でしょう。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

file

fopen() が返すファイルハンドル。

fielddesc

fielddesc は、各フィールドの設定を保持する配列です。 それぞれのフィールド設定は、2 つあるいは 3 つの要素を含む配列です。 最初の要素は常に文字列で、フィールドの名前を表します。 10 文字より長くなってはいけません。2 番目の要素にはフィールドの型を指定します。 これは、フィールド型の定数 の表の中のどれかになります。文字列フィールドや bcd フィールドでは、 3 番目の要素でその要素の精度を指定します。 フィールド定義の中に blob フィールドが含まれている場合は、 確実にすべての値を格納できるだけのフィールドサイズを確保するか、 あるいは blob を保存する際に px_set_blob_file() でファイルを指定しなければなりません。さもないと、 フィールドのデータが切り詰められてしまいます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 2 つのフィールドを持つ Paradox データベースを作成する

```
<?php
if(!$pxdoc = px_new()) {
  /* エラー処理 */
}
$fp = fopen("test.db", "w+");
$fields = array(array("col1", "S"), array("col2", "I"));
if(!px_create_fp($pxdoc, $fp, $fields)) {
  /* エラー処理 */
}
px_set_parameter($pxdoc, "tablename", "testtable");
for($i=-50; $i<50; $i++) {
  $rec = array($i, -$i);
  px_put_record($pxdoc, $rec);
}
px_close($pxdoc);
px_delete($pxdoc);
fclose($fp);
?>
```

参考

- px_new()
- px_put_record()
- fopen()

---

px_date2string

(PECL paradox:1.4.0-1.4.1)

px_date2string — 日付を文字列に変換する

説明

string **px_date2string** (resource $pxdoc , int $value , string $format)

paradox ファイルに保存されている日付を、人間が理解しやすい形式に変換します。 paradox の日付は、0000 年 1 月 1 日からの経過日数で保存されています。 この関数は利便性を高めるためだけのもので、 以下の例のように数学関数やカレンダー関数で同等のことを実現できます。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

value

PX_FIELD_DATE 型の paradox データベースフィールドに格納される値。

format

date() で使用するのと同じ形式の文字列フォーマット。 この関数がサポートするプレースホルダは、date() でサポートしているもの (Y, y, m, n, d, j, L) のサブセットです。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 paradox の日付を人間が理解しやすい形式に変換する

```
<?php
$px = px_new();

/* paradox db 形式の日付データを再現します。*/
/* 0000 年 1 月 1 日から 700000 日後です。*/
$days = 700000;

/* カレンダー関数を使用して、人間が理解しやすい形式で */
/* 日付を表示します。                                 */
echo jdtogregorian($days+1721425)."¥n";
/* px_date2string() で同じように出力します。*/
echo px_date2string($px, $days, "n/d/Y")."¥n";

px_delete($px);
?>
```

上の例の出力は以下となります。

```
7/15/1917
7/15/1917
```

参考

- px_timestamp2string()
- jdtogregorian()

---

px_delete_record

(PECL paradox:1.4.0-1.4.1)

px_delete_record — paradox データベースからレコードを削除する

説明

bool **px_delete_record** (resource $pxdoc , int $num)

この関数は、データベースからレコードを削除します。 データベースファイル内の領域を開放するわけではなく、単に削除マークをつけます。 その後、新しいレコードが挿入される際にこの領域が再利用されます。

> **注意**: この関数は、pxlib >= 0.6.0 が使用されている場合にのみ有効です。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

num

レコード番号は人為的な番号で、レコードがデータベースに格納された順番を表します。 最初のレコードの番号は 0 です。

---

px_delete

(PECL paradox:1.0-1.4.1)

px_delete — paradox データベースのリソースを削除する

説明

bool **px_delete** (resource $pxdoc)

paradox ファイルのリソースを削除し、メモリを開放します。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

px_get_field

(PECL paradox:1.0-1.4.1)

px_get_field — 単一のフィールドの仕様を返す

説明

array **px_get_field** (resource $pxdoc , int $fieldno)

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

fieldno

フィールド番号。最初のフィールドが 0 番となります。 0 より小さい番号やフィールド数以上の番号を指定すると、 エラーが発生します。

返り値

fieldno 番目のデータベースフィールドの仕様を、 連想配列で返します。 name、type および size という 3 つのフィールドが含まれます。

---

px_get_info

(PECL paradox:1.0-1.4.1)

px_get_info — paradox ファイルに関する多くの情報を返す

説明

array **px_get_info** (resource $pxdoc)

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

返り値

paradox ファイルに関する多くの情報を含む連想配列を返します。 この配列は、将来的にさらに拡張される予定です。

fileversion

ファイルのバージョンを 10 倍した値。例えば 70。

tablename

ファイルに格納されているテーブルの名前。pxlib が作成したデータベースの場合は、 ファイル名から拡張子を取り除いたものになります。

numrecords

このテーブルのレコード数。

numfields

このテーブルのフィールド数。

headersize

ヘッダが使用しているバイト数。通常は 0x800。

recordsize

各レコードが使用しているバイト数。すべてのフィールドサイズの和です (バージョン 1.4.2 以降で使用可能です)。

maxtablesize

データブロックのバイト数に 0x400 をかけた値。

numdatablocks

ファイル内のデータブロック数。各データブロックには 複数のレコードが含まれます。そのレコード数は、 レコードサイズおよびデータブロックサイズ (maxtablesize) に依存します。データブロック内が完全にデータで埋め尽くされる必要はありません。

numindexfields

プライマリインデックスに使用されるフィールドの数。 このフィールドは、常にフィールド番号 1 番から始まります。

codepage

文字データのフィールドをエンコードするために使用される DOS コードページ。px_set_targetencoding() で対象のエンコーディングが指定されていない場合、これが、 px_get_record() あるいは px_retrieve_record() でレコードが関連付けられた際の 文字フィールドのエンコーディングとなります。

参考

- px_numfields()
- px_numrecords()

---

px_get_parameter

(PECL paradox:1.1.0-1.4.1)

px_get_parameter — パラメータを取得する

説明

string **px_get_parameter** (resource $pxdoc , string $name)

さまざまなパラメータを取得します。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

name

name は、以下のいずれかです。

tablename

データベースヘッダに格納されるテーブル名。

targetencoding

出力用のエンコーディング。 px_get_record() あるいは px_retrieve_record() で文字フィールドから読み込まれたデータは、targetencoding で記録されます。指定されていない場合は、 データベースファイルに保存されている形式が使用されます。

inputencoding

入力データをデータベースに保存する際のエンコーディング。 文字フィールドのデータをデータベースに保存する際には、 データをこのエンコーディングに変換することになります。

返り値

パラメータの値、あるいは失敗した場合に **FALSE** を返します。

---

px_get_record

(PECL paradox:1.0-1.4.1)

px_get_record — paradox データベースのレコードを返す

説明

array **px_get_record** (resource $pxdoc , int $num [, int $mode])

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

num

レコード番号は人為的な番号で、レコードがデータベースに格納された順番を表します。 最初のレコードの番号は 0 です。

mode

返される配列のキーを小文字または大文字に変換するために、 オプションの mode に **PX_KEYTOLOWER** あるいは **PX_KEYTOUPPER** を指定することができます。mode が渡されなかったりあるいは 0 であった場合は、 キーはフィールド名そのものとなります。 要素の値にはフィールドの値が含まれます。 NULL 値もそのまま残ります。NULL は 0.0、0 あるいは 空の文字列とはことなる値です。**PX_FIELD_TIME** 型のフィールドは、深夜 0 時から数えたミリ秒数を整数で返します。 timestamp は浮動小数点値で、ユリウス暦の開始時からの経過ミリ秒数を返します。 timestamp (**PX_FIELD_TIMESTAMP**) および date (**PX_FIELD_DATE**) は浮動小数点値で、 それぞれユリウス暦の開始時からの経過ミリ秒数、日数を返します。これらの値を文字列表現に変換するには px-timestamp2string() および px-date2string() を使用します。

返り値

paradox データベースから、num 番目のレコードを返します。 レコードは連想配列形式で返され、フィールド名がキーとなります。

参考

- px_retrieve_record()

---

px_get_schema

(PECL paradox:1.0-1.4.1)

px_get_schema — データベーススキーマを返す

説明

array **px_get_schema** (resource $pxdoc [, int $mode])

px_get_schema() は、データベーススキーマを返します。

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

mode

返される配列のキーを小文字または大文字に変換するために、 オプションの mode に **PX_KEYTOLOWER** あるいは **PX_KEYTOUPPER** を指定することができます。mode が渡されなかったりあるいは 0 であった場合は、 キーはフィールド名そのものとなります。

返り値

データベースファイルのスキーマを連想配列で返します。 キーの名前はフィールド名と等しくなります。 配列の各要素もまた連想配列となっており、2 つのフィールド type および size が含まれます。type は、 フィールド型の定数 の中のいずれかです。 size は、レコード内でこのフィールドが使用するバイト数です。 すべてのフィールドのサイズを合計したものが、 px-get-info() で得られるレコードサイズと等しくなります。

---

px_get_value

(PECL paradox:1.1.0-1.4.1)

px_get_value — 値を取得する

説明

float **px_get_value** (resource $pxdoc , string $name)

さまざまな値を取得します。

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

name

name は以下のいずれかです。

numprimkeys

プライマリキーの数。paradox データベースは、 常に最初の numprimkeys フィールドを プライマリキーとして扱います。

返り値

パラメータの値、あるいは失敗した場合に **FALSE** を返します。

---

px_insert_record

(PECL paradox:1.4.0-1.4.1)

px_insert_record — paradox データベースにレコードを挿入する

説明

int **px_insert_record** (resource $pxdoc , array $data)

データベースに新しいレコードを挿入します。新しいレコードは、 必ずデータベースの最後に格納されるとは限りません。 最初に見つかった空きスロットの位置に格納されることになります。

レコードのデータは、フィールド値の配列として渡します。 配列の要素がデータベースのフィールドに対応していなければなりません。 配列の要素数がデータベースのフィールド数より少ない場合は、 それ以降のフィールドには null が設定されます。

ほとんどのフィールドは、その型に対応する PHP の型でデータを渡します。 例えば PX_FIELD_LONG、PX_FIELD_SHORT および PX_FIELD_AUTOINC の場合は long 型、PX_FIELD_CURRENCY および PX_FIELD_NUMBER の場合は double 型が使用されます。blob 型や alpha 型のフィールドには、文字列を使用します。

PX_FIELD_TIME および PX_FIELD_DATE については long 型を指定します。 前者は深夜 0 時からの経過ミリ秒数、後者は 0000 年 1 月 1 日からの 経過日数を指定します。以下に、現在の日付および時刻を paradox のフィールドで使用できる値に変換する例を 2 種類示します。

注意: この関数は、pxlib >= 0.6.0 が使用されている場合にのみ有効です。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

data

px_retrieve_record() などが返す、 フィールド値を含む連想配列あるいは数値添字配列。

返り値

失敗した場合には **FALSE**、成功した場合にはレコード数を返します。

例

Example#1 paradox データベースの日付/時刻フィールドに、現在の日付/時刻を設定する

```
<?php
$px = px_new();
$fp = fopen("test.db", "w+");
px_create_fp($px, $fp, array(array("timestamp", "@"), array("time", "T"), array("date", "D")));

$curdate = getdate();
$jd = gregoriantojd($curdate["mon"], $curdate["mday"], $curdate["year"]);
$days = $jd - 1721425; /* 紀元前 4714 年 1 月 1 日と 0000 年 1 月 1 日の差 */
$secs = $curdate["hours"]*3600 + $curdate["minutes"]*60 + $curdate["seconds"];
px_insert_record($px, array($days*86400000.0 + $secs*1000.0, $secs*1000.0, $days));

$curtimestamp = microtime(true);
$days = (int) ($curtimestamp/86400);
$secs = $curtimestamp - ($days * 86400.0);
$days += 2440588; /* 紀元前 4714 年 1 月 1 日と 1970 年 1 月 1 日の差 */
$days -= 1721425; /* 紀元前 4714 年 1 月 1 日と 0000 年 1 月 1 日の差 */
px_insert_record($px, array($days*86400000.0 + $secs*1000.0, $secs*1000.0, $days));
for($i=0; $i<2; $i++) {
    $rec = px_retrieve_record($px, $i);
    echo px_timestamp2string($px, $rec["timestamp"], "n/d/Y H:i:s")."¥n";
    echo px_date2string($px, $rec["date"], "n/d/Y")."¥n";
}
px_close($px);
px_delete($px);
?>
```

上の例の出力は以下となります。

```
2/21/2006 21:42:30
2/21/2006
2/21/2006 20:42:30
2/21/2006
```

jdtogregorian() に渡すユリウス日は、 紀元前 4714 年 1 月 1 日を基準としており、paradox ファイルで使用している値に変換するには 1721425 日ぶん加算しなければなりません。 日数をタイムスタンプに変換するのは簡単で、 日数に 86400000.0 を掛ければミリ秒数になります。

参考

px_update_record()

---

px_new

(PECL paradox:1.0-1.4.1)

px_new — 新しい paradox オブジェクトを作成する

説明

resource **px_new** (void)

新しい paradox オブジェクトを作成します。この関数は、 その他の関数をコールする前にコールする必要があります。 **px_new()** は、ディスク上にファイルを作成するのではなく、 ただ単に paradox オブジェクトのインスタンスを作成するだけです。 オブジェクト指向のインターフェイスを使用している場合は、 この関数をコールしてはいけません。代わりに new paradox_db() を使用してください。

返り値

失敗した場合は **FALSE** を返します。

例

Example#1 Paradox データベースをオープンする

```
<?php
if(!$pxdoc = px_new()) {
  /* エラー処理 */
```

```
}
$fp = fopen("test.db", "r");
if(!px_open_fp($pxdoc, $fp)) {
  /* エラー処理 */
}
// ...
px_close($pxdoc);
px_delete($pxdoc);
fclose($fp);
?>
```

オブジェクト指向の API がお好みなら、上の例は、このようになります。

Example#2 Paradox データベースをオープンする

```
<?php
$fp = fopen("test.db", "r");
$pxdoc = new paradox_db();
if(!$pxdoc->open_fp($fp)) {
  /* エラー処理 */
}
// ...
$pxdoc->close();
fclose($fp);
?>
```

参考

- px_delete()
- px_open_fp()

---

px_numfields

(PECL paradox:1.0-1.4.1)

px_numfields — データベース内のフィールドの数を返す

説明

int **px_numfields** (resource $pxdoc)

データベースファイル内のフィールドの数を返します。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

返り値

データベースファイル内のフィールドの数を返します。この関数の返す値は、 px_get_info() が返す配列における要素 numfields の値と等しくなります。

---

px_numrecords

(PECL paradox:1.0-1.4.1)

px_numrecords — データベース内のレコードの数を返す

説明

int **px_numrecords** (resource $pxdoc)

データベースファイル内のレコードの数を取得します。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

返り値

データベースファイル内のレコードの数を返します。この関数の返す値は、 px_get_info() が返す配列における要素 numrecords の値と等しくなります。

---

px_open_fp

(PECL paradox:1.0-1.4.1)

px_open_fp — paradox データベースをオープンする

説明

bool **px_open_fp** (resource $pxdoc , resource $file)

既存の paradox データベースファイルをオープンします。事前に fopen() で実際のファイルをオープンしておく必要があります。 この関数は、プライマリインデックスファイルをオープンして、 それを paradox データベースのように扱うために使用することもできます。 この機能は、プライマリインデックスの内容を調査したい人のためにサポートされています。 データベースファイルへのアクセスを高速化するなどのために使用することはできません。

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

file

file は、 実際のデータベースファイルをパラメータに指定して fopen() をコールしたときの返り値です。レコードの更新や挿入を考えているのならば、 ファイルが書き込み可能でなければならないことに注意しましょう。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- fopen()
- The example at px_new()

---

px_put_record

(PECL paradox:1.0-1.4.1)

px_put_record — paradox データベースにレコードを保存する

説明

bool **px_put_record** (resource $pxdoc , array $record [, int $recpos])

paradox データベースにレコードを保存します。たとえ空きスロットがあったとしても、 レコードは常にデータベースの最後に追記されます。 最初に見つかった空きスロットにデータを書き込みたい場合は、 px_insert_record() を使用します。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

record

例えば px_retrieve_record() が返すような形式でのフィールド値を含む、 連想配列あるいは数値添字配列。

recpos

このオプションパラメータで、データベース内の現在の位置以降の レコード番号を指定します。この関数は、必要なだけ空のレコードを追加します。このパラメータが必要になることは、まずないでしょう。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

px_retrieve_record

(PECL paradox:1.4.0-1.4.1)

px_retrieve_record — paradox データベースのレコードを返す

説明

array **px_retrieve_record** (resource $pxdoc , int $num [, int $mode])

この関数は px_get_record() と非常によく似ていますが、 関数内でデータを取得するために使用している方法が違います。 この関数は、各フィールドの値を取得するのに pxlib の機能を使用しており、 結果としてより多くのフィールド型をサポートしています。

> **注意**: この関数は、pxlib >= 0.6.0 が使用されている場合にのみ有効です。

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

num

レコード番号は人為的な番号で、レコードがデータベースに格納された順番を表します。 最初のレコードの番号は 0 です。

mode

キーを小文字または大文字に変換するために、 オプションの mode に PX_KEYTOLOWER あるいは PX_KEYTOUPPER を指定することができます。 mode が渡されなかったりあるいは 0 であった場合は、 キーはフィールド名そのものとなります。 要素の値にはフィールドの値が含まれます。 NULL 値もそのまま残ります。NULL は 0.0、0 あるいは 空の文字列とはことなる値です。PX_FIELD_TIME 型のフィールドは、深夜 0 時から数えたミリ秒数を整数で返します。 timestamp は浮動小数点値で、ユリウス暦の開始時からの経過ミリ秒数を返します。

返り値

paradox データベースから、num 番目のレコードを返します。 レコードは連想配列形式で返され、フィールド名がキーとなります。

参考

- px_get_record()

---

px_set_blob_file

(PECL paradox:1.3.0-1.4.1)

px_set_blob_file — blob を読み込むファイル名を設定する

説明

bool ***px_set_blob_file*** *(resource $pxdoc , string $filename)*

blob の読み込みあるいは書き込みを行おうとしているファイルの名前を設定します。 この関数をコールしていない場合、もしデータがレコードの一部であるが blob ファイルに保存されていないときには、 px_get_record() あるいは px_retrieve_record() は blob フィールドのデータのみを返します。 blob データが blob フィールドのサイズに収まるほど小さい場合は、 それはレコード内に保存されます。

px_set_blob_file() *をコールせずに px_put_record()、 px_insert_record() あるいは px_update_record() をコールすると、 データベースファイルに収まらない場合にデータが切り詰められます。*

この関数を 2 度コールすると、 最初の blob ファイルを閉じて新しいほうをオープンします。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

filename

ファイルの名前。拡張子は .MB でなければなりません。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

---

px_set_parameter

(PECL paradox:1.1.0-1.4.1)

px_set_parameter — パラメータを設定する

説明

bool ***px_set_parameter*** *(resource $pxdoc , string $name , string $value)*

さまざまなパラメータを設定します。

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

name

設定したいパラメータの内容に応じて、name には以下のいずれかを指定します。

tablename

データベースヘッダに格納されるテーブル名。

targetencoding

出力用のエンコーディング。文字フィールドから読み込まれたデータは、 targetencoding で記録されます。

inputencoding

入力データをデータベースに保存する際のエンコーディング。

value

パラメータに設定する値。inputencoding および targetencoding の場合は、iconv あるいは recode が理解できるエンコーディング名 (例: iso-8859-1、utf-8、cp850) である必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- DOS コードページについては px_get_info() を参照ください。

---

px_set_tablename

(PECL paradox:1.0-1.4.1)

px_set_tablename — テーブルの名前を設定する (非推奨)

説明

void **px_set_tablename** (resource $pxdoc , string $name)

px_create_fp() で作成した、paradox データベースのテーブル名を設定します。この関数は非推奨です。代わりに px_set_parameter() を使用してください。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

tablename

テーブルの名前。明示的に設定されなかった場合は データベースファイルの名前が設定されます。

返り値

成功した場合に **NULL**、失敗した場合に **FALSE** を返します。

参考

px_set_parameter()

---

px_set_targetencoding

(PECL paradox:1.0-1.4.1)

px_set_targetencoding — 文字フィールドのエンコーディングを設定する (非推奨)

説明

bool **px_set_targetencoding** (resource $pxdoc , string $encoding)

文字フィールドからデータを取得する際のエンコーディングを設定します。 すべての文字フィールドは、この関数で設定したエンコーディングに変換されます。 エンコーディングが指定されていない場合は、データベースファイルで指定した DOS コードページでデータが返されます。 encoding には iconv あるいは recode が理解できる文字列 ID を指定することができます。 Unix システムでは、iconv -l を実行すると使用可能なエンコーディングの一覧を取得できます。

この関数は非推奨です。代わりに px_set_parameter() を使用してください。

データベースファイルに保存されている DOS コードページを知るには px_get_info() も参照ください。

パラメータ

pxdoc

px_new() が返す paradox データベースのリソース ID。

encoding

出力用のエンコーディング。文字フィールドから読み込まれたデータは targetencoding で記録されます。

返り値

エンコーディングが設定されていない場合は **FALSE** を返します。 これは、例えば未知のエンコーディングが設定されていたり、 pxlib がコード変換をサポートしていない場合にも起こります。 後者の場合は警告が発生します。

参考

px_set_parameter()

---

px_set_value

(PECL paradox:1.1.0-1.4.1)

px_set_value — 値を設定する

説明

bool **px_set_value** (resource $pxdoc , string $name , float $value)

さまざまな値を設定します。

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

name

name は以下のいずれかです。

numprimkeys

プライマリキーの数。paradox データベースは、 常に最初の numprimkeys フィールドを プライマリキーとして扱います。

value

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

px_set_parameter()

---

px_timestamp2string

(PECL paradox:1.4.0-1.4.1)

px_timestamp2string — タイムスタンプを文字列に変換する

説明

string **px_timestamp2string** (resource $pxdoc , float $value , string $format)

paradox ファイルに保存されているタイムスタンプを、 人間が理解しやすい形式に変換します。 paradox のタイムスタンプは、0000 年 1 月 1 日からの経過ミリ秒数で保存されています。 この関数は利便性を高めるためだけのもので、 以下の例のように数学関数やカレンダー関数で同等のことを実現できます。

パラメータ

pxdoc

paradox データベースのリソース ID。

value

paradox データベースフィールドに保存された、 PX_FIELD_TIME 型あるいは PX_FIELD_TIMESTAMP 型の値。

format

date() で使用するのと同じ形式の文字列フォーマット。 この関数がサポートするプレースホルダは、date() でサポートしているもの (Y, y, m, n, d, j, H, h, G, g, i, s, A, a, L) のサブセットです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 paradox のタイムスタンプを人間が理解しやすい形式に変換する

```
<?php
$px = px_new();

/* paradox db 形式の日付データを再現します。*/
/* 0000 年 1 月 1 日から 700000 日後です。*/
$days = 700000;

/* カレンダー関数を使用して、人間が理解しやすい形式で */
/* 日付を表示します。                                  */
echo jdtogregorian($days+1721425)."¥n";

/* paradox データベースのタイムスタンプに変換します。          */
/* タイムスタンプは 0000 年 1 月 1 日からの経過ミリ秒数です。*/
$stamp = $days * 86400.0 * 1000.0;
/* 一時間足します */
$stamp += 3600000.0;
/* これは '7/15/1917 01:00:00' を出力します。*/
echo px_timestamp2string($px, $stamp, "n/d/Y H:i:s")."¥n";

px_delete($px);
?>
```

上の例の出力は以下となります。

```
7/15/1917
7/15/1917  01:00:00
```

jdtogregorian() に渡すユリウス日は、 紀元前 4714 年 1 月 1 日を基準としており、paradox ファイルで使用している値に変換するには 1721425 日ぶん加算しなければなりません。 日数をタイムスタンプに変換するのは簡単で、 日数に 86400000.0 を掛ければミリ秒数になります。

参考

- px_date2string()
- jdtogregorian()

---

px_update_record

(PECL paradox:1.4.0-1.4.1)

px_update_record — paradox データベースのレコードを更新する

説明

bool **px_update_record** (resource $pxdoc , array $data , int $num)

データベース内のレコードを更新します。レコード番号は 0 から始まります。

レコードのデータは、フィールド値の配列として渡します。 配列の要素がデータベースのフィールドに対応していなければなりません。 配列の要素数がデータベースのフィールド数より少ない場合は、 それ以降のフィールドには null が設定されます。

注意: この関数は、pxlib >= 0.6.0 が使用されている場合にのみ有効です。

パラメータ

pxdoc

px_new() が返す、 paradox データベースのリソース ID。

data

px_retrieve_record() が返すフィールド値を含む連想配列。

num

レコード番号は人為的な番号で、レコードがデータベースに格納された順番を表します。 最初のレコードの番号は 0 です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

px_insert_record()

目次

- px_set_tablename — テーブルの名前を設定する（非推奨）
- px_set_targetencoding — 文字フィールドのエンコーディングを設定する（非推奨）
- px_set_value — 値を設定する
- px_timestamp2string — タイムスタンプを文字列に変換する
- px_update_record — paradox データベースのレコードを更新する

Parsekit 関数

導入

これらの関数によって、PHP スクリプトをコンパイルした opcode を実行時に解析することができます。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/parsekit.

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

PARSEKIT_QUIET (int)
完全な詳細を返しますが、不必要な NULL エントリは返しません。
PARSEKIT_SIMPLE (int)
省略した opcode 記法で返します。
PARSEKIT_EXTENDED_VALUE (int)
Opnode フラグ
PARSEKIT_RESULT_CONST (int)
Opnode フラグ
PARSEKIT_RESULT_EA_TYPE (int)
Opnode フラグ
PARSEKIT_RESULT_JMP_ADDR (int)
Opnode フラグ
PARSEKIT_RESULT_OPARRAY (int)
Opnode フラグ
PARSEKIT_RESULT_OPLINE (int)
Opnode フラグ
PARSEKIT_RESULT_VAR (int)
Opnode フラグ
PARSEKIT_USAGE_UNKNOWN (int)
Opnode フラグ
PARSEKIT_ZEND_INTERNAL_CLASS (int)
Class 型
PARSEKIT_ZEND_USER_CLASS (int)
Class 型
PARSEKIT_ZEND_EVAL_CODE (int)
Function 型
PARSEKIT_ZEND_INTERNAL_FUNCTION (int)
Function 型
PARSEKIT_ZEND_OVERLOADED_FUNCTION (int)
Function 型
PARSEKIT_ZEND_OVERLOADED_FUNCTION_TEMPORARY (int) PHP >= 5.0.0
Function 型
PARSEKIT_ZEND_USER_FUNCTION (int)
Function 型
PARSEKIT_IS_CONST (int)
Node 型
PARSEKIT_IS_TMP_VAR (int)
Node 型
PARSEKIT_IS_UNUSED (int)
Node 型
PARSEKIT_IS_VAR (int)
Node 型
PARSEKIT_ZEND_ADD (int)
Opcode
PARSEKIT_ZEND_ADD_ARRAY_ELEMENT (int)
Opcode
PARSEKIT_ZEND_ADD_CHAR (int)
Opcode
PARSEKIT_ZEND_ADD_INTERFACE (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_ADD_STRING (int)
Opcode
PARSEKIT_ZEND_ADD_VAR (int)

Opcode
PARSEKIT_ZEND_ASSIGN (int)
Opcode
PARSEKIT_ZEND_ASSIGN_ADD (int)
Opcode
PARSEKIT_ZEND_ASSIGN_BW_AND (int)
Opcode
PARSEKIT_ZEND_ASSIGN_BW_OR (int)
Opcode
PARSEKIT_ZEND_ASSIGN_BW_XOR (int)
Opcode
PARSEKIT_ZEND_ASSIGN_CONCAT (int)
Opcode
PARSEKIT_ZEND_ASSIGN_DIM (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_ASSIGN_DIV (int)
Opcode
PARSEKIT_ZEND_ASSIGN_MOD (int)
Opcode
PARSEKIT_ZEND_ASSIGN_MUL (int)
Opcode
PARSEKIT_ZEND_ASSIGN_OBJ (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_ASSIGN_REF (int)
Opcode
PARSEKIT_ZEND_ASSIGN_SL (int)
Opcode
PARSEKIT_ZEND_ASSIGN_SR (int)
Opcode
PARSEKIT_ZEND_ASSIGN_SUB (int)
Opcode
PARSEKIT_ZEND_BEGIN_SILENCE (int)
Opcode
PARSEKIT_ZEND_BOOL (int)
Opcode
PARSEKIT_ZEND_BOOL_NOT (int)
Opcode
PARSEKIT_ZEND_BOOL_XOR (int)
Opcode
PARSEKIT_ZEND_BRK (int)
Opcode
PARSEKIT_ZEND_BW_AND (int)
Opcode
PARSEKIT_ZEND_BW_NOT (int)
Opcode
PARSEKIT_ZEND_BW_OR (int)
Opcode
PARSEKIT_ZEND_BW_XOR (int)
Opcode
PARSEKIT_ZEND_CASE (int)
Opcode
PARSEKIT_ZEND_CAST (int)
Opcode
PARSEKIT_ZEND_CATCH (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_CLONE (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_CONCAT (int)
Opcode
PARSEKIT_ZEND_CONT (int)
Opcode
PARSEKIT_ZEND_DECLARE_CLASS (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_DECLARE_FUNCTION (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_DECLARE_INHERITED_CLASS (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_DIV (int)
Opcode
PARSEKIT_ZEND_DO_FCALL (int)
Opcode
PARSEKIT_ZEND_DO_FCALL_BY_NAME (int)
Opcode
PARSEKIT_ZEND_ECHO (int)
Opcode
PARSEKIT_ZEND_END_SILENCE (int)
Opcode
PARSEKIT_ZEND_EXIT (int)
Opcode
PARSEKIT_ZEND_EXT_FCALL_BEGIN (int)
Opcode
PARSEKIT_ZEND_EXT_FCALL_END (int)
Opcode
PARSEKIT_ZEND_EXT_NOP (int)
Opcode
PARSEKIT_ZEND_EXT_STMT (int)
Opcode
PARSEKIT_ZEND_FETCH_CLASS (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_FETCH_CONSTANT (int)
Opcode
PARSEKIT_ZEND_FETCH_DIM_FUNC_ARG (int)
Opcode
PARSEKIT_ZEND_FETCH_DIM_IS (int)
Opcode
PARSEKIT_ZEND_FETCH_DIM_R (int)
Opcode
PARSEKIT_ZEND_FETCH_DIM_RW (int)
Opcode
PARSEKIT_ZEND_FETCH_DIM_TMP_VAR (int)
Opcode

PARSEKIT_ZEND_FETCH_DIM_UNSET (int)
Opcode
PARSEKIT_ZEND_FETCH_DIM_W (int)
Opcode
PARSEKIT_ZEND_FETCH_FUNC_ARG (int)
Opcode
PARSEKIT_ZEND_FETCH_IS (int)
Opcode
PARSEKIT_ZEND_FETCH_OBJ_FUNC_ARG (int)
Opcode
PARSEKIT_ZEND_FETCH_OBJ_IS (int)
Opcode
PARSEKIT_ZEND_FETCH_OBJ_R (int)
Opcode
PARSEKIT_ZEND_FETCH_OBJ_RW (int)
Opcode
PARSEKIT_ZEND_FETCH_OBJ_UNSET (int)
Opcode
PARSEKIT_ZEND_FETCH_OBJ_W (int)
Opcode
PARSEKIT_ZEND_FETCH_R (int)
Opcode
PARSEKIT_ZEND_FETCH_RW (int)
Opcode
PARSEKIT_ZEND_FETCH_UNSET (int)
Opcode
PARSEKIT_ZEND_FETCH_W (int)
Opcode
PARSEKIT_ZEND_FE_FETCH (int)
Opcode
PARSEKIT_ZEND_FE_RESET (int)
Opcode
PARSEKIT_ZEND_FREE (int)
Opcode
PARSEKIT_ZEND_HANDLE_EXCEPTION (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_IMPORT_CLASS (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_IMPORT_CONST (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_IMPORT_FUNCTION (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_INCLUDE_OR_EVAL (int)
Opcode
PARSEKIT_ZEND_INIT_ARRAY (int)
Opcode
PARSEKIT_ZEND_INIT_CTOR_CALL (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_INIT_FCALL_BY_NAME (int)
Opcode
PARSEKIT_ZEND_INIT_METHOD_CALL (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_INIT_STATIC_METHOD_CALL (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_INIT_STRING (int)
Opcode
PARSEKIT_ZEND_INSTANCEOF (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_ISSET_ISEMPTY (int) *PHP < 5.0.0*
Opcode
PARSEKIT_ZEND_ISSET_ISEMPTY_DIM_OBJ (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_ISSET_ISEMPTY_PROP_OBJ (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_ISSET_ISEMPTY_VAR (int) *PHP >= 5.0.0*
Opcode
PARSEKIT_ZEND_IS_EQUAL (int)
Opcode
PARSEKIT_ZEND_IS_IDENTICAL (int)
Opcode
PARSEKIT_ZEND_IS_NOT_EQUAL (int)
Opcode
PARSEKIT_ZEND_IS_NOT_IDENTICAL (int)
Opcode
PARSEKIT_ZEND_IS_SMALLER (int)
Opcode
PARSEKIT_ZEND_IS_SMALLER_OR_EQUAL (int)
Opcode
PARSEKIT_ZEND_JMP (int)
Opcode
PARSEKIT_ZEND_JMPNZ (int)
Opcode
PARSEKIT_ZEND_JMPNZ_EX (int)
Opcode
PARSEKIT_ZEND_JMPZ (int)
Opcode
PARSEKIT_ZEND_JMPZNZ (int)
Opcode
PARSEKIT_ZEND_JMPZ_EX (int)
Opcode
PARSEKIT_ZEND_JMP_NO_CTOR (int)
Opcode
PARSEKIT_ZEND_MOD (int)
Opcode
PARSEKIT_ZEND_MUL (int)
Opcode
PARSEKIT_ZEND_NEW (int)
Opcode
PARSEKIT_ZEND_NOP (int)
Opcode
PARSEKIT_ZEND_OP_DATA (int) *PHP >= 5.0.0*

Opcode
PARSEKIT_ZEND_POST_DEC (int)
Opcode
PARSEKIT_ZEND_POST_DEC_OBJ (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_POST_INC (int)
Opcode
PARSEKIT_ZEND_POST_INC_OBJ (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_PRE_DEC (int)
Opcode
PARSEKIT_ZEND_PRE_DEC_OBJ (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_PRE_INC (int)
Opcode
PARSEKIT_ZEND_PRE_INC_OBJ (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_PRINT (int)
Opcode
PARSEKIT_ZEND_QM_ASSIGN (int)
Opcode
PARSEKIT_ZEND_RAISE_ABSTRACT_ERROR (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_RECV (int)
Opcode
PARSEKIT_ZEND_RECV_INIT (int)
Opcode
PARSEKIT_ZEND_RETURN (int)
Opcode
PARSEKIT_ZEND_SEND_REF (int)
Opcode
PARSEKIT_ZEND_SEND_VAL (int)
Opcode
PARSEKIT_ZEND_SEND_VAR (int)
Opcode
PARSEKIT_ZEND_SEND_VAR_NO_REF (int)
Opcode
PARSEKIT_ZEND_SL (int)
Opcode
PARSEKIT_ZEND_SR (int)
Opcode
PARSEKIT_ZEND_SUB (int)
Opcode
PARSEKIT_ZEND_SWITCH_FREE (int)
Opcode
PARSEKIT_ZEND_THROW (int) PHP >= 5.0.0
Opcode
PARSEKIT_ZEND_TICKS (int)
Opcode
PARSEKIT_ZEND_UNSET_DIM_OBJ (int)
Opcode
PARSEKIT_ZEND_UNSET_VAR (int)
Opcode
PARSEKIT_ZEND_VERIFY_ABSTRACT_CLASS (int) PHP >= 5.0.0
Opcode

---

parsekit_compile_file

(PECL parsekit:0.2-1.2)

parsekit_compile_file — PHP コードの文字列をコンパイルし、結果を op コードの配列で返す

説明

array **parsekit_compile_file** (string $filename [, array &$errors [, int $options]])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

filename

コンパイルするファイル名を含む文字列。 include() の引数と同じ形式です。

errors

コンパイル時に発生したエラー（致命的なエラーを含む）の二次元配列。参照で返されます。

options

PARSEKIT_QUIET あるいは **PARSEKIT_SIMPLE** のいずれかです。 返される出力の冗長性を設定します。

返り値

複雑な形式の配列を返します。詳細は以下に示します。

例

Example#1 parsekit_compile_file() の例

```
<?php
var_dump(parsekit_compile_file('hello_world.php', $errors, PARSEKIT_SIMPLE));
?>
```

上の例の出力は以下となります。

```
array(5) {
  [0]=>
  string(37) "ZEND_ECHO UNUSED 'Hello World' UNUSED"
  [1]=>
  string(30) "ZEND_RETURN UNUSED NULL UNUSED"
  [2]=>
  string(42) "ZEND_HANDLE_EXCEPTION UNUSED UNUSED UNUSED"
  ["function_table"]=>
  NULL
  ["class_table"]=>
  NULL
}
```

参考

- parsekit_compile_string()

---

parsekit_compile_string

(PECL parsekit:0.2-1.2)

parsekit_compile_string — PHP コードの文字列をコンパイルし、結果を op コードの配列で返す

説明

array **parsekit_compile_string** (string $phpcode [, array &$errors [, int $options]])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

phpcode

php コードを含む文字列。eval() の引数と同じ形式です。

errors

コンパイル時に発生したエラー（致命的なエラーを含む）の二次元配列。参照で返されます。

options

PARSEKIT_QUIET あるいは **PARSEKIT_SIMPLE** のいずれかです。 返される出力の冗長性を設定します。

返り値

複雑な形式の配列を返します。詳細は以下に示します。

例

Example#1 parsekit_compile_string() の例

```
<?php
  $ops = parsekit_compile_string('
echo "Foo¥n";
', $errors, PARSEKIT_QUIET);

  var_dump($ops);
?>
```

上の例の出力は以下となります。

```
array(20) {
  ["type"]=>
  int(4)
  ["type_name"]=>
  string(14) "ZEND_EVAL_CODE"
  ["fn_flags"]=>
  int(0)
  ["num_args"]=>
  int(0)
  ["required_num_args"]=>
  int(0)
  ["pass_rest_by_reference"]=>
  bool(false)
  ["uses_this"]=>
  bool(false)
  ["line_start"]=>
  int(0)
  ["line_end"]=>
  int(0)
  ["return_reference"]=>
  bool(false)
  ["refcount"]=>
  int(1)
  ["last"]=>
```

```
   int(3)
   ["size"]=>
   int(3)
   ["T"]=>
   int(0)
    ["last_brk_cont"]=>
   int(0)
    ["current_brk_cont"]=>
   int(-1)
    ["backpatch_count"]=>
   int(0)
    ["done_pass_two"]=>
   bool(true)
    ["filename"]=>
    string(17)  "Parsekit  Compiler"
    ["opcodes"]=>
   array(3)  {
      [8594800]=>
     array(5)  {
        ["opcode"]=>
       int(40)
        ["opcode_name"]=>
        string(9)  "ZEND_ECHO"
        ["flags"]=>
       int(768)
        ["op1"]=>
       array(3)  {
          ["type"]=>
         int(1)
          ["type_name"]=>
          string(8)  "IS_CONST"
          ["constant"]=>
          &string(4)  "Foo
"
       }
        ["lineno"]=>
       int(2)
     }
      ["859484C"]=>
     array(6)  {
        ["opcode"]=>
       int(62)
        ["opcode_name"]=>
        string(11)  "ZEND_RETURN"
        ["flags"]=>
        int(16777984)
        ["op1"]=>
       array(3)  {
          ["type"]=>
         int(1)
          ["type_name"]=>
          string(8)  "IS_CONST"
          ["constant"]=>
         &NULL
       }
        ["extended_value"]=>
       int(0)
        ["lineno"]=>
       int(3)
     }
      [8594898]=>
     array(4)  {
        ["opcode"]=>
       int(149)
        ["opcode_name"]=>
         string(21)  "ZEND_HANDLE_EXCEPTION"
        ["flags"]=>
       int(0)
        ["lineno"]=>
       int(3)
     }
   }
}
```

参考

- parsekit_compile_file()

---

parsekit_func_arginfo

(PECL parsekit:0.3-1.2)

parsekit_func_arginfo — 関数の引数に関する情報を返す

説明

array **parsekit_func_arginfo** (mixed $function)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

function

関数を表す文字列、あるいはクラス/メソッドを表す配列。

返り値

引数の情報を含む配列を返します。

例

Example#1 parsekit_func_arginfo() の例

```
<?php
function foo($bar, stdClass $baz, &$bomb, $bling = false) {
}

var_dump(parsekit_func_arginfo('foo'));
?>
```

上の例の出力は以下となります。

```
array(4) {
  [0]=>
  array(3) {
    ["name"]=>
    string(3) "bar"
    ["allow_null"]=>
    bool(true)
    ["pass_by_reference"]=>
    bool(false)
  }
  [1]=>
  array(4) {
    ["name"]=>
    string(3) "baz"
    ["class_name"]=>
    string(8) "stdClass"
    ["allow_null"]=>
    bool(false)
    ["pass_by_reference"]=>
    bool(false)
  }
  [2]=>
  array(3) {
    ["name"]=>
    string(4) "bomb"
    ["allow_null"]=>
    bool(true)
    ["pass_by_reference"]=>
    bool(true)
  }
  [3]=>
  array(3) {
    ["name"]=>
    string(5) "bling"
    ["allow_null"]=>
    bool(true)
    ["pass_by_reference"]=>
    bool(false)
  }
}
```

目次

プロセス制御関数

導入

PHP がサポートするプロセス制御関数は、Unix 形式のプロセス生成、 プロセス実行、シグナル処理、プロセス終了機能を実装しています。 プロセス制御は Web サーバ環境で有効にするべきではなく、プロセス 制御関数のどれかが Web サーバ環境で使用された場合には、予期しない結果を 生じる可能性があります。

この文書は、プロセス制御関数の一般的な使用法を説明しようとするものです。 Unix のプロセス制御に関する詳細な情報については、fork(2), waitpid(2) および signal(2) のようなシステムのドキュメントや、 W. Richard Stevens による Advanced Programming in the UNIX Environment (Addison-Wesley) のような優れた参考書を読まれることを推奨します。

現在 PCNTL はシグナルハンドルコールバックの仕組みとして ticks を 使用しており、これは以前の仕組みよりずっと高速です。この変更は "user ticks" を使用するのと同じことです。**declare()** を使用して、プログラム中でコールバックの発生を許可する場所を指定する 必要があります。これにより、非同期のイベントを処理する際の オーバーヘッドを最小限に抑えることが可能となります。以前は、pcntl を 有効にして PHP をコンパイルすると、pcntl の使用の有無にかかわらず 常にこのオーバーヘッドの被害を被っていたのです。

PHP 4.3.0 より前のバージョンで pcntl を使用していたすべてのスクリプトに ついて、1 点だけ修正する必要があります。それは、コールバックを許可 したい場所に declare() を使用するか、あるいは declare() の新しいグローバル書式を使用して スクリプト全体で ticks を有効にすることです。

注意: この拡張モジュールは Windows 環境では利用できません。

要件

外部ライブラリを必要としません。

インストール手順

PHPがサポートするプロセス制御機能は、デフォルトでは有効となってい ません。プロセス制御機能を有効にするには、 configure のオプションに --enable-pcntl を付け、CGI 版あるいは CLI 版の PHP をコンパイルする必要があります。

注意: 現在、このモジュールは非 Unix 環境(Windows)では動作しません。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下のシグナルのリストがプロセス制御関数でサポートされています。 これらのシグナルのデフォルト動作の詳細については、 signal(7) のマニュアルを参照ください。

WNOHANG (integer)
WUNTRACED (integer)
SIG_IGN (integer)
SIG_DFL (integer)
SIG_ERR (integer)
SIGHUP (integer)
SIGINT (integer)
SIGQUIT (integer)
SIGILL (integer)
SIGTRAP (integer)
SIGABRT (integer)
SIGIOT (integer)
SIGBUS (integer)
SIGFPE (integer)
SIGKILL (integer)
SIGUSR1 (integer)
SIGSEGV (integer)
SIGUSR2 (integer)
SIGPIPE (integer)
SIGALRM (integer)
SIGTERM (integer)
SIGSTKFLT (integer)
SIGCLD (integer)
SIGCHLD (integer)
SIGCONT (integer)
SIGSTOP (integer)
SIGTSTP (integer)
SIGTTIN (integer)
SIGTTOU (integer)
SIGURG (integer)
SIGXCPU (integer)
SIGXFSZ (integer)
SIGVTALRM (integer)
SIGPROF (integer)
SIGWINCH (integer)
SIGPOLL (integer)
SIGIO (integer)
SIGPWR (integer)
SIGSYS (integer)
SIGBABY (integer)

例

この例は、シグナルハンドラを有するデーモンプロセスをフォークします。

Example#1 プロセス制御の例

```
<?php
declare(ticks=1);

$pid = pcntl_fork();
if ($pid == -1) {
     die("fork できません");
} else if ($pid) {
     exit(); // 親プロセスの場合
} else {
     // 子プロセスの場合
}

// 制御側の端末からデタッチ
if (posix_setsid() == -1) {
     die("could not detach from terminal");
}
```

```
// シグナルハンドラを設定
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");

// 無限ループでタスク実行
while (1) {

    // 何か面白いことをここで行う

}

function sig_handler($signo)
{

      switch ($signo) {
          case SIGTERM:
             // 終了タスクを処理
              exit;
               break;
          case SIGHUP:
             // 再起動タスクを処理
               break;
           default:
             // その他の全てのシグナルを処理
      }

}

?>
```

参考

POSIX 関数の節を参照することも 有用でしょう。

---

pcntl_alarm

(PHP 4 >= 4.3.0, PHP 5)

pcntl_alarm — シグナルを送信するアラームを設定する

説明

int **pcntl_alarm** (int $seconds)

プロセスに対して、 指定した秒数後に SIGALRM シグナルを送信するタイマーを作成します。 **pcntl_alarm()** をコールすると、 それまでに設定されていたアラームはすべて取り消されます。

パラメータ

seconds

待機する秒数。seconds がゼロの場合は、 新しいアラームは作成されません。

返り値

それまでに予定されていたアラームの予定時刻までの秒数を返します。 事前に予定されていたアラームがなかった場合には 0 を返します。

---

pcntl_exec

(PHP 4 >= 4.2.0, PHP 5)

pcntl_exec — 現在のプロセス空間で指定したプログラムを実行する

説明

void **pcntl_exec** (string $path [, array $args [, array $envs]])

指定した引数でプログラムを実行します。

パラメータ

path

path は、バイナリ実行ファイルへのパスか、 あるいは有効な実行ファイルを指す shebang (例: #!/usr/local/bin/perl) が一行目に存在するスクリプトへのパスである必要があります。 詳細な情報については、システムの man ページで execve(2) を参照ください。

args

args は、プログラムに渡す引数文字列の配列です。

envs

envs は、プログラムに渡す環境変数の配列です。 この配列は name => value のような形式で、key が環境変数名・value が その値となります。

返り値

エラー時に **FALSE** を返し、成功時には何も返しません。

---

pcntl_fork

(PHP 4 >= 4.0.7, PHP 5)

pcntl_fork — 現在実行中のプロセスをフォークする

説明

int **pcntl_fork** (void)

pcntl_fork() 関数は、親プロセスとその PID および PPID のみが異なる子プロセスを生成します。システム上でのフォークの 動作の具体的な詳細については、実行するシステムの fork(2) の マニュアルを参照ください。

返り値

成功時に、子プロセスの PID が親プロセスの実行スレッドに返され、 子プロセスの実行スレッドには 0 が返されます。失敗した場合、親プロセスのコンテキストに -1 が返され、子プロセスは生成されずに、PHP のエラーが 出力されます。

例

Example#1 pcntl_fork() の例

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die('fork できません');
} else if ($pid) {
     // 親プロセスの場合
     pcntl_wait($status); //  ゾンビプロセスから守る
} else {
     // 子プロセスの場合
}

?>
```

参考

- pcntl_waitpid()
- pcntl_signal()

---

pcntl_getpriority

(PHP 5)

pcntl_getpriority — プロセスの優先度を取得する

説明

int **pcntl_getpriority** ([int $pid [, int $process_identifier]])

pcntl_getpriority() は、pid の優先度を設定します。システムの型やカーネルの バージョンによって優先度の扱いは違うので、詳細についてはシステムの getpriority(2) の man ページを参照ください。

パラメータ

pid

指定しなかった場合は、現在のプロセスの PID を使用します。

process_identifier

PRIO_PGRP、**PRIO_USER** あるいは **PRIO_PROCESS** のいずれか。

返り値

pcntl_getpriority() はプロセスの優先度を返します。 エラー時には **FALSE** を返します。数字が小さいほど、優先順位は上となります。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

参考

- pcntl_setpriority()

---

pcntl_setpriority

(PHP 5)

pcntl_setpriority — プロセスの優先度を変更する

説明

bool ***pcntl_setpriority*** *(int $priority [, int $pid [, int $process_identifier]])*

pcntl_setpriority() *は、 pid の優先度を設定します。*

パラメータ

priority

priority は一般的には -20 から 20 までの値です。 デフォルトの優先度は 0 で、数字が小さいほど 優先順位が上となります。システムの型やカーネルの バージョンによって優先度の扱いは違うので、詳細についてはシステムの setpriority(2) の man ページを参照ください。

pid

指定しない場合は、現在のプロセスの PID を使用します。

process_identifier

PRIO_PGRP、***PRIO_USER*** *あるいは* ***PRIO_PROCESS*** *のいずれかです。*

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *pcntl_getpriority()*

---

pcntl_signal

(PHP 4 >= 4.0.7, PHP 5)

pcntl_signal — シグナルハンドラを設定する

説明

bool ***pcntl_signal*** *(int $signo , callback $handler [, bool $restart_syscalls])*

pcntl_signal() *関数は、signo が指すシグナルに関するハンドラを設定します。*

パラメータ

signo

シグナル番号。

handler

*ユーザ定義関数の名前、あるいは次のふたつのグローバル定数 **SIG_IGN** または **SIG_DFL** のうちのいずれかとなります。*

> ***注意**: オブジェクトのメソッドをハンドラとして指定した場合には、 そのハンドラを別のものに変えたりスクリプトが終了したりするまでは オブジェクトの参照カウントが増加しないことに注意しましょう。*

restart_syscalls

*再起動のシステムコールに対応するかどうかを設定します。デフォルトは **TRUE** です。*

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

変更履歴

バージョン	説明
4.3.0	パラメータ *restart_syscalls* が追加されました。
4.3.0	オブジェクトのメソッドをコールバックとして使用できるようになりました。
4.3.0	PHP 4.3.0 以降、PCNTL はシグナルハンドルコールバックの仕組みとして ticks を使用しており、これは以前の仕組みよりずっと高速です。この変更は "user ticks" を使用するのと同じことです。上の例で示したように、 declare() を使用して プログラム中でコールバックの発生を許可する場所を指定し、シグナルハンドラが 正しく動作するようにする必要があります。

例

Example#1 pcntl_signal() の例

```
<?php
// PHP 4.3.0 以降では tick を使用しなければなりません
declare(ticks = 1);

// シグナルハンドラ関数
function sig_handler($signo)
{
```

```
      switch ($signo) {
          case SIGTERM:
             // シャットダウンの処理
              exit;
               break;
          case SIGHUP:
             // 再起動の処理
              break;
          case SIGUSR1:
              echo "SIGUSR1 を受け取りました...\n";
               break;
           default:
             // それ以外のシグナルの処理
      }

}

echo "シグナルハンドラを設定します...\n";

// シグナルハンドラを設定します
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP,  "sig_handler");
pcntl_signal(SIGUSR1, "sig_handler");

// あるいは PHP 4.3.0 以降ならオブジェクトも指定できます
//  pcntl_signal(SIGUSR1,  array($obj,  "do_something");

echo"自分自身に SIGTERM シグナルを送信します...\n";

// SIGUSR1 をカレントのプロセス ID に送信します
posix_kill(posix_getpid(), SIGUSR1);

echo "終了\n"

?>
```

参考

- pcntl_fork()
- pcntl_waitpid()

---

pcntl_wait

(PHP 5)

pcntl_wait — 待つかフォークした子プロセスのステータスを返す

説明

```
int pcntl_wait ( int &$status [, int $options ] )
```

この関数は、子プロセスが終了する・ カレントのプロセスを終了させるシグナルが送信される・シグナル処理関数を コールするシグナルが送信されるのいずれかが発生するまでカレントのプロセスの実行を中断します。 子プロセスが、 コール時に 既に終了している場合("ゾンビ"プロセスと呼ばれます)、この関数は 直ちに処理を返します。子プロセスにより使用される全てのシステム リソースは、解放されます。waitpid のシステムでの動作に関する詳細は、 システムの wait(2) についての man ページを参照ください。

注意: この関数は、pid に -1 を指定し、options を何も設定せずに pcntl_waitpid() をコールするのと等価です。

パラメータ

status

pcntl_wait() は、パラメータ status の中にステータス情報を保存します。 このステータスは、次の関数を用いて評価可能です。 pcntl_wifexited()、 pcntl_wifstopped()、 pcntl_wifsignaled()、 pcntl_wexitstatus()、 pcntl_wtermsig() および pcntl_wstopsig() 。

options

システム上で wait3 が使用可能な場合 (ほとんどの BSD 系システムが 該当します)、オプションのパラメータ options を使用可能です。このパラメータが指定されない場合、wait はシステムコールに 対して使用されます。wait3 が使用できない場合、options に値を設定しても何の影響も及ぼしません。 options の値は、次の 2 つのグローバル定数の ゼロまたはそれ以上の論理和です。

options のとりうる値

WNOHANG	子プロセスが終了していない場合に直ちに処理を返します。
WUNTRACED	停止した子プロセスの場合に処理を返します。 そして、ステータスは報告されません。

返り値

pcntl_wait() は、終了した子プロセスの プロセス ID を返します。エラーの場合は -1、(wait3 が使用可能なシステムで) WNOHANG が使用され、子プロセスが利用できない場合に 0 を返します。

参考

- pcntl_fork()
- pcntl_signal()
- pcntl_wifexited()
- pcntl_wifstopped()
- pcntl_wifsignaled()

- pcntl_wexitstatus()
- pcntl_wtermsig()
- pcntl_wstopsig()
- pcntl_waitpid()

---

pcntl_waitpid

(PHP 4 >= 4.0.7, PHP 5)

pcntl_waitpid — 待つかフォークした子プロセスのステータスを返す

説明

int **pcntl_waitpid** (int $pid , int &$status [, int $options])

引数 pid で指定した子プロセスが終了する・ 現在のプロセスを終了させるシグナルが送信される・シグナル処理関数を コールするシグナルが送信される のいずれかが発生するまで、現在のプロセスの実行を中断します。

pid でリクエストされた子プロセスが、 コール時に 既に終了している場合("ゾンビ"プロセスと呼ばれます)、この関数は 直ちに処理を返します。子プロセスにより使用される全てのシステム リソースは、解放されます。waitpid のシステムでの動作に関する詳細は、 システムの waitpid(2) についての man ページを参照ください。

パラメータ

pid

pid の値は、次のどれかとなります。

pid のとりうる値

< -1	プロセスグループ ID が pid の絶対値に等しい 子プロセスを待ちます。
-1	全ての子プロセスを待ちます。これは、wait 関数の動作と同じです。
0	プロセスグループ ID がコール側のプロセスと等しい子プロセスを 待ちます。
> 0	プロセス ID が pid の値に等しい 子プロセスを待ちます。

注意: -1 を pid に指定した際の動きは、 pcntl_wait() の機能と (options を除いて) 同じです。

status

pcntl_waitpid() は、パラメータ status の中にステータス情報を保存します。 このステータスは、次の関数を用いて評価可能です。 pcntl_wifexited()、 pcntl_wifstopped()、 pcntl_wifsignaled()、 pcntl_wexitstatus()、 pcntl_wtermsig() および pcntl_wstopsig() 。

options

options の値は、次の 2 つのグローバル定数の ゼロまたはそれ以上の論理和です。

options のとりうる値

WNOHANG	子プロセスが終了していない場合に直ちに処理を返します。
WUNTRACED	停止した子プロセスの場合に処理を返します。そして、ステータス は報告されません。

返り値

pcntl_waitpid() は、終了した子プロセスの プロセス ID を返します。エラーの場合は -1、WNOHANG が使用され、 子プロセスが利用できない場合に 0 を返します。

参考

- pcntl_fork()
- pcntl_signal()
- pcntl_wifexited()
- pcntl_wifstopped()
- pcntl_wifsignaled()
- pcntl_wexitstatus()
- pcntl_wtermsig()
- pcntl_wstopsig()

---

pcntl_wexitstatus

(PHP 4 >= 4.0.7, PHP 5)

pcntl_wexitstatus — 終了した子プロセスのリターンコードを返す

説明

int **pcntl_wexitstatus** (int $status)

終了した子プロセスのリターンコードを返します。この関数は、 pcntl_wifexited() が **TRUE** を返す場合のみ 有用です。

パラメータ

status

パラメータ status は、 pcntl_waitpid() が正常にコールされた際に得られます。

返り値

リターンコードを表す整数値を返します。

参考

- *pcntl_waitpid()*
- *pcntl_wifexited()*

---

pcntl_wifexited

(PHP 4 >= 4.0.7, PHP 5)

pcntl_wifexited — ステータスコードが正常終了を表しているかどうかを調べる

説明

bool ***pcntl_wifexited*** *(int $status)*

ステータスコードが正常終了を表しているかどうかを調べます。

パラメータ

status

パラメータ status は、 pcntl_waitpid() が正常にコールされた際に得られます。

返り値

子プロセスのステータスコードが終了に成功した場合に ***TRUE****、 それ以外の場合に* ***FALSE*** *を返します。*

参考

- *pcntl_waitpid()*
- *pcntl_wexitstatus()*

---

pcntl_wifsignaled

(PHP 4 >= 4.0.7, PHP 5)

pcntl_wifsignaled — ステータスコードがシグナルによる終了を表しているかどうかを調べる

説明

bool ***pcntl_wifsignaled*** *(int $status)*

子プロセスが終了した原因が、 シグナルが捕捉されなかったことであるかどうかを調べます。 caught.

パラメータ

status

パラメータ status は、 pcntl_waitpid() が正常にコールされた際に得られます。

返り値

捕捉されなかったシグナルのせいで子プロセスが終了した場合に ***TRUE****、 それ以外の場合に* ***FALSE*** *を返します。*

参考

- *pcntl_waitpid()*
- *pcntl_signal()*

---

pcntl_wifstopped

(PHP 4 >= 4.0.7, PHP 5)

pcntl_wifstopped — 子プロセスが現在停止しているかどうかを調べる

説明

bool ***pcntl_wifstopped*** *(int $status)*

リターンを生じた子プロセスが現在停止しているかどうかを調べます。 この関数は、pcntl_waitpid() のコールが オプション WUNTRACED を用い

ている場合のみ使用可能です。

パラメータ

status

パラメータ status は、 pcntl_waitpid() が正常にコールされた際に得られます。

返り値

リターンを生じた子プロセスが現在停止している場合に TRUE、 それ以外の場合に FALSE を返します。

参考

- pcntl_waitpid()

---

pcntl_wstopsig

(PHP 4 >= 4.0.7, PHP 5)

pcntl_wstopsig — 子プロセスを停止させたシグナルを返す

説明

int **pcntl_wstopsig** (int $status)

子プロセスを停止させたシグナル番号を返します。この関数は、 pcntl_wifstopped() が TRUE を返す場合のみ有用です。

パラメータ

status

パラメータ status は、 pcntl_waitpid() が正常にコールされた際に得られます。

返り値

シグナル番号を返します。

参考

- pcntl_waitpid()
- pcntl_wifstopped()

---

pcntl_wtermsig

(PHP 4 >= 4.0.7, PHP 5)

pcntl_wtermsig — 子プロセスの終了を生じたシグナルを返す

説明

int **pcntl_wtermsig** (int $status)

子プロセスを終了させたシグナルの数を返します。この関数は、 pcntl_wifsignaled() が TRUE を返す場合のみ有用です。

パラメータ

status

パラメータ status は、 pcntl_waitpid() が正常にコールされた際に得られます。

返り値

シグナル番号を表す整数値を返します。

参考

- pcntl_waitpid()
- pcntl_signal()
- pcntl_wifsignaled()

目次

- pcntl_setpriority — プロセスの優先度を変更する
- pcntl_signal — シグナルハンドラを設定する
- pcntl_wait — 待つかフォークした子プロセスのステータスを返す
- pcntl_waitpid — 待つかフォークした子プロセスのステータスを返す
- pcntl_wexitstatus — 終了した子プロセスのリターンコードを返す
- pcntl_wifexited — ステータスコードが正常終了を表しているかどうかを調べる
- pcntl_wifsignaled — ステータスコードがシグナルによる終了を表しているかどうかを調べる
- pcntl_wifstopped — 子プロセスが現在停止しているかどうかを調べる
- pcntl_wstopsig — 子プロセスを停止させたシグナルを返す
- pcntl_wtermsig — 子プロセスの終了を生じたシグナルを返す

正規表現関数（Perl 互換）

導入

この正規表現関数で使用するパターンの構文は、Perl と類似しています。 正規表現は、スラッシュ (/) などのデリミタで囲う必要があります。 デリミタとしては、英数字およびバックスラッシュ(\) 以外のすべての文字を使用可能です。 デリミタ文字を正規表現本体において使用する必要がある場合は、 バックスラッシュでエスケープします。PHP 4.0.4 以降、 Perl形式の (), {}, [], <> も使用可能です。 パターンの詳細については、パターン構文 を参照してください。

様々な修飾子を終端デリミタの後に付け、 マッチングに変化を与えることができます。 パターン修飾子 を参照ください。

PHP は、POSIX 拡張正規表現関数 において、 POSIX 拡張構文を用いた正規表現もサポートしています。

> **注意**: この拡張モジュールでは、コンパイルした正規表現のために スレッド単位のグローバルキャッシュ（最大 4096）を管理しています。

警告

PCRE には、いくつかの制限があります。詳細は、» http://www.pcre.org/pcre.txt を参照してください。

要件

外部ライブラリを必要としません。

インストール手順

PHP 4.2.0 以降、本関数はデフォルトで有効となっています。 --without-pcre-regex で PCRE 関数を 無効にすることができます。 付属のライブラリを使用しない場合、 --with-pcre-regex=DIR を使用して PCRE のインクルードおよびライブラリファイルがある場所 DIR を指定してください。 以前のバージョンでは、本関数を使用するためには --with-pcre-regex[=DIR] を指定して PHP を configure およびコンパイルする必要があります。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

php.ini の設定により動作が変化します。

PCRE 設定オプション

名前	デフォルト	変更の可否	変更履歴
pcre.backtrack_limit	"100000"	PHP_INI_ALL	PHP 5.2.0 以降で使用可能
pcre.recursion_limit	"100000"	PHP_INI_ALL	PHP 5.2.0 以降で使用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

pcre.backtrack_limit integer

PCRE のバックトラック処理の制限値です。

pcre.recursion_limit integer

PCRE の再帰処理の制限値です。この値を大きくすると、 使用可能なプロセススタックを使い切ってしまい、 (OS のスタックサイズの制限値に達して) PHP をクラッシュさせてしまうことに注意しましょう。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

PREG 定数

定数	説明
PREG_PATTERN_ORDER	$matches[0] はパターン全体にマッチした文字列の配列、 $matches[1] は第 1 のキャプチャ用サブパターンにマッチした文字列の配列、 といったように結果の順序を指定します。 このフラグは、preg_match_all() でのみ使用されます。
PREG_SET_ORDER	$matches[0] は 1 回目のマッチングでキャプチャした値の配列、 $matches[1] は 2 回目のマッチングでキャプチャした値の配列、 といったように結果の順序を指定します。 このフラグは、preg_match_all() でのみ使用されます。
PREG_OFFSET_CAPTURE	**PREG_SPLIT_OFFSET_CAPTURE** の説明を参照してください。 このフラグは、PHP 4.3.0 以降で利用可能です。
PREG_SPLIT_NO_EMPTY	このフラグは、preg_split() が、空文字列でないものだけ を返すようにします。
PREG_SPLIT_DELIM_CAPTURE	このフラグは、preg_split() が 文字列分割用のパターン中のカッコによるサブパターンでキャプチャされた値も 同時に返すようにします。 このフラグは、PHP 4.0.5 以降で利用可能です。
PREG_SPLIT_OFFSET_CAPTURE	このフラグを設定した場合、各マッチに対応する文字列のオフセットも返されます。 これにより、返り値は配列となり、配列の要素 0 はマッチした文字列、 要素 1 は対象文字列中におけるマッチした文字列のオフセット値となることに注意してください。 このフラグは、*PHP* 4.3.0 以降で利用可能で、 preg_split() のみで使用されます。
PREG_NO_ERROR	エラーが存在しなかった場合に preg_last_error() から返されます。 PHP 5.2.0 以降で使用可能です。
PREG_INTERNAL_ERROR	PCRE 内部エラーが発生した場合に preg_last_error() から返されます。 PHP 5.2.0 以降で使用可能です。
PREG_BACKTRACK_LIMIT_ERROR	backtrack_limit に達した場合に preg_last_error() から返されます。 PHP 5.2.0 以降で使用可能です。
PREG_RECURSION_LIMIT_ERROR	recursion_limit に達した場合に preg_last_error() から返されます。 PHP 5.2.0 以降で使用可能です。
PREG_BAD_UTF8_ERROR	壊れている UTF8 データによって直近のエラーが発生した場合に preg_last_error() から返されます (UTF-8 モード で正規表現を実行した場合のみ)。 PHP 5.2.0 以降で使用可能です。
PCRE_VERSION	PCRE のバージョンおよびリリース日 (例: "7.0 18-Dec-2006")。 PHP 5.2.4 以降で使用可能です。

例

Example#1 有効なパターンの例

- */<¥/¥w+>/*
- */(¥d{3})-¥d+/Sm*
- */^(?i)php[34]/*
- *{^¥s+(¥s+)?$}*

Example#2 無効なパターンの例

- */href='(.*)' - 終端デリミタが抜けている*
- */¥w+¥s*¥w+/J - 未知の修飾子 'J'*
- *1-¥d3-¥d3-¥d4/ - 始端デリミタが抜けている*

パターン修飾子

(No version information available, might be only in CVS)

パターン修飾子 — 正規表現パターンに使用可能な修飾子

説明

現在使用可能な PCRE 修飾子の一覧を以下に示します。 括弧の中の名前は、これらの修飾子に関する PCRE 内部の名前です。 修飾子中での空白文字および改行は無視されます。他の文字はエラーになります。

i (PCRE_CASELESS)
この修飾子を設定すると、パターンの中の文字は 大文字にも小文字にもマッチします。
m (PCRE_MULTILINE)
デフォルトで、PCRE は、検索対象文字列を（実際には複数行からなる 場合でも）単一の行からなるとして処理します。 「行頭」メタ文字 (^) は、対象文字列の最初にしかマッチしません。 一方、「行末」メタ文字 ($) は、文字列の最後、または (D 修飾子が設定されていない場合) 最後にある改行記号の前のみにしかマッチしません。 この動作は Perl と同じです。 この修飾子を設定すると、「行頭」および「行末」メタ文字は 対象文字列において、文字列の最初と最後に加えて、 各改行の直前と直後にそれぞれマッチします。 この動作は、Perl の /m 修飾子と同じです。 対象文字列の中に "\n" 文字がない場合や、 またはパターンに ^ または $ がない場合は、 この修飾子を設定しても意味はありません。
s (PCRE_DOTALL)
この修飾子を設定すると、パターン中のドットメタ文字は 改行を含む全ての文字にマッチします。 これを設定しない場合は、改行にはマッチしません。 この修飾子は、Perl の /s 修飾子と同じです。 [^a] のような否定の文字クラスは、 この修飾子の設定によらず、常に改行文字にマッチします。
x (PCRE_EXTENDED)
この修飾子を設定すると、エスケープするか 文字クラスの内部を除き、 パターンの空白文字は完全に無視されます。 文字クラスの外にあって、かつエスケープされていない # と次の改行文字の間の文字も無視されます。 この動作は、Perl の /x 修飾子と同じであり、複雑なパターンの内部に コメントを記述することが可能となります。 しかし、この修飾子は、データ文字にのみ適用されることに注意してください。空白文字をパターンの特殊文字の並びの中、 例えば条件付きサブパターン (?(の内部に置くことはできません。
e (PREG_REPLACE_EVAL)
この修飾子を設定すると、preg_replace() は、置換文字列において後方参照に関する通常の置換を行った後、 PHP コードとして評価し、検索文字列を置換するためにその結果を 使用します。 置換された後方参照においては、 単引用符や二重引用符、バックスラッシュおよび NULL 文字は バックスラッシュでエスケープされます。 この修飾子を使用するのは、preg_replace()のみです。 他の PCRE 関数では無視されます。
A (PCRE_ANCHORED)

この修飾子を設定すると、パターンは強制的に固定（anchored）となります。 つまり、検索対象文字列の先頭でのみマッチするように制限されます。 パターン自体の中に適当な指定を行うことでも同様の効果を得ることが可能です。 Perl ではパターン中に指定する方法しか使用できません。
D （PCRE_DOLLAR_ENDONLY）
この修飾子を設定すると、パターン内のドルメタ文字は、検索対象文字列の 終わりにのみマッチします。この修飾子を設定しない場合、ドル記号は、 検索対象文字列の最後の文字が改行文字であれば、その直前にもマッチします。 この修飾子は、m を設定している場合に無視されます。 Perl には、この修飾子に等価なものはありません。
S
あるパターンを複数回使用する場合は、マッチングにかかる時間を 高速化することを目的として、パターンの分析に幾分か時間をかけても 良いでしょう。この修飾子を設定すると、追加のパターン分析が 行われます。現在、パターン分析は、最初の文字が単一ではなく、かつ固定でないパターンに対してのみ有用です。
U （PCRE_UNGREEDY）
この修飾子を設定すると、量指定子の「貪欲さ」が反転します。 つまり、量指定子は、デフォルトで貪欲でなく、 疑問符を後ろに付けてはじめて貪欲になるようになります。 この修飾子は Perl 互換では有りません。 同様の設定は、(?U) 修飾子を パターン内で設定 するか、 (.*? のように）量指定子の後に疑問符を 付けるかすることで行うこともできます。
X （PCRE_EXTRA）
この修正子は、Perl 非互換な PCRE の機能を有効にします。 パターン内で後ろに文字が続くバックスラッシュで特別な意味がないものは、 将来的な拡張の際の互換性の維持のため、エラーになります。 デフォルトでは、Perl のように文字が後ろに続くバックスラッシュで特に意味がないものは、リテラルとして処理されます。 この修飾子により制御される機能は、現在の所、これだけです。
J （PCRE_INFO_JCHANGED）
(?J) 内部オプションは、ローカルのオプション PCRE_DUPNAMES の設定を変更します。 サブパターンで重複した名前を使用できるようになります。
u （PCRE_UTF8）
この修正子は、Perl 非互換な PCRE の機能を有効にします。パターン 文字列は、UTF-8 エンコードされた文字列として処理されます。この修正子は、UNIX では PHP 4.1.0 以降、Win32 では PHP 4.2.3 以降で 使用可能です。 また、PHP 4.3.5 以降では、パターンの UTF-8 としての妥当性も確認されます。

パターン構文

(No version information available, might be only in CVS)

パターン構文 — PCRE 正規表現の説明

概要

PCRE ライブラリは、Perl 5 と同じ構文と意味体系を持つ正規表現の パターンマッチを実装した関数群です。ただし、若干の差異もあります （別記参照）。現在の実装は、 Perl 5.005 に対応しています。

Perl との差異

Perl 5.005 との違いについて以下に説明します。

1. デフォルトでは、空白文字は C ライブラリ関数 isspace() が認識する文字となります。PCRE を別の文字型テーブルを用いて コンパイルすることも可能です。通常、isspace() はスペース、改ページ、 改行、復帰、水平タブ、垂直タブにマッチします。 Perl 5 では、現在、 垂直タブが空白文字として扱われていません。 Perl ドキュメントには \v というエスケープが記載されていましたが、実際は認識されていませんでした。 ただし、垂直タブ文字は少なくとも 5.002 までは空白文字として 処理されていました。 5.004 および 5.005 では、\s にマッチしなくなっています。
2. PCRE では、先読み言明に量指定子を指定できません。Perl では可能ですが、思ったような動作を意味しないかもしれません。 例えば、(?!a){3} は、続く 3 文字が "a" でないことの言明ではありません。 この指定は、次の 1 文字が "a" ではないことを 3 回言明するだけです。
3. 否定の先読み言明の中に記述したキャプチャ用サブパターンは カウントされますが、対応するオフセットにそのエントリは セットされません。Perlでは、言明のマッチングに失敗する前に マッチしたパターンからその変数を設定しますが、それが行われるのは、 否定の先読み言明中の選択肢が 1 つだけの場合のみです。
4. ヌル文字は、検索対象文字列においては使用できますが、 パターン文字列内では使用できません。これは、パターン文字列が 0 を終端とする通常の C 文字列として渡されるためです。パターン中では、 エスケープシーケンス "\x00" を使ってヌル文字を表すことができます。
5. 次の Perl エスケープシーケンスはサポートされません。 \l, \u, \L, \U, \E, \Q 。 これらのエスケープシーケンスは、Perl のパターンマッチエンジン内 ではなく、文字列処理の部分で実装されているためです。
6. Perl の \G 言明は、単一のパターンマッチに対しては意味がなく、 サポートされません。
7. 当然ながら、PCRE により、(?{code}) 構文および (??{code}) 構文はサポートされません。しかし、再帰的なパターンはサポートしています。
8. Perl 5.005_02 では、パターンの一部を繰り返すと、キャプチャ文字列の セットに関して奇妙な動作をすることがあるようです。例えば、"aba" を パターン /^(a(b)?)+$/ に対してマッチングを行うと、 $2 には値 "b" が設定されますが、"aabbaa" を /^(aa(bb)?)+$/ に対して マッチングを行うと、$2 はセットされません。 しかし、パターンを /^(aa(b(b))?)+$/ に変えると、 $2 （および $3) はセットされます。 Perl 5.004 では、どちらの場合も $2 はセットされます。 PCRE の場合も、 どちらの場合でもセットされます。将来的に Perl が矛盾のない状態に 変更された場合は、PCRE も追従する可能性があります。
9. 他の未解決の食い違いとして、パターン /^(a)?(?(1)a|b)+$/ は、 Perl 5.005_02 では文字列 "a" にマッチしますが、PCRE ではマッチしない ということがあります。しかし、Perl と PCRE のいずれでも、/^(a)?a/ で "a" をマッチした場合は $1 が未定義のままとなります。
10. Perl の正規表現の機能よりさらに拡張された機能を使うことができます。
    1. 戻り読み言明は、固定長の文字列にマッチする必要がありますが、 このとき、戻り読み言明内の選択肢は、それぞれ異なる長さの文字列に マッチするパターンとしても問題ありません。Perl 5.005 では すべての選択肢が同じ長さである必要があります。
    2. PCRE_DOLLAR_ENDONLY が設定され PCRE_MULTILINE が設定されていない場合、メタ文字 $ は文字列の終端にのみ マッチします。
    3. PCRE_EXTRA を設定すると、バックスラッシュの後に意味がない文字が続くと エラーとなります。
    4. PCRE_UNGREEDY を設定すると、量指定子の貪欲さが反転します。つまり、量指定子は、 デフォルトで貪欲でなく、疑問符を後ろに付けてはじめて 貪欲になるようになります。

正規表現の詳細

導入

PCRE がサポートする正規表現の構文と意味体系を以下に示します。 正規表現については、Perl のドキュメントや他の多くの書籍においても 解説されており、実例が豊富に記載されているものもあります。特に、 O'Reilly 発刊 Jeffrey Friedl 著 "Mastering Regular Expressions" (ISBN 1-56592-257) 〔日本語版：「詳説 正規表現」〕には、 たいへん詳細に解説されています。

以降の説明は、参考文書 (reference documentation) として 扱われることを想定しています。

正規表現は、ある種のパターンであり、検索対象文字列 (subject string) に対して左から右〔文字列の初めから終わり〕の順にマッチングが行われます。 一部の例外を除き、パターン中の文字はその文字自体を表し、 検索対象文字列内の同じ文字にマッチします。簡単な例をあげると、パターン

```
The quick brown fox
```

は、検索対象文字列内にある、このパターンと同一の部分にマッチします。

メタ文字

正規表現の強力さは、パターン中に選択肢や繰り返しを記述できることに あります。選択肢や繰り返しは、メタ文字 (meta-character) を使ってパターン中に記述します。メタ文字は、 その文字自体を表わさず、代わって特別な解釈が行われます。

メタ文字には、2 種類あります。ひとつは、角カッコ内を除き、 パターン中のどこででも使用できる文字です。もうひとつは、 角カッコで括られた中でだけ使用できる文字です。

前者の角カッコ外で使用できるメタ文字には、次のものがあります。

¥
多目的に使う一般的なエスケープ文字
^
検索対象（複数行モードでは行）の始まりを言明
$
検索対象（複数行モードでは行）の終わりを言明
.
改行を除くすべての文字にマッチ（デフォルト時）
[
文字クラス定義の開始
]
文字クラス定義の終了
|
選択枝の開始
(
サブパターンの開始
)
サブパターンの終了
?
(の意味を拡張／0 または 1 回マッチ／なるべく少ない回数だけマッチ
*
0 回以上の繰り返し
+
1 回以上の繰り返し
{
最小／最大を指定する量指定子の開始
}
量指定子の終了

パターン中の角カッコで括まれた部分を「文字クラス」と言います。 文字クラスで使えるメタ文字は、次のものだけです。

¥
一般的なエスケープ文字
^
クラスの否定。ただし、文字クラスの最初の文字に用いた場合のみ
-
文字の範囲の指定
]
文字クラスの終了

以降のセクションで、各メタ文字の使用法の説明を行います。

バックスラッシュ

バックスラッシュ〔日本語環境では円記号となる場合もある〕には、 いくつかの使用法があります。ひとつめの使用法は、 非英数字の前に記述する場合で、続く文字が表す特別な意味を取り去ります。 このエスケープ文字としての使用法は、 文字クラスの内外部いずれでも可能です。

たとえば、"*" 文字とマッチさせたい場合は、パターンを "¥*" と記述します。 続く文字がメタ文字として解釈されるものであるかには関係ありませんので、 いかなる非英数字に対しても、"¥" を付けると、その文字自体が 表わされることになります。特に、 バックスラッシュとマッチさせたい場合は、"¥¥" と記述します。

> **注意**: シングルクォートあるいはダブルクォートで囲まれた PHP の 文字列 の中では、バックスラッシュは特別な意味を表します。 そのため、正規表現 ¥¥ を使用して ¥ とマッチさせたい場合は PHP のコード内では "¥¥¥¥" あるいは '¥¥¥¥' と記述する必要があります。

パターンを、 PCRE_EXTENDED オプションを付けてコンパイルすると、(文字クラス内部を除き) パターン中の空白文字、および "#" とその次の改行文字との間の文字は 無視されます。空白文字や "#" をパターン中に含めるには、 バックスラッシュを用いてエスケープします。

バックスラッシュの 2 番目の使用法は、非表示文字〔制御コードなど〕を パターン中に目に見える形で記述するための方法です。ヌル文字は パターンを終了させてしまうため使えませんが、その他の非表示文字は、 パターンにそのまま含めても問題はありません。しかし、パターンの編集には、 バイナリ文字をそのまま用いるよりも、以下に示すエスケープシーケンスを 用いる方が便利でしょう。

¥a
アラーム、ベル文字 (16進 07)
¥cx
"control-x", ここで x は任意の文字
¥e
エスケープ文字 (16進 1B)
¥f
改ページ (formfeed) (16進 0C)
¥n
改行 (newline) (16進 0A)
¥r
復帰 (carriage return) (16進 0D)
¥t

タブ (16進 09)
¥xhh
16 進コードで hh の文字
¥ddd
8 進コードで ddd の文字、もしくは、後方参照

"\cx" の正確な働きは、次の通りです。"x" が小文字の場合、 大文字に変換されます。続いて、文字の 6 ビット目 (16進数 40) が 反転されます。つまり、"z" は 16 進数の 1A になり、"{" は 3B になり、 ";" は 7B になります。

"\x" の後では、2 桁までの 16 進数が読まれます (大小文字どちらも可能です)。 "\0" の後では、さらに 2 桁の 8 進数が 読みこまれます。いずれの場合も、2 桁より少ない場合、桁があるだけ 読みこまれます。つまり、"\0\x\07" は ヌル文字 2 つの後にベル文字が 続いたものを表します。8 進数を指定する場合は、必ず最初のゼロに続いて 残りの 2 桁の数字を指定するように注意してください。

バックスラッシュの後に 0 以外の数字が続く場合の処理は複雑です。 文字クラスの外部では、 PCRE は、続く数字全体を 10 進数として読みます。数字が 10 よりも小さい場合、または、正規表現の中に含まれる キャプチャ用左カッコの数以下の場合、 後方参照 として解釈されます。 この動作に関する詳しい説明は、後ほど、カッコによるサブパターンの説明を 行ってから示します

文字クラスの中、または、指定された 10 進数が 9 より大きく、 キャプチャ用サブパターンの数がこの数に満たない場合は、PCRE は バックスラッシュの後から最大 3 文字の 8 進数を再度読みこみ、 その値の最下位 8 ビットから 1 バイトを生成します。 その後に続く数字は、それ自体を表します。以下に例を示します。

¥040
スペースの別の表記法
¥40
上と同じ。ただし、キャプチャ用サブパターンが 40 個未満の場合
¥7
常に後方参照
¥11
後方参照、または、タブの別記法
¥011
常にタブ
¥0113
タブの後に文字 "3" が続いたもの
¥113
8進コードで 113 の文字 (99 を超える後方参照は存在しないため)
¥377
全ビットが 1 である 1 バイト
¥81
後方参照、または、ヌル文字の後に 2つの文字 "8" および "1" が続いたもの

3 桁を超えて 8 進数は読みこまれないため、100 以上の 8 進数には ゼロを前につけてはいけないことに注意してください。

これらの 1 バイト値を定義するエスケープシーケンスは、 文字クラスの内外部のいずれでも使用できます。加えて、文字クラス内では エスケープシーケンス "¥b" はバックスペース (16進 0x) として解釈されます。 文字クラス外では、別の意味を有します (別記参照)。

バックスラッシュの第 3 の使用法は、包括的な文字型を指定する用途です。

¥d
10 進数字
¥D
10 進数字でない文字
¥h
水平方向の空白文字 (PHP 5.2.4 以降)
¥H
水平方向の空白文字でない文字 (PHP 5.2.4 以降)
¥s
空白文字
¥S
空白文字でない文字
¥v
垂直方向の空白文字 (PHP 5.2.4 以降)
¥V
垂直方向の空白文字でない文字 (PHP 5.2.4 以降)
¥w
単語構成文字 (word character)
¥W
非単語構成文字 (non-word character)

これらエスケープシーケンスの各組により、文字集合が 2 つに分割されます。 文字は、各組のどちらか片方だけにマッチします。

単語構成文字とは、英字または数字またはアンダースコア文字であり、Perl が定義するところの「単語」と成り得る文字のことです。文字および数字の 定義は、PCRE の文字テーブルにより制御され、ロケールを指定して マッチングを行うと変わる可能性があります。 例えば、"fr" (フランス語) ロケールの場合、128 を超える 文字コードのいくつかは、アクセント付きの文字に使われており、 これらは ¥w とマッチします。

これらの文字型表記は、文字クラスの内外によらず使用可能で、 対応する型のたかだか 1 文字とマッチします。カレントのマッチング位置が 検索対象文字列の終端である場合、マッチできる文字が無いので、 マッチは失敗します。

バックスラッシュの第 4 の使用法は、簡単な言明 (assertion) です。 言明とは、マッチがある特定の位置でだけ可能だという条件を指定するもので、 検索対象文字列から文字を消費 (consume)〔つまり文字自体にマッチ〕 しません。サブパターンを使ったより複雑な言明の方法もありますが、それについての解説は後ほど行います。バックスラッシュを使った言明は、 次のものがあります

¥b
単語境界
¥B
非単語境界
¥A
検索対象文字列の始端 (複数行モードとは独立)
¥Z
検索対象文字列の終端、または終端の改行 (複数行モードとは独立)
¥z
検索対象文字列の終端 (複数行モードとは独立)
¥G
マッチングの開始位置

これらの言明は、文字クラス内では使用できません (また、文字クラス内では、 "¥b" はバックスペース文字という別の意味を持つので 注意してください)。

単語境界 (word boundary) とは、検索対象文字列において、 カレントの文字およびその前の文字が同時に ¥w もしくは ¥W にマッチしない (すなわち、片方が ¥w にマッチし、もう片方が ¥W にマッチする) 位置、もしくは、文字列の始めか終わりで、 その始めか終わりの文字が ¥w にマッチする位置の ことです。

言明 ¥A, ¥Z, ¥z は、（以降で説明する）ハット記号やドル記号とは 異なり、オプション設定によらず、文字列の始端または終端だけに マッチします。 これらの言明は、PCRE_MULTILINE および PCRE_DOLLAR_ENDONLY オプションの影響を受けません。 ¥Z と ¥z との違いは、 ¥Z は文字列の末尾の改行の前の位置および文字列の 終端にマッチするのに対し、¥z は文字列の終端にのみ マッチすることです。

言明 ¥G は、カレントのマッチング位置が、 preg_match() 関数の offset 引数に指定されたマッチングの開始位置である場合に真になります。 offset が非ゼロの場合は、¥A と等価ではありません。PHP 4.3.3 以降で使用可能です。

PHP 4.3.3 以降では、 ¥Q と ¥E とを使って、 パターン中のメタ文字を無視させることができます。 たとえば、

```
¥w+¥Q.$.¥E$
```

は、文字列の終端において1つ以上の単語構成文字のあとに .$. というリテラルが続いたものにマッチします。

¥K を使用すると、マッチの開始位置をリセットできます。 これは PHP 5.2.4 以降で使用可能です。たとえば、パターン foo¥Kbar は "foobar" にマッチしますが、 結果は "bar" にマッチしたと報告されます。 ¥K を使用しても、キャプチャした部分文字列には影響を及ぼしません。 たとえば、パターン (foo)¥Kbar が "foobar" にマッチしたときの最初の部分文字列は "foo" です。

Unicode 文字プロパティ

PHP 4.4.0 および 5.1.0 以降、UTF-8 モード を設定した場合に、 一般的な文字タイプにマッチする新たなエスケープシーケンスが 3 つ追加されました。

¥p{xx}
xx プロパティを持つ文字
¥P{xx}
xx プロパティを持たない文字
¥X
拡張 Unicode シーケンス

ここで xx で表されているプロパティ名は、Unicode で 一般カテゴリプロパティ (general category properties) として規定されているものになります。すべての文字は、いずれかひとつのプロパティを持ちます。 プロパティは、2 文字の略語で表されます。Perl と同じく、 開き波カッコとプロパティ名との間にハット文字を記述することで否定を指定できます。 たとえば、¥p{^Lu} は ¥P{Lu} と同じです。

¥p もしくは ¥P の後に、一文字だけを記述すると その文字で始まるすべてのプロパティが指定されたことになります。 この場合、否定の指定をしていない場合、波カッコを使用しなくても構いません。 以下の 2 つの例は等価になります。

```
¥p{L}
¥pL
```

使用可能なプロパティコード

C	その他 (Other)
Cc	コントロール文字 (Control)
Cf	非可視整形用文字 (Format)
Cn	未定義コードポイント (Unassigned)
Co	私的利用領域 (Private use)
Cs	サロゲート (Surrogate)
L	アルファベット (Letter)
Ll	小文字アルファベット (Lower case letter)
Lm	擬似文字 (Modifier letter)
Lo	その他の文字 (Other letter)
Lt	タイトル文字 (Title case letter)
Lu	大文字アルファベット (Upper case letter)
M	記号 (Mark)
Mc	修飾文字 (Spacing mark)
Me	他の文字を囲むための文字 (Enclosing mark)
Mn	他の文字を修飾するための文字 (Non-spacing mark)
N	数字 (Number)
Nd	10 進数字 (Decimal number)
Nl	数値を表す文字 (Letter number)
No	その他の数字 (Other number)
P	句読記号 (Punctuation)
Pc	連結用句読記号 (Connector punctuation)
Pd	ダッシュ (Dash punctuation)
Pe	閉じ句読記号 (Close punctuation)
Pf	末尾句読記号 (Final punctuation)
Pi	先頭句読記号 (Initial punctuation)
Po	その他の句読記号 (Other punctuation)
Ps	開き句読記号 (Open punctuation)
S	記号 (Symbol)
Sc	通貨記号 (Currency symbol)
Sk	合わせ文字 (Modifier symbol)
Sm	数学記号 (Mathematical symbol)
So	その他の記号 (Other symbol)
Z	区切り文字 (Separator)
Zl	行区切り文字 (Line separator)
Zp	段落区切り文字 (Paragraph separator)
Zs	空白文字 (Space separator)

"Greek" や "InMusicalSymbols" といった拡張プロパティ (extended properties) は PCRE によりサポートされていません。

大小文字を区別しないマッチングを設定していても、これらのエスケープ シーケンスには影響しません。たとえば、¥p{Lu} は 常に小文字にのみマッチします。

¥X は、拡張 Unicode シーケンスを構成する Unicode 文字群に マッチします。¥X は、(?>¥PM¥pM) と等価です。*

つまり、記号 (mark) プロパティの付いていない文字と、その後に続く 0 以上の 記号プロパティ付きの文字にマッチし、その並びをアトミック (atomic) な まとまりとして取り扱います。記号プロパティ付きの文字とは、アクセント記号などの 直前の文字に対して影響するようなもののことです。

Unicode プロパティを使った文字列マッチングは速くありません。PCRE は 15,000 以上のデータからなるストラクチャを検索する必要が有るためです。 そのため、PCRE では、¥d や ¥w といった 以前から有るエスケープシーケンスは Unicode プロパティを使用しないように なっています。

ハット記号とドル記号

ハット記号は、文字クラス外でマッチモード (matching mode) がデフォルトの場合、 カレントのマッチング位置が対象文字列の始端である場合に真だ という言明です。文字クラス内では、ハット記号はまったく別の意味と なります (別記参照)。

パターン中で選択肢を用いる場合は、ハット記号は、それが必要な選択肢の 先頭に記述すればよく、パターン全体の先頭にしか記述できない訳では ありません。すべての選択肢がハット記号で始まる場合、つまり パターンが対象文字列の始端でのみマッチするように制限されている パターンは、固定 (anchored) パターンと呼ばれます (もちろん、 他にもパターンを固定にする方法はあります)。

ドル記号は、（デフォルトでは）カレントのマッチング位置が 対象文字列の終端にあるか、文字列の終わりの改行文字の直前にある場合に 真だという言明です。パターン中で選択肢を用いる場合は、ドル記号は、 必要な選択肢の最後に記述すればよく、パターン全体の最後にしか 記述できない訳ではありません。 ドル記号は、文字クラス内では特別な意味を持ちません。

コンパイル時またはマッチング時に PCRE_DOLLAR_ENDONLY オプションを設定すると、ドル記号の動作を文字列の終端でのみ マッチするように変更することができます このオプションは \Z 言明には影響しません。

ハット記号とドル記号の動作は、PCRE_MULTILINE オプションを設定すると変化します。この場合、対象文字列の始端および 終端にマッチするのに加えて、対象文字列の "\n" 文字の直前および直後に それぞれマッチします。例えば、パターン /^abc$/ は、複数行モードに おいては、対象文字列 "def\nabc" にマッチしますが、 複数行モードでない場合はマッチしません。すなわち、すべての選択肢が "^" で始まっており単一行モードでは固定のパターンも、複数行モードでは 固定パターンでなくなります。PCRE_DOLLAR_ENDONLY オプションは、 PCRE_MULTILINE オプションが設定されている場合は無視されます。

どちらのモードでも、エスケープシーケンス \A, \Z, \z は、 対象文字列の始端および終端にマッチすることに注意してください。ですので、 パターン中の選択肢がすべて \A で始まる場合、PCRE_MULTILINE オプションの設定によらず、そのパターンは固定となります。

ドット

パターン中のドット（ピリオド、終止符）は、文字クラス外では、 対象文字列の任意の 1 文字にマッチします。非出力文字も含まれます。 ただし、（デフォルトでは）改行文字とはマッチしません。 PCRE_DOTALL オプションを設定すると、ドットは改行にもマッチするようになります。 ドットの処理は、ハット記号およびドル記号とは完全に独立しています。 共通な点は共に改行文字と関連することだけです。ドットは、 文字クラス内では特別な意味を持ちません。

¥C は単一バイトの文字にマッチします。 これは、UTF-8 モード で、ドットが複数バイトの文字にも マッチするような場合に有用です。

角カッコ

開き角カッコは文字クラス (character class) の開始を表し、 閉じ角カッコにより文字クラスは終了します。閉じ角カッコだけでは、 特別な意味を持ちません。閉じ角カッコを文字クラスのメンバとしたい場合は、 文字クラスの最初の文字（否定のハット記号がある場合はその直後）とするか、バックスラッシュでエスケープする必要があります。

文字クラスは、検索対象文字列のたかだか 1 文字にマッチします。 マッチする文字は、その文字クラスによって定義された文字集合のうちの どれかです。ただし、文字クラスの最初の文字がハット記号の場合は、 マッチする文字は、その文字クラスに定義されていない文字となります。 ハット記号自体をクラスのメンバとしたい場合は、文字クラスの 最初の文字としないか、バックスラッシュでエスケープしてください。

例えば、文字クラス [aeiou] は小文字の母音にマッチしますが、[^aeiou] は 小文字の母音以外の文字にマッチします。ハット記号は、含まれない文字を 列挙することにより文字クラスに含まれる文字を指定するための簡略表記です。 文字クラスは、言明ではありません。対象文字列から文字を消費します。また、 カレントのポインタが文字列の終端にある場合には、マッチに失敗します。

大小文字を区別しないマッチングを行うよう設定した場合は、 クラス内の文字は大小文字の両方を表します。例えば、 大小文字を区別しない場合、[aeiou] は "a" にも "A" にもマッチします。 同じく大小文字を区別しない場合 [^aeiou] は "A" にマッチしませんが、 区別する場合はマッチします

PCRE_DOTALL または PCRE_MULTILINE オプションをどのように設定しようとも、改行文字は、文字クラスにおいて 特別な扱いはされません。たとえば、 [^a] のようなクラスは、 常に改行にマッチします。

マイナス（ハイフン）記号は、文字クラスで文字の範囲を指定するために 使われます。例えば、[d-m] は d と m の間のあらゆる文字にマッチします。 マイナス記号が文字クラス内に必要な場合は、バックスラッシュで エスケープしてください。もしくは、文字クラスの最初または最後など、 範囲を示すと解釈されない場所に記述してください。

文字リテラル "]" を、文字範囲を示す最後の文字として使うことは できません。 [W-]46] というパターンは、2 つの文字（"W" および"-"）が含まれるクラスの後に文字列リテラル "46]" が続いていると解釈され、 "W46]" や "-46]" にマッチします。しかし、"]" をバックスラッシュでエスケープすると文字範囲の終端として解釈され、 [W-]46] は、 範囲指定の後に 2 つの文字が続く 1 つのクラスとして解釈されます。 "]" の 8 進あるいは 16 進表現も文字範囲の終端として使用可能です。

文字範囲指定では、ASCII 照合順序 (collating sequence) が用いられます 〔つまり、範囲指定する際の文字の並び順として ASCII が用いられます〕。 [\000-\037] のような、数値的な文字の指定も使用可能です。大小文字を 区別しないマッチングを行うよう設定した場合、パターン中の英字は 大小文字の両方にマッチします。例えば、[W-c] は、[][\^_`wxyzabc] に 等価であり、大小文字に関係なくマッチします。また、 "fr" ロケールの文字テーブルを使う場合、[\xc8-\xcb] は、 大小文字の区別無くアクセント付きの E にマッチします。

文字型 \d, \D, \s, \S, \w, \W も、文字クラス内で使え、 マッチする文字を追加することが可能です。例えば、[ABCDEF] は、 16進数にマッチします。ハット記号と大文字の文字型を組み合わせることで、 小文字の文字型がマッチするものより狭い文字集合を 簡便に指定することができます。例えば、クラス [^\W_] は、 〔単語構成〕文字および数字にマッチしますが、アンダースコアには マッチしません。

\, -, (始端の)^ および終端の] 以外のすべての非英数字は、 文字クラスにおいて特別な意味を持たない文字ですが、 エスケープしても問題はありません。

縦線

縦線は、パターンに選択肢 (alternative) を列挙するために使われます。 例えば、パターン

```
gilbert|sullivan
```

は、"gilbert" または "sullivan" にマッチします。 選択肢の数に制限はありません。また、空の選択肢も可能です（空の文字列にマッチします）。マッチの手順としては、 各選択肢が左から右に順にマッチするかどうか試行され、 最初にマッチに成功した選択肢が使われます。選択肢を（後述する） サブパターン内に記述した場合、マッチの「成功」とは、 サブパターン内の選択肢も、メインのパターンの他の部分もマッチした ということを意味します。

内部オプション設定

PCRE_CASELESS, PCRE_MULTILINE, PCRE_DOTALL PCRE_UNGREEDY, PCRE_EXTRA, PCRE_EXTENDED および PCRE_DUPNAMES オプションの設定は、パターン中で変更できます。これには、Perl オプション文字を "(?" と ")" とで括った並びを用います。 オプション文字には以下のものがあります。

内部オプション文字

i	PCRE_CASELESS
m	PCRE_MULTILINE
s	PCRE_DOTALL
x	PCRE_EXTENDED
U	PCRE_UNGREEDY
X	PCRE_EXTRA
J	PCRE_INFO_JCHANGED

例えば、(?im) は、大小文字を区別しない、複数行モードのマッチングを 設定します。ハイフンを前につけることにより、そのオプションを 解除することも可能です。 (?im-sx) のように設定と解除とを 組み合わせることも可能です。この場合、PCRE_CASELESS と PCRE_MULTILINE とが設定され、PCRE_DOTALL と PCRE_EXTENDED とが解除されます。 オプション文字がハイフンの前にも後にも指定された場合、 そのオプションは解除されます。

オプションの変更がトップレベル（つまりサブパターンのカッコの外）で行われた場合、 その変更はパターンの後ろの部分に適用されます。 パターン /ab(?i)c/ は "abc" および "abC" にマッチします。 PHP 4.3.3 以降にバンドルされている PCRE 4.0 以降で、このような動作に変更になりました。 以前のバージョンでは、/ab(?i)c/ は /abc/i と同じ処理をされていました（つまり "ABC" や "aBc" にもマッチしました）。

サブパターンの内部でオプションの変更を行った場合は、 その効果は違ったものになります。この動作変更は、Perl 5.005 で行われました。サブパターン中のオプション変更は、そのサブパターンの 設定が行われた場所以降の部分にのみ影響します。そのため、パターン

```
(a(?i)b)c
```

は、abc および aBc にマッチし、他の文字列にはマッチしません （PCRE_CASELESS が設定されていないと仮定）。このように、パターンの各場所に 異なった設定を行うようなオプション指定が可能です。 ある選択肢に行われた設定変更は、同じサブパターン中の後に続く選択肢に 波及します。例えば、パターン

```
(a(?i)b|c)
```

は、"ab", "aB", "c", "C" にマッチします。"C" にマッチする場合を 見てみると、オプション設定が行われている最初の選択肢は マッチせずに捨てられてしまっています。それでも、オプションの設定は 行われ "C" にマッチします。これは、オプション設定の効果が コンパイル時に生じることによります。さもないと、 非常に奇妙な動作をするかもしれません。

PCRE 特有のオプション PCRE_UNGREEDY および PCRE_EXTRA は、 それぞれ文字 U および X を用いて、Perl 互換のオプションと同様に 変更することが可能です。 (?X) フラグの設定は特別で、最上位においても、 パターン中で他の設定を有効にする前に指定する必要があります。 このフラグは、最初に指定するのが最善です。

サブパターン

サブパターンは、丸カッコで括られたパターンのことで、ネストも可能です。 パターンの一部をサブパターンにすると、以下の 2 つのことが可能になります。

1. 選択肢の範囲の指定 (localize)。例えば、パターン

```
cat(aract|erpillar|)
```

は、単語 "cat", "cataract", "caterpillar" にマッチします。 カッコをつけないと、このパターンは、"cataract", "erpillar" または空の文字列にマッチしてしまいます。

2. サブパターンによる値の取得（キャプチャ）。 パターン全体としてマッチに成功した場合、対象文字列の内、 サブパターンにマッチした部分の値がコールした側に返されます。 開きカッコの数が（1 から始まって）左から右に数えられ、 キャプチャ用サブパターン (capturing subpattern) の番号が指定されます。

例えば、文字列 "the red king" に対し、パターン

```
the ((red|white) (king|queen))
```

をマッチングさせた場合、キャプチャされる部分文字列は、 "red king", "red", "king" であり、 それぞれ 1 番, 2 番, 3 番と番号がふられます。

カッコに 2 つの機能があるということが、いつも良い方に働くわけでは ありません。値をキャプチャする必要はないが、グループ分けのために サブパターンを複数用いたい場合も少なくありません。開きカッコの後に "?:" を付けると、そのサブパターンは値のキャプチャを行わず、 キャプチャ用サブパターンの番号としてもカウントされません。例えば、 文字列 "the white queen" に対し、次のパターンをマッチングさせてみましょう。

```
the ((?:red|white) (king|queen))
```

キャプチャされる部分文字列は、"white queen" と "queen" であり、 それぞれ 1 番と 2 番に番号付けされます。キャプチャ可能な部分文字列の数は最大で 99 までです。また、キャプチャを行うものと行わないものを 合わせて、サブパターンの数は最大 200 までです。

簡略形として、値のキャプチャをしないサブパターンの先頭で オプションの設定をする場合、オプションの文字を "?" と ":" の間に入れることができます。つまり、次の 2 つのパターン、

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

は、まったく同じ文字列集合にマッチします。 選択肢は左から右に試行され、オプションはサブパターンの終端に達するまで リセットされないので、ある選択枝にあるオプション設定は後に続く 選択枝にも作用します。このため、上のパターンは、"Saturday" と同様に "SUNDAY" にもマッチします

PHP 4.3.3 以降、(?P<name>pattern) という記法を用いて サブパターンに名前をつけることができる様になりました。 マッチ時に返される配列には、番号による添字に加えて、指定した文字列を添字とする 要素も含まれます。

繰り返し

繰り返し (repetition) は、量指定子 (quantifier) を使って指定します。 量指定子は、以下の要素の後に付けることが出来ます。

- 個々の文字。エスケープされた文字も含む
- メタ文字 . （ドット）
- 文字クラス
- 後方参照（次のセクションを参照）
- カッコで括られたサブパターン（言明は除く - 別記参照）

汎用の量指定子 (general repetition quantifier) は、波カッコの中に 2 つの数をカンマで区切って記述したもので、マッチ可能な 最小と最大の繰り返し数を指定します。ただし、65536 以上の数は 指定できません。さらに、最初の数は 2 番目の数以下である必要があります。 例えば、パターン

```
z{2,4}
```

は、"zz", "zzz", "zzzz" にマッチします。閉じ波カッコだけでは 特別な意味を持ちません。カンマを残したまま 2 番目の数だけを省略すると、繰り返しの上限が設定されません。 2 番目の数字とカンマの両方を省略すると、 必要なマッチの数を過不足なく指定できます。つまり、

```
[aeiou]{3,}
```

は、最短で 3 回母音が連続するものにマッチし、 もっと多く連続する場合にもマッチします。一方、

```
¥d{8}
```

は、ぴったり 8 桁の数字にのみマッチします。開き波カッコは、 量指定子を置けない場所、つまり量指定子の構文に適合しない場所に 記述された場合、文字リテラルとして解釈されます。例えば、 {,6} は量指定子ではなく、4つの文字からなる文字リテラルとなります。

{0} という量指定子の指定も可能です。 直前の項目および量指定子が存在しないという指定になります。

簡単（および歴史的な互換性）のため、よく使われる 3 つの量指定子には、 次のような 1 文字の省略型があります。

単一文字の量指定子

*	{0,} と同じ
+	{1,} と同じ
?	{0,1} と同じ

たとえば

```
(a?)*
```

の様に、「文字無し」にマッチし得るサブパターンの後ろに 上限指定の無い量指定子を記述すると、無限ループができてしまう可能性が あります。

以前のバージョンの Perl および PCRE は、このようなパターンに関して コンパイル時にエラーを発生していました。しかし、有用な場合もあるので、 現在はこのようなパターンも許可されています。ただし、 繰り返しが指定されたサブパターンが文字無しにマッチすると、 ループは強制的に中断されます。

デフォルトでは、量指定子は「貪欲 (greedy)」です。つまり、 残りのパターンが失敗しない限りにおいて、（許可された回数の上限まで） 出来るだけ多くマッチします。 この動作が問題を生じる場合もあります。 そのよく知られた例としては、 C プログラムのコメントに マッチさせようとする場合が挙げられます。 /* と */ との間が コメントですが、コメント中にも文字 * や / が現れる可能性があります。 そこで、C のコメントにマッチさせようとして、パターン

```
/¥*.*¥*/
```

を使って、文字列

```
/* first comment */  not comment  /* second comment */
```

に対して、マッチングを行うと、文字列全体にマッチしてしまい上手く行きません。 .* 要素が貪欲であるためです。

しかし、量指定子の後に疑問符を付けると、貪欲さは消え、 できるだけ少ない回数だけマッチします。このため、パターン

```
/¥*.*?¥*/
```

は、C コメントに正しくマッチします。量指定子の他の意味は変化せず、 マッチの回数だけが変更されます。疑問符のこの使用法と、 量指定子としての使用法とを混同しないようにしてください。 このように疑問符には 2 種類の使用法があるため、

```
¥d??¥d
```

のように 2 重に使うこともできます。 このパターンは、可能であれば 1 桁の数字にマッチします。 パターンの残り部分がそうでないとマッチできない場合に限り、 2 桁の数字にもマッチします。

PCRE_UNGREEDY オプション（Perl ではこのオプションは利用できません）を設定すると、 量指定子は、デフォルトで貪欲でなくなり、 各量指定子の後ろに疑問符をつけてはじめて貪欲になります。言いかえると、 疑問符のデフォルトの動作を逆転します。

量指定子の後ろに + を記述すると「独占的 (possessive)」 になります。可能なだけ多くの文字を取り込み、パターンの残りの部分のマッチの際に取り込んだ文字を戻すことはしません。したがって、パターン .*abc は "aabc" にマッチしますが、.*+abc はマッチしません。 これは、.*+ が対象文字列のすべてを取り込んでしまうためです。 PHP 4.3.3 以降で、処理速度の向上を目的として、独占的量指定子を使うことが可能となりました。

カッコを使ったサブパターンに、最小の繰り返し回数（1 以上）や 最大の繰り返し回数を指定した場合、その繰り返し数の大きさに応じて、 コンパイル済みのパターンが用いる記憶領域がより多く必要となります。

パターンが .* または .{0,} で始まっており、PCRE_DOTALL オプション（Perl の /s に相当）が設定されている（つまり、 . が改行文字にもマッチする）場合、そのパターンは暗黙的に 固定パターンになります。.* もしくは .{0,} の後に続くパターンは、 対象文字列のすべての位置でマッチングが行われますが、パターン全体としては 対象文字列の始端以外でマッチングを再試行しても無駄だからです。PCRE は、 このようなパター

ンについては、その前に \A が記述されているものとして 扱います。こうした最適化を利用するために、 対象文字列が改行文字を含んでいないことが明らかな場合は、 .* で始まるパターンに対し PCRE_DOTALL を設定するか、 ^ を用いて明示的に固定パターンとなるよう指定すると良いでしょう。

キャプチャ用サブパターンを繰り返した場合、キャプチャされる値は、 繰り返しの最後でマッチした部分文字列です。例えば、

```
(tweedle[dume]{3}¥s*)+
```

を "tweedledum tweedledee" にマッチングさせた場合、キャプチャされる 部分文字列は、 "tweedledee" です。しかし、キャプチャ用サブパターンが ネストしている場合、キャプチャされる値は、より前の繰り返しで 得られたものとなる可能性があります。例えば

```
/(a|(b))+/
```

を "aba" にマッチングさせると、キャプチャされた部分文字列の 2 番目のものは、 "b" になります。

後方参照

文字クラス外で、バックスラッシュに続いて 1 以上の数値（複数桁可） を記述したものは、パターン中のより前方（すなわち左）にある キャプチャ用サブパターンに対する後方参照 (back reference) です。ただし、 その左方に、その数値以上の個数のキャプチャ用サブパターンの開きカッコがある必要があります。

なお、バックスラッシュの後に 10 未満の 10 進数が続く場合は、 常に後方参照として解釈され、パターン全体で指定した個数以上の キャプチャ用サブパターンが無いとエラーが発生します。言いかえると、 参照されるカッコは、10未満の番号に対しては、参照する側の左にある必要が ないということです。バックスラッシュの後に数字が続く場合の処理の詳細に ついては、前述の 「バックスラッシュ」のセクションを参照してください。

後方参照は、カレントの対象文字列においてキャプチャ用サブパターンが 実際にマッチした文字列にマッチします。サブパターンがパターンとしてマッチし得るものではありません。すなわち、パターン

```
(sens|respons)e and ¥1ibility
```

は、"sense and sensibility" および "response and responsibility" にマッチしますが、 "sense and responsibility" にはマッチしません。また、 後方参照が記述されている位置で大小文字を区別するマッチングが有効ならば、 文字の大小文字の別も関係します。例えば、

```
((?i)rah)¥s+¥1
```

は、"rah rah" および "RAH RAH" にマッチしますが、 元のキャプチャ用サブパターンは大小文字を区別しないマッチングを 行っているにもかかわらず、"RAH rah" にはマッチしません

同じサブパターンに対して、複数回の後方参照を行うことができます。 また、使われなかったサブパターンに対する後方参照を行おうとすると、 マッチが失敗します。例えば、パターン

```
(a|(bc))¥2
```

は、はじめに "bc" でなく "a" にマッチした場合は、マッチが失敗します。 最大 99 番までの後方参照を使用できるため、バックスラッシュの後に数字が続くものはすべて後方参照番号の可能性があるものとして解釈されます。 後に数字が続く場合、後方参照を終了するためになんらかの区切り文字を 置く必要があります。PCRE_EXTENDED オプションを設定している場合は空白文字を区切り文字として使えます。 その他の場合は空のコメントを使います。

後方参照を、それ自身が参照するサブパターンのカッコ内に記述した場合、 そのサブパターンが最初に使われた際にマッチが失敗します。ですので、(a¥1) は、何にもマッチしません。しかし、このような参照は、 複数回繰り返されるサブパターンの内部では有用です。例えば、パターン

```
(a|b¥1)+
```

は、"a" が連続するものや "aba", "ababaa" 等にマッチします。 サブパターンが繰り返される場合、後方参照は、直前の繰り返しで一致した 文字列にマッチします。こうしたパターンを動作させるためには、 繰り返しの1 回目に、後方参照を含むパターンとのマッチングが 行われないことが必要です。これには、上の例のように選択肢を使うか、 下限が 0 回の量指定子を使います。

名前を指定したサブパターンへの後方参照を行うには (?P=name) とします。PHP 5.2.4 以降では、これ以外に ¥k<name> や ¥k'name'、¥k{name} あるいは ¥g{name} などとすることも可能です。

言明

言明 (assertion) とは、カレントのマッチング位置の直前・直後の文字に対する テストであり、文字を消費 (consume)〔つまり文字自体にマッチ〕しません。 単純な言明コード \b, \B, \A, \Z, \z, ^ および $ については、 前述しました。より複雑な言明は、サブパターンを用いて記述します 〔言明サブパターン (assertion subpattern) と呼ばれる〕。 言明サブパターンには、2 種類あります。対象文字列において カレントの位置の先を読むものと、後ろを読むものです。

言明サブパターンは、カレントのマッチング位置を変更しないことを除き、 通常と同じようにマッチングが行われます。先読み言明 (lookahead assertion) は、 肯定の言明 (positive assertion) の場合 (?= で始まり、 否定の言明 (negative assertion) の場合 (?! で始まります。例えば、

```
¥w+(?=;)
```

は、セミコロンが後に続く単語にマッチしますが、 マッチ対象それ自体にはセミコロンは含まれません。また、

```
foo(?!bar)
```

は、"bar" が後ろに続かない "foo" にマッチします。 なお、一見、良く似たパターンですが

```
(?!foo)bar
```

は、"foo" 以外のものの後にある "bar" を見つけるものではないことに 注意してください。これは、どこにある "bar" とでもマッチしてしまいます。 続く 3 文字が "bar" である場合、 (?!foo) は常に真となってしまうからです。 このような探索を実現するためには、戻り読み言明 (lookbehind assertion) が必要です。

戻り読み言明は、肯定の言明の場合 (?<= で始まり、 否定の言明の場合 (?<! で始まります。例えば、

```
(?<!foo)bar
```

は、"foo" 以外の後にある "bar" の存在を見つけるものです。 戻り読み言明内のパターンは、それがマッチし得る文字列の長さが 固定でなければなりません〔繰り返しを指定できません〕。ただし、 選択肢を用いた場合、各選択肢は〔固定長でなければいけませんが〕 すべて同じ長さである必要はありません。つまり、

```
(?<=bullock|donkey)
```

とはできますが、

```
(?<!dogs?|cats?)
```

は、コンパイル時にエラーになります。 戻り読み言明の最上位においてのみ、異なる長さの文字列にマッチするような 選択肢が使用可能です。Perl 5.005 においては、すべての選択肢が 同じ長さの文字列にマッチする必要があります。つまり、この機能は PCRE の拡張機能です。

```
(?<=ab(c|de))
```

という戻り読み言明は、最上位にひとつの選択肢しかなく、 その選択肢は異なる長さの文字列にマッチしうるので、不正です。 しかし、

```
(?<=abc|abde)
```

のように、最上位において選択肢を 2 つ用いるように 書き換えると使用可能です。

後方言明の実装においては、選択肢ことに一時的に固定の幅だけ カレントの位置を後退させ、マッチを試みます。カレントの位置の前に 十分な文字がない場合は、マッチは失敗とみなされます。 再試行無しのサブパターンと組み合わせた戻り読み言明は、 文字列の終端でのマッチングに特に有用です。 再試行無しのサブパターンについてのセクションの最後にて例を示します。

（任意の種類の）複数の言明を連続して指定することも可能です。例えば、

```
(?<=¥d{3})(?<!999)foo
```

は、"999" でない 3 桁の数字の後にある "foo" にマッチします。 それぞれの言明は、対象文字列の同じ場所に独立して適用されることに 注意して下さい。まず、前の 3 文字がすべて数字であることがチェックされ、 続いて、同じ 3 文字が "999" でないことが確認されます。このパターンは、"foo" の前に 6 個の文字があり、その前半が数字で後半の 3 文字が "999" でないというパターンにマッチするものではありません。例えば、"123abcfoo" にはマッチしません。これを行うパターンは次のようになります。

```
(?<=¥d{3}...)(?<!999)foo
```

この時、最初の言明は、先行する 6 つの文字を探し、 最初の 3 文字が数字であることを確認します。続いて、2 番目の言明は、 先行する 3 文字が "999" でないことを確認します。

言明は、自由に組み合わせてネスト可能です。例えば、

```
(?<=(?<!foo)bar)baz
```

は、"foo" 以外の後にある "bar" の後に有る "bar" があればマッチします。 一方、

```
(?<=¥d{3}...(?<!999))foo
```

は、999 でない 3 桁の数字とさらに 3 文字の後に続く "foo" にマッチするパターンです。

言明サブパターンは、キャプチャ用サブパターンではありません 〔値のキャプチャは行われません〕。繰り返しもできません。 同じことを複数回言明しても意味がないからです。 キャプチャ用サブパターンが言明の内部に含まれている場合、 言明の種類に関係なく、キャプチャ用サブパターンの番号付けにあたって カウントされます。しかし、文字列のキャプチャは、肯定の言明に対してのみ 行われます。否定の言明の中で行っても無意味だからです。

カッコ付サブパターンによる言明は、最大 200 まで用いることができます。

再試行無しのサブパターン

繰り返し回数の下限もしくは上限の指定をした場合、〔繰り返しを指定した要素の〕続きがマッチに失敗すると、繰り返し指定した要素が再評価され、繰り返し回数を変えた上で残りのパターンがマッチするかどうか試されます。マッチングを続けても無駄なことが明らかな場合、マッチングの性質を変え、より速くマッチに失敗させるために、こうした動作を停止させることが有用な場合があります。

例えば、パターン \d+foo を "123456bar" という対象文字列に適用した場合を考えてみましょう。

\d+ が 6 桁の数字すべてにマッチしますが、その後 "foo" とのマッチが失敗します。通常のマッチング処理の動作だと、5桁の数字のみが \d+ にマッチするとして再試行され、次いで 4 桁等々と続けられ、最後には完全にマッチが失敗します。再試行無しのサブパターン (once-only subpattern) を用いると、パターンの一部が一度マッチしたら、その後再評価されないよう指定することができます。つまり、最初に "foo" とのマッチに失敗した時点で、ただちにマッチングを取り止めることが可能となります。表記には、

```
(?>¥d+)bar
```

のように、(?> で始まる特別なカッコを用います。

この種類のカッコは、一度マッチしたパターンの部分に鍵をかけ (lock up) ます。そのパターンへの再マッチは失敗し、バックトラック (backtrack) が起こらないようにします。それより前の要素に対するバックトラックは、通常と同様に動作します。

別の説明をすると、このタイプのサブパターンは、同一のスタンドアローンの パターンが対象文字列のカレントの位置に固定されたかのように、文字列とマッチします。

再試行無しのサブパターンは、キャプチャ用サブパターンではありません〔つまり、値のキャプチャは行われません〕。上の簡単な例では、できるだけ多くのものを呑み込むよう繰り返しが最大化されました。つまり、+ や +? は残りのパターンがマッチするよう数字の桁数が調整されるのに対して、(?>+) は数字の並び全体に対するマッチングだけしか行われません。

この構文には、どんな複雑なものでも、任意のサブパターンを含むことができ、ネストも可能です。

再試行無しのサブパターンと戻り読み言明とを組み合わせると、対象文字列の終端における効率的なマッチングを行うことができます。

```
abcd$
```

というパターンを見てましょう。マッチが成功しない長い文字列に適用した場合を考えます。マッチングは左から右に行われるため、PCRE は対象文字列のすべての "a" を探し、後に続く文字が残りのパターンにマッチするかどうか調べられます。パターンを

```
^.*abcd$
```

のように少し変更してみます。この場合、最初の .* は、まず文字列全体にマッチします。("a" がその後に続かないので) マッチが失敗すると、最後の 1 文字を除く 文字列にマッチするようバックトラックが行なわれ、続いて最後の 2 文字を 除く文字列に、という風に動作します。"a" の検索は、やはり文字列全体に 対して、右から左に、行われるため効率は良くありません。しかし、パターンを

```
^(?>.*)(?<=abcd)
```

のようにしてみましょう。要素 .* に対してバックトラックは行われず、文字列全体にのみマッチします。続く戻り読み言明は、最後の 4 文字に対するテストを 1 回だけ行います。テストが失敗すると、マッチはただちに失敗します。長い文字列に対しては、この方法を用いると実行時間にかなりの差が生じます。

パターン中にサブパターンがあって、その中に繰り返し数に 上限の無い要素があり、そのサブパターン自身も何回でも繰り返しが 可能な場合、マッチの失敗に非常に長い時間がかかってしまう事があります。それを避ける唯一の方法は再試行無しのサブパターンを使うことです。パターン

```
(¥D+|<¥d+>)*[!?]
```

は、非数字もしくは <> で括られた数字に ! または ? が続く 任意の長さの部分文字列にマッチします。マッチが成功するような 対象文字列に対しては、速く動作します。しかし、これを

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

に適用すると、マッチが失敗するまでに長い時間がかかります。これは、この対象文字列を分割するやり方が数多くあり、それらすべてに対し マッチを試みられる事になるためです。(この例で、終端に単一の文字ではなく [!?] を使っているのは、PCRE と Perl の双方とも、〔終端に〕単一の文字が使われると、より速く失敗と判定できるように最適化が行われる〔ので、それを避ける〕ためです。マッチに必要な最後の一文字が記憶され、文字列にその文字が無い場合、早期に失敗と判定されます。) このパターンを

```
((?>¥D+)|<¥d+>)*[!?]
```

のように変更した場合、非数字の部分が分割されることがなくなるので、より速くマッチが失敗するようになります。

条件付きサブパターン

サブパターンを条件付きでマッチング処理させることが可能です。言明の結果や直前のキャプチャ用サブパターンがマッチしたかどうかにより、サブパターン中の 2 つの選択肢を選択させます。条件付きサブパターン (conditional subpattern) には、2 つの形式があります。

```
(?(条件)真パターン)
(?(条件)真パターン|偽パターン)
```

条件が満たされた場合、真パターンが使われます。そうでない場合は、(もしあれば) 偽パターンが使われます。サブパターン内に 3 つ以上の 選択肢があると、コンパイル時にエラーになります。

条件には 2 種類あります。条件のカッコ内が数値の場合、その番号のキャプチャ用サブパターンがマッチしている場合に条件が 真となります。カッコ内の数値は 1 以上でないといけません。次のパターンを見てみましょう。可読性を高めるために意味のない空白文字が 挿入され (PCRE_EXTENDED オプションが指定されていると仮定します)、説明を簡単にするため 3 つの部分に分割されています。

```
( ¥( )?    [^()]+    (?(1) ¥) )
```

最初の部分は、オプションで、開きカッコにマッチします。開きカッコが有る場合、1 番のキャプチャ文字列にセットされます。第 2 の部分は、カッコでない一つ以上の文字にマッチします。第 3 の部分は、最初のカッコがマッチしたかどうかをテストする条件付きサブパターンです。カッコにマッチしている場合、つまり、対象文字列が開きカッコで 始まっている場合、条件は真となり、真パターンが実行され、閉じカッコが 必要となります。そうでない場合、偽パターンが存在しないため、サブパターンは何にもマッチしません。言いかえると、このパターンは、カッコなしの並びにマッチするものですが、オプションで カッコで囲まれた並びにもマッチします。

(R) という文字列を条件として指定すると、パターン全体もしくは サブパターンへの再帰的呼び出しがなされた場合に、その条件が満たされることになります。「トップレベル」においては、この条件は偽となります。

数値の他にも、条件として言明が使用できます。先読み言明と 戻り読み言明のいずれも使え、肯定の言明も否定の言明も使用できます。 次のパターンを見てみましょう。このパターンにも意味のない空白文字が 挿入されており、2 行目に 2 つの選択肢が 書かれています。

```
(?(?=[^a-z]*[a-z])
¥d{2}-[a-z]{3}-¥d{2}  |  ¥d{2}-¥d{2}-¥d{2} )
```

条件は、肯定の先読み言明であり、英字以外の文字の並びの後に英字が 続くものにマッチします。言いかえると、対象文字列に少なくとも英字が 一つあるかどうかが調べられます。英字がみつかると、最初の選択肢に対して 検索対象とのマッチングが行われます。みつからない場合は、2番目の選択肢に 対してマッチングが行われます。このパターンは、2つの形式 dd-aaa-dd または dd-dd-dd のどちらかの文字列にマッチします。ここで、aaa は英字、 dd は数字です。

コメント

コメントは、(?# という並びにより始まり、次の閉じカッコで終わります。 カッコのネストはできません。コメント内の文字は、パターンマッチには全く関係しません。

PCRE_EXTENDED オプションが設定されている場合は、パターン中の文字クラス外にある エスケープされていない # 文字からもコメントが始まり、次の改行文字で終わります。

再帰的パターン

どうすればカッコに括られた文字列とマッチできるか、という問題を 考えて見ましょう。このとき、カッコは何回でもネストできるとします。 再帰が使えないとすると、パターンを用いて、せいぜい、ある一定の深さの ネストまでしかマッチできないでしょう。任意の深さのネストを 処理することは不可能です。Perl 5.6 では、正規表現において再帰を行う 実験的な機能が導入されています。 PCRE では、再帰という特殊なケースに対して専用のシーケンス (?R) が導入されました。上記のカッコ問題を解決する PCRE のコードは 以下のようになります (PCRE_EXTENDED オプションが設定され空白文字が無視されると仮定)。

```
¥( ( (?>[^()]+) | (?R) )* ¥)
```

まず、このパターンは開きカッコにマッチします。続いて、 カッコ以外の文字が並んでいるか、または、再帰的にパターン自体に マッチする (すなわち正しくカッコで括られている) かする部分文字列に 何回でもマッチします。最後に閉じカッコにマッチします。

このパターン例には、ネストした無制限の繰り返しが含まれているため、 マッチが成功し得ない文字列に対してこのパターンが適用される場合も考えて、 カッコ以外の文字列にマッチングさせる部分に再試行無しのサブパターンを 使うことが重要です。例えば、このパターンを

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

という文字列に適用すると、「マッチしない」という判定が速やかに 行われます。しかし、再試行無しのサブパターンを使わないと、 対象文字列を分割し得る + と * の繰り返し数の種類が非常に多く、 そのすべてが確認された後にマッチの失敗が報告されるため、 マッチングに非常に時間がかかります。

キャプチャ用サブパターンにセットされる値は、そのサブパターンに 値がセットされ得る最も外側で最も後の再帰レベルからのものになります。 上記のパターンを

```
(ab(cd)ef)
```

にマッチングさせると、キャプチャされる値は "ef" であり、 最上位レベルの最後の値です

```
¥( ( ( (?>[^()]+) | (?R) )* ) ¥)
```

次のようにカッコを追加すると、キャプチャされる文字列は、"ab(cd)ef" となり、最上位レベルのカッコの中身となります。一つのパターン中に 15 以上のキャプチャ用サブパターンを用いると、PCRE は 再帰を行っている間のデータ保存用に追加の記憶領域を確保する必要が あります。記憶領域が確保できない場合、メモリ不足エラーを再帰の内側から 出力する手段がないため、最初の 15 個のキャプチャ用サブパターンに ついてのみデータが保存されます。

PHP 4.3.3 以降、(?1), (?2) といった記法が 再帰的サブパターンとして使用可能になりました。 (?P<name>foo) あるいは (?P&name) といった名前付きのサブパターンも使用可能です。

この再帰的サブパターンの記法は (数字によるものも名前によるものの双方とも) 参照するカッコの外で使用でき、プログラム言語のサブルーチンの様に処理されます。 前に示した例で、パターン (sens|respons)e and ¥1ibility は "sense and sensibility" および "response and responsibility" にはマッチしますが "sense and responsibility" にはマッチしませんでした。 一方、パターン (sens|respons)e and (?1)ibility は、 上記 2 つに加え "sense and responsibility" にもマッチします。 ただし、参照先のサブパターンの後に記述する必要があります。

対象文字列の最大の長さは、integer 型の最大の値となります。 しかし、PCRE は再帰を使用してサブパターンの処理や繰り返し処理を行います。つまり、そのパターンで処理する対象文字列の長さによって、 使用可能なスタック空間の大きさが制限されるということです。

パフォーマンス

パターンに記述可能な要素のうち、幾つかの要素は、他の要素よりも 効率的に処理されます。 (a|e|i|o|u) のような選択肢の集合よりも [aeiou] のような文字クラスの方が効率的です。一般に、構文が 最も単純なものが、たいてい最も効率が良くなります。Jeffrey Friedl の本には、正規表現の性能を向上させる最適化について 様々な検討が行われています。

パターンが .* で始まり、PCRE_DOTALL オプションを設定した場合、対象文字列の始端でしかマッチできないため、 PCRE は、暗黙のうちにそのパターンを固定パターンとみなします。しかし、 PCRE_DOTALL を設定しない場合、メタ文字 . が改行にマッチせず、 対象文字列が改行を含む場合、パターンは文字列の始端からではなく各改行の 直後の文字からマッチする可能性があるため、PCRE は この最適化を行うことができません。 例えば、パターン

```
(.*) second
```

を、対象文字列 "firstnd second"（ここで、は改行文字を意味します）に マッチングさせます。 1 番目のキャプチャ文字列は、"and" になります。 このような動作をするには、 PCRE は対象文字列の各改行文字の後から マッチングを繰り返し行う必要があります。

このようなパターンを、改行を含まない対象文字列に適用する場合は、 PCRE_DOTALL を設定するか、固定パターンであることを明示的に示すためにパターンを ^.* で開始すると最高の性能が得られます。そうすることで、PCRE が対象文字列の改行を探し、そこからマッチングが再び始められることを 防止します。

制限のない繰り返しをネストするようなパターンについては注意が必要です。 そのようなパターンが、マッチが成功し得ない文字列に適用された場合には、 実行に長い時間を要します。

```
(a+)*
```

というパターンを考えてみましょう。

このパターンが "aaaa" にマッチし得る方法は 33 通りもあります（つまり、 * による繰り返しは 0, 1, 2, 3 もしくは 4 回の繰り返しにマッチし、 0 回以外のそれぞれの場合について、 + による繰り返しは様々な回数分 マッチします）〔訳注：a 1文字の 4 回繰り返しとか、 a 1文字とa 3文字の 組合せとかを意味する〕。この数は、文字列が長くなるにつれて急激に 増大します。このパターンの後ろに、マッチが失敗するような別のパターンが 続いていて、パターン全体としてマッチが失敗してしまう場合、PCREは、基本的に 取り得るすべての選択肢を調べるため、非常に長い時間がかかります。

```
(a+)*b
```

のように単一の文字リテラルが最後にある場合には最適化が行われます。 標準のマッチング手順に着手する前に、PCRE は対象文字列の後方に "b" があるかどうかを調べます。無い場合には、直ちにマッチは失敗します。 文字リテラルが最後にない場合には、この最適化は行われません。

```
(a+)*¥d
```

というパターンと上に挙げたパターンの動作を比較してみましょう。 "a" 文字だけが連続する行に適用した場合、前者のパターンでは、 ほぼ瞬間的に失敗と判定されます。一方、後者のパターンでは、 文字列の長さがおよそ 20 文字を超えると、かなりの時間がかかります。

---

preg_grep

(PHP 4, PHP 5)

preg_grep — パターンにマッチする配列の要素を返す

説明

array **preg_grep** (string $pattern , array $input [, int $flags])

input 配列の要素のうち、 指定した pattern にマッチするものを要素とする配列を返します。

パラメータ

pattern

検索するパターンを表す文字列。

input

入力の配列。

flags

PREG_GREP_INVERT を設定すると、この関数は 与えた pattern にマッチ しない 要素を返します。

返り値

input 配列のキーを使用した配列を返します。

変更履歴

バージョン	説明
4.2.0	パラメータ flags が追加されました。
4.0.4	このバージョンより前は、返される配列の添字は input 配列のキーにかかわらず設定されていました。 以前のこの挙動がお好みの場合は、返される配列に array_values() を適用し、添字を再設定してください。

例

Example#1 preg_grep() の例

```php
<?php
// すべての浮動小数点数を含む配列要素を返す
$fl_array = preg_grep("/^(¥d+)?¥.¥d+$/", $array);
?>
```

---

preg_last_error

(PHP 5 >= 5.2.0)

preg_last_error — 直近の PCRE 正規表現処理のエラーコードを返す

説明

int **preg_last_error** (void)

直近の PCRE 正規表現処理のエラーコードを返します。

Example#1 preg_last_error() の例

```php
<?php

preg_match('/(?:¥D+|<¥d+>)*[!?]/', 'foobar foobar foobar');

if (preg_last_error() == PREG_BACKTRACK_LIMIT_ERROR) {
    print 'Backtrack limit was exhausted!';
}

?>
```

上の例の出力は以下となります。

```
Backtrack limit was exhausted!
```

返り値

以下の定数のいずれかを返します (別のページで説明します)。

- **PREG_NO_ERROR**
- **PREG_INTERNAL_ERROR**
- **PREG_BACKTRACK_LIMIT_ERROR** (pcre.backtrack_limit も参照ください)
- **PREG_RECURSION_LIMIT_ERROR** (pcre.recursion_limit も参照ください)
- **PREG_BAD_UTF8_ERROR**

---

preg_match_all

(PHP 4, PHP 5)

preg_match_all — 繰り返し正規表現検索を行う

説明

int **preg_match_all** (string $pattern , string $subject , array &$matches [, int $flags [, int $offset]])

subject を検索し、 pattern に指定した正規表現にマッチした すべての文字列を、flags で指定した 順番で、matches に代入します。

正規表現にマッチすると、そのマッチした文字列の後から 検索が続行されます。

パラメータ

pattern

検索するパターンを表す文字列。

subject

入力文字列。

matches

マッチしたすべての内容を含む、 flags で指定した形式の多次元配列。

flags

以下のフラグの組み合わせ (**PREG_PATTERN_ORDER** を **PREG_SET_ORDER** と組み合わせて使用することは 無意味ですので注意してください)。

PREG_PATTERN_ORDER

$matches[0] はパターン全体にマッチした文字列の配列、 $matches[1] は第 1 のキャプチャ用サブパターンにマッチした文字列の配列、といった順番となります。

```php
<?php
preg_match_all("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_PATTERN_ORDER);
```

```
echo $out[0][0] . ", " . $out[0][1] . "¥n";
echo $out[1][0] . ", " . $out[1][1] . "¥n";
?>
```

上の例の出力は以下となります。

```
<b>example: </b>, <div align=left>this is a test</div>
example: , this is a test
```

$out[0] はパターン全体にマッチした文字列の配列を有しており、 $out[1] はタグで囲まれた文字列の配列を有しています。

PREG_SET_ORDER

$matches[0] は 1 回目のマッチングでキャプチャした値の配列、 $matches[1] は 2 回目のマッチングでキャプチャした値の配列、 といった順序となります。

```
<?php
preg_match_all("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=¥"left¥">this is a test</div>",
    $out, PREG_SET_ORDER);
echo $out[0][0] . ", " . $out[0][1] . "¥n";
echo $out[1][0] . ", " . $out[1][1] . "¥n";
?>
```

上の例の出力は以下となります。

```
<b>example: </b>, example:
<div align="left">this is a test</div>, this is a test
```

PREG_OFFSET_CAPTURE

このフラグを設定した場合、各マッチに対応する文字列のオフセットも返されます。 これにより、返り値は配列となり、配列の要素 0 はマッチした文字列、 要素 1 は subject における マッチした文字列のオフセット値となることに注意してください。

flags を指定しない場合は、 **PREG_PATTERN_ORDER** が指定されたことになります。

offset

通常、検索は対象文字列の先頭から開始されます。 オプションのパラメータ offset を使用して 検索の開始位置を（バイト単位で）指定することも可能です。

注意: offset を用いるのと、 substr($subject, $offset) を **preg_match_all()**の対象文字列として指定するのとは 等価ではありません。 これは、pattern には、 ^, $ や (?<=x) のような言明を含めることができるためです。 これに関する例については、 preg_match() を参照してください。

返り値

パターンがマッチした総数を返します（ゼロとなる可能性もあります）。 または、エラーが発生した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.3	パラメータ offset が追加されました。
4.3.0	フラグ **PREG_OFFSET_CAPTURE** が追加されました。

例

Example#1 テキストからすべての電話番号を得る

```
<?php
preg_match_all("/¥(?  (¥d{3})?  ¥)?  (?(1)  [¥-¥s] ) ¥d{3}-¥d{4}/x",
               "Call 555-1212 or 1-800-555-1212", $phones);
?>
```

Example#2 HTML タグにマッチするものを見付ける（貪欲）

```
<?php
// \\2 は後方参照の例。これは、pcre に正規表現中の括弧の 2 番目の
// 組、つまりこの場合は ([\w]+)、にマッチする。文字列が二重引用符で
// 括られているため、バックスラッシュの追加が必要。
$html = "<b>bold text</b><a href=howdy.html>click me</a>";

preg_match_all("/(<([¥w]+)[^>]*>)(.*)(<¥/¥¥2>)/", $html, $matches, PREG_SET_ORDER);

foreach ($matches as $val) {
    echo "matched: " . $val[0] . "¥n";
    echo "part 1: " . $val[1] . "¥n";
    echo "part 2: " . $val[3] . "¥n";
    echo "part 3: " . $val[4] . "¥n¥n";
}
?>
```

上の例の出力は以下となります。

```
matched: <b>bold text</b>
part 1: <b>
part 2: bold text
part 3: </b>
```

```
matched:  <a  href=howdy.html>click  me</a>
part  1:  <a  href=howdy.html>
part  2:  click  me
part  3:  </a>
```

参考

- preg_match()
- preg_replace()
- preg_split()

---

preg_match

(PHP 4, PHP 5)

preg_match — 正規表現によるマッチングを行う

説明

int **preg_match** (string $pattern , string $subject [, array &$matches [, int $flags [, int $offset]]])

pattern で指定した正規表現により subject を検索します。

パラメータ

pattern

検索するパターンを表す文字列。

subject

入力文字列。

matches

matches を指定した場合、検索結果が代入されます。 $matches[0] にはパターン全体にマッチしたテキストが代入され、 $matches[1] には 1 番目ののキャプチャ用サブパターンにマッチした 文字列が代入され、といったようになります。

flags

flags には以下のフラグを指定できます。

PREG_OFFSET_CAPTURE
このフラグを設定した場合、各マッチに対応する文字列のオフセットも返されます。 これにより、返り値は配列となり、配列の要素 0 はマッチした文字列、 要素 1は対象文字列中におけるマッチした文字列のオフセット値 となることに注意してください。

offset

通常、検索は対象文字列の先頭から開始されます。 オプションのパラメータ offset を使用して 検索の開始位置を（バイト単位で）指定することも可能です。

注意: offset を用いるのと、 substr($subject, $offset) を **preg_match()**の対象文字列として指定するのとは 等価ではありません。 これは、pattern には、 ^, $ や (?<=x) のような言明を含めることができるためです。 以下を比べてみてください。

```php
<?php
$subject = "abcdef";
$pattern = '/^def/';
preg_match($pattern, $subject, $matches, PREG_OFFSET_CAPTURE, 3);
print_r($matches);
?>
```

上の例の出力は以下となります。

```
Array
(
)
```

一方、この例を見てください。

```php
<?php
$subject = "abcdef";
$pattern = '/^def/';
preg_match($pattern, substr($subject,3), $matches, PREG_OFFSET_CAPTURE);
print_r($matches);
?>
```

出力は以下のようになります。

```
Array
(
    [0] => Array
        (
            [0] => def
            [1] => 0
        )
```

```
)
```

返り値

preg_match() は、*pattern* がマッチした回数を返します。つまり、*0* 回（マッチせず）または *1* 回となります。 これは、最初にマッチした時点で**preg_match()** は検索を止めるためです。逆に[preg_match_all()](#)は、 *subject* の終わりまで検索を続けます。 **preg_match()** は、エラーが発生した場合に**FALSE**を返します。

変更履歴

バージョン	説明
4.3.3	パラメータ *offset* が追加されました。
4.3.0	フラグ **PREG_OFFSET_CAPTURE** が追加されました。
4.3.0	パラメータ *flags* が追加されました。

例

Example#1 文字列 "php" を探す

```
<?php
// パターンのデリミタの後の "i" は、大小文字を区別しない検索を示す
if (preg_match("/php/i", "PHP is the web scripting language of choice.")) {
    echo "A match was found.";
} else {
    echo "A match was not found.";
}
?>
```

Example#2 単語 "web" を探す

```
<?php
/* パターン内の \b は単語の境界を示す。このため、独立した単語の
 *  "web"にのみマッチし、"webbing" や "cobweb" のような単語の一部にはマッチしない */
if (preg_match("/¥bweb¥b/i", "PHP is the web scripting language of choice.")) {
    echo "A match was found.";
} else {
    echo "A match was not found.";
}

if (preg_match("/¥bweb¥b/i", "PHP is the website scripting language of choice.")) {
    echo "A match was found.";
} else {
    echo "A match was not found.";
}
?>
```

Example#3 URL からドメイン名を得る

```
<?php
// get host name from URL
preg_match('@^(?:http://)?([^/]+)@i',
    "http://www.php.net/index.html", $matches);
$host = $matches[1];

// get last two segments of host name
preg_match('/[^.]+¥.[^.]+$/', $host, $matches);
echo "domain name is: {$matches[0]}¥n";
?>
```

上の例の出力は以下となります。

```
domain name is: php.net
```

注意

ヒント

ある文字列が他の文字列内に含まれているかどうかを調べるためだけに **preg_match()** を使うのは避けた方が良いでしょう。 [strpos()](#) か [strstr()](#) 関数を 使う方が速くなります。

参考

- [preg_match_all()](#)
- [preg_replace()](#)
- [preg_split()](#)

---

preg_quote

(PHP 4, PHP 5)

preg_quote — 正規表現文字をクオートする

説明

string **preg_quote** (string $str [, string $delimiter])

preg_quote() は、str を引数とし、正規表現構文の特殊文字の前にバックスラッシュを挿入します。 この関数は、実行時に生成される文字列をパターンとしてマッチングを行う必要があり、 その文字列には正規表現の特殊文字が含まれているかも知れない場合に有用です。

正規表現の特殊文字は、次のものです。 . ¥ + * ? [^] $ ( ) { } = ! < > | :

パラメータ

str

入力文字列。

delimiter

オプションの delimiter を指定すると、 ここで指定した文字もエスケープされます。これは、PCRE 関数が使用する デリミタをエスケープする場合に便利です。'/' がデリミタとしては 最も一般的に使用されています。

返り値

クォートされた文字列を返します。

例

Example#1 preg_quote() の例

```php
<?php
$keywords = '$40 for a g3/400';
$keywords = preg_quote($keywords, '/');
echo $keywords; // \$40 for a g3\/400 を返します
?>
```

Example#2 テキスト内の単語の斜体変換

```php
<?php
// この例では、preg_quote($word) を使って、アスタリスクが
// 正規表現での特殊な意味を帯びないようにしています

$textbody = "This book is *very* difficult to find.";
$word = "*very*";
$textbody = preg_replace ("/" . preg_quote($word) . "/",
                          "<i>" . $word . "</i>",
                          $textbody);
?>
```

注意

> **注意**: この関数はバイナリデータに対応しています。

---

preg_replace_callback

(PHP 4 >= 4.0.5, PHP 5)

preg_replace_callback — 正規表現検索を行い、コールバック関数を使用して置換を行う

説明

mixed **preg_replace_callback** (mixed $pattern , callback $callback , mixed $subject [, int $limit [, int &$count]])

この関数の動作は、ほぼ preg_replace() と同じですが、 replacement の代わりに callback を指定するところが異なります。

パラメータ

pattern

検索するパターン。 文字列あるいは文字列の配列のいずれかとなります。

callback

このコールバック関数は、検索対象文字列でマッチした要素の配列が指定されて コールされます。このコールバック関数は、置換後の文字列を返す必要があります。

しばしば、1カ所だけで **preg_replace_callback()**用の callback 関数が必要となることがあります。 この場合、create_function()を使用して、 **preg_replace_callback()** をコールする際 に使用するコールバック関数として匿名の関数を宣言することができます。 このようにすることにより、コールに関するすべての情報を 1 ヶ所に集め、 他の場所で使用されないコールバック関数名で関数の名前空間を 汚染しないようにすることができます。

Example#1 preg_replace_callback() と create_function()

```php
<?php
/* Unix 方式のコマンドラインフィルタです。
 * 段落の冒頭の大文字を、小文字に変換します。*/
$fp = fopen("php://stdin", "r") or die("標準入力から読み込めません");
while (!feof($fp)) {
    $line = fgets($fp);
    $line = preg_replace_callback(
        '|<p>¥s*¥w|',
```

```
        create_function(
            // ここでは単一引用符の使用が不可欠です。
             // そうでない場合は、$ をすべて \$ とエスケープします。
            '$matches',
            'return  strtolower($matches[0]);'
        ),
        $line
    );
    echo $line;
}
fclose($fp);
?>
```

subject

文字列あるいは文字列の配列で、 検索および置換の対象となる文字列を指定します。

limit

subject 文字列における 各パターンの最大置換回数。デフォルトは -1（無制限）です。

count

指定した場合は、置換を行った回数がここに格納されます。

返り値

***preg_replace_callback()** は、 subject が配列の場合には配列を、 それ以外の場合は文字列を返します。*

マッチするものが見つかった場合は新しい subject を返し、それ以外の場合はもとの subject をそのまま返します。

変更履歴

バージョン	説明
5.1.0	パラメータ *count* が追加されました。

例

Example#2 preg_replace_callback() の例

```
<?php
// このテキストは 2002 に使われていたものなのですが、
// これを 2003 年対応の日付に変更したいのです
$text = "エイプリルフールの日付は 04/01/2002 です\n";
$text.= "この前のクリスマスの日付は 12/24/2001 でした\n";
// コールバック関数
function next_year($matches)
{
  // 通常は、$matches[0] がマッチした全体を表します。
   // $matches[1] は、マッチした中で、パターン内の最初の '(...)'
  // にあてはまる部分を表します。それ以降も同様です。
  return $matches[1].($matches[2]+1);
}
echo preg_replace_callback(
            "|(¥d{2}/¥d{2}/)(¥d{4})|",
            "next_year",
            $text);

?>
```

上の例の出力は以下となります。

```
エイプリルフールの日付は 04/01/2003 です
この前のクリスマスの日付は 12/24/2002 でした
```

Example#3 カプセル化された BB code を処理するための、 preg_replace_callback() での再帰構造の使用

```
<?php
$input = "通常の位置 [indent] 字下げ [indent] もっと字下げ [/indent] 字下げ [/indent] 通常の位置";

function parseTagsRecursive($input)
{

    $regex = '#¥[indent](([^[]|¥[(?!/?indent])|(?R))+)¥[/indent]#';

    if (is_array($input)) {
        $input = '<div style="margin-left: 10px">'.$input[1].'</div>';
    }

    return preg_replace_callback($regex, 'parseTagsRecursive', $input);
}

$output = parseTagsRecursive($input);

echo $output;
?>
```

参考

- *preg_replace()*
- *create_function()*

- callback 型に関する情報

---

preg_replace

(PHP 4, PHP 5)

preg_replace — 正規表現検索および置換を行う

説明

mixed **preg_replace** (mixed $pattern , mixed $replacement , mixed $subject [, int $limit [, int &$count]])

subject に関して pattern を用いて検索を行い、 replacement に置換します。

パラメータ

pattern

検索を行うパターン。文字列もしくは配列とすることができます。

e 修飾子を設定すると、**preg_replace()** は、参照先の対応する置換を行う際に replacement 引数を PHP コードであるとして取り扱います。 replacement には有効な PHP コードを記述してください。 さもないと、**preg_replace()** がある行でパースエラーが 発生してしまいます。

replacement

置換を行う文字列もしくは文字列の配列。 この引数が文字列で、pattern 引数が配列の場合、 すべてのパターンがこの文字列に置換されます。 pattern および replacement のいずれもが配列の場合、各 pattern は 対応する replacement に置換されます。 もし、replacement 配列の要素の数が pattern 配列よりも少ない場合は、余った pattern は 空文字に置換されます。

replacement では、 ¥¥n 形式または $n 形式 (PHP 4.0.4 以降) で参照を指定することができます。 後者の形式の方が好ましい形式です。各参照は、n 番目のキャプチャ用サブパターンにマッチしたテキストにより置換されます。 n は 0 から 99 までとすることができ、 ¥¥0 または $0 はパターン全体にマッチするテキストを参照します。キャプチャ用サブパターンの番号 については、その左括弧が左から右に (1から) カウントされます。

後方参照の直後に他の数字が続くような置換 (replacement) パターンを 使用する場合 (すなわち、マッチしたパターンの直後に数字リテラルを置く場合) 、後方参照を行うために通常の ¥¥1 表記を 使用することができません。例えば、¥¥11 は、 後方参照 ¥¥1 の後にリテラル 1 が続くのか、後方参照 ¥¥11 で その後には何も続かないのかが不明のため、 **preg_replace()** を混乱させる可能性があります。 この場合、解決策は、¥${1}1 を使用することです。 こうすることで、1 はリテラルとなり、後方参照 $1 を明確に作成できます。

e 修飾子を使用する際に、 この関数は後方参照を置換する文字列のうちの特定の文字 (具体的には '、"、 ¥ および NULL) をエスケープします。これは、後方参照をシングルクォートやダブルクォートを共用した場合 (たとえば 'strlen(¥'$1¥')+strlen("$2")') に構文エラーが発生しないようにするためのものです。 PHP の 文字列構文 を意識し、 文字列がどのように解釈されるのかを正確に知っておくようにしましょう。

subject

検索・置換対象となる文字列もしくは文字列の配列

subject が配列の場合、検索と置換は subject の各要素に対して行われ、返り値も配列となります。

limit

subject 文字列において、各パターンによる 置換を行う最大回数。デフォルトは -1 (制限無し)。

count

この引数が指定されると、置換回数が渡されます。

返り値

preg_replace() は、 subject 引数が配列の場合は配列を、 その他の場合は文字列を返します。

パターンがマッチした場合、〔置換が行われた〕新しい subject が返されます。マッチしなかった場合、subject が そのまま返されます。

変更履歴

バージョン	説明
5.1.0	count 引数が追加されました。
4.0.4	replacement 引数に '$n' 形式が使用できるようになりました。
4.0.2	limit 引数が追加されました。

例

Example#1 数字リテラルが後に続く後方参照

```
<?php
$string  = 'April 15, 2003';
$pattern  = '/(¥w+) (¥d+), (¥d+)/i';
$replacement  = '${1}1,$3';
echo preg_replace($pattern, $replacement, $string);
?>
```

上の例の出力は以下となります。

```
April1,2003
```

Example#2 添字配列の使用

```
<?php
$string = 'The quick brown fox jumped over the lazy dog.';
$patterns[0] = '/quick/';
$patterns[1] = '/brown/';
$patterns[2] = '/fox/';
$replacements[2] = 'bear';
$replacements[1] = 'black';
$replacements[0] = 'slow';
echo preg_replace($patterns, $replacements, $string);
?>
```

上の例の出力は以下となります。

```
The bear black slow jumped over the lazy dog.
```

pattern と replacement を ksort すると、所望のものが得られます。

```
<?php
ksort($patterns);
ksort($replacements);
echo preg_replace($patterns, $replacements, $string);
?>
```

上の例の出力は以下となります。

```
The slow black bear jumped over the lazy dog.
```

Example#3 複数値の置換

```
<?php
$patterns = array ('/(19|20)(¥d{2})-(¥d{1,2})-(¥d{1,2})/',
                   '/^¥s*{(¥w+)}¥s*=/');
$replace = array ('¥3/¥4/¥1¥2', '$¥1 =');
echo preg_replace($patterns, $replace, '{startDate} = 1999-5-27');
?>
```

上の例の出力は以下となります。

```
$startDate = 5/27/1999
```

Example#4 'e' 修飾子の使用

```
<?php
preg_replace("/(<¥/?)(¥w+)([^>]*>)/e",
             "'¥¥1'.strtoupper('¥¥2').'¥¥3'",
             $html_body);
?>
```

入力テキストのすべての HTML タグを大文字に変換します。

Example#5 空白の削除

この例は、文字列から余分な空白を取り除きます。

```
<?php
$str = 'foo   o';
$str = preg_replace('/¥s¥s+/', ' ', $str);
// This will be 'foo o' now
echo $str;
?>
```

Example#6 count 引数の使用

```
<?php
$count = 0;

echo preg_replace(array('/¥d/', '/¥s/'), '*', 'xp 4 to', -1 , $count);
echo $count; //3
?>
```

上の例の出力は以下となります。

```
xp***to
3
```

注意

> **注意**: pattern および replacement を使用する際、配列の並び順に処理されます。添字は整数であっても、 その並びは値の小さい順になっているとは限りません。 ですから、配列の添字を使って、どの pattern が、どの replacement に置換されるかを指定しようとする場合は、 **preg_replace()** をコールする前に、各配列に対し ksort() を実行しておくべきです。

参考

- preg_match()

- preg_replace_callback()
- preg_split()

---

preg_split

(PHP 4, PHP 5)

preg_split — 正規表現で文字列を分割する

説明

array **preg_split** (string $pattern , string $subject [, int $limit [, int $flags]])

指定した文字列を、正規表現で分割します。

パラメータ

pattern

検索するパターンを表す文字列。

subject

入力文字列。

limit

これを指定した場合、最大 limit 個の部分文字列が返されます。そして、 limit が -1 の場合は「制限が無い」ことを意味します。 これは、flags を指定する場合に有用です。

flags

flags は、次のフラグを組み合わせたものとする （ビット和演算子 | で組み合わせる）ことが可能です。

PREG_SPLIT_NO_EMPTY
　　このフラグを設定すると、空文字列でないものだけが **preg_split()** により返されます。
PREG_SPLIT_DELIM_CAPTURE
　　このフラグを設定すると、文字列分割用のパターン中の カッコによるサブパターンでキャプチャされた値も同時に返されます。
PREG_SPLIT_OFFSET_CAPTURE

　　このフラグを設定した場合、各マッチに対応する文字列のオフセットも返されます。 これにより、返り値は配列となり、配列の要素 0 はマッチした文字列、 要素 1 は subject におけるマッチした文字列のオフセット値となることに 注意してください。

返り値

pattern にマッチした境界で分割した subject の部分文字列の配列を返します。

変更履歴

バージョン	説明
4.3.0	**PREG_SPLIT_OFFSET_CAPTURE** が追加されました。
4.0.5	**PREG_SPLIT_DELIM_CAPTURE** が追加されました。
4.0.0	パラメータ flags が追加されました。

例

Example#1 preg_split() の例 : 検索文字列のある部分を取得

```
<?php
// カンマまたは " ", \r, \t, \n , \f などの空白文字で句を分割する。
$keywords = preg_split("/[¥s,]+/", "hypertext language, programming");
?>
```

Example#2 文字列を文字要素に分割

```
<?php
$str = 'string';
$chars = preg_split('//', $str, -1, PREG_SPLIT_NO_EMPTY);
print_r($chars);
?>
```

Example#3 文字列をマッチするものとそのオフセットに分割

```
<?php
$str = 'hypertext language programming';
$chars = preg_split('/ /', $str, -1, PREG_SPLIT_OFFSET_CAPTURE);
print_r($chars);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => Array
        (
            [0] => hypertext
            [1] => 0
        )
```

```
    [1] => Array
        (
            [0] => language
            [1] => 10
        )

    [2] => Array
        (
            [0] => programming
            [1] => 19
        )

)
```

注意

ヒント

正規表現の威力を必要としないのなら、より高速な（機能はシンプルですが）代替関数として explode() あるいは str_split() のような選択肢があります。

参考

- splitI()
- split()
- implode()
- preg_match()
- preg_match_all()
- preg_replace()

目次

PDF 関数

導入

PHP の PDF 関数を使用すると、 Thomas Merz が作成して現在は » PDFlib GmbH がメンテナンスしている PDFlib ライブラリを使用した PDF ファイルが作成できるようになります。

本節のドキュメントは、PDFlib ライブラリで利用可能な関数の概要のみを 説明することを意図しており、完全なリファレンスではありません。 ここで扱う各関数の完全で詳細な説明については、PDFlib GmbH が配布する すべての PDFlib パッケージに含まれている PDFlib リファレンスマニュアルを 参照ください。このドキュメントは、PDFlib の機能に関する 概要を非常に良くまとめており、全ての関数に関する最新のドキュメントが 含まれています。

はじめの一歩としては、 PDFlib 配布パッケージに含まれるサンプルプログラムを眺めることをお勧めします。 このサンプルでは、基本的なテキスト、ベクター、グラフィックスの出力だけではなく、 PDF インポート機能（PDI）のような高度な関数も扱っています。

PDFlib のほとんどの関数と PHP モジュール内の関数の名前とパラメータは共通になっています。 別途設定されていない限り、 全ての長さと座標は、Postscript のポイント数で計られます。通常、1 インチあたり 72 Postscript ポイントですが、これは出力解像度に依存します。 使用する座表系に関するより詳細な説明については、PDFlib の配布物に含まれる PDFlib リファレンスマニュアルを参照ください。

PDFlib のバージョン 6 では、PHP 4 用の関数指向の API に加えて PHP 5 用のオブジェクト指向の API も提供しています。 主な違いは以下のとおりです。

PHP 4 では、まず最初に以下のような関数コールによって PDF リソースを取得しなければなりませんでした。

```
$p = PDF_new().
```

この PDF リソースは、それ以降のすべての関数コールの 第一パラメータとして次のように使用されます。

```
PDF_begin_document($p, "", "").
```

しかし、PHP 5 では PDFlib オブジェクトは次のように作成します。

```
$p = new PDFlib().
```

このオブジェクトは、PDFlib API のすべての関数をメソッドとして 提供しています。たとえば次のようになります。

$p->begin_document("", "").

さらに、PHP 5 の新機能である例外についても PDFlib 6 以降でサポートしています。

詳細な情報は、以下の例を参照ください。

> ***注意**: 外部 PDF ライブラリを使用しない、他のフリーな PDF ジェネレータに 関心がある場合には、 関連する FAQ を参照してください。*

要件

PDFlib Lite はオープンソースで公開されていますが、 フリーで使用するためにはいくつかの条件があります。 PDFlib Lite は、PDFlib の機能の一部をサポートしています。詳細は PDFlib のウェブサイトを参照ください。 PDFlib の完全版は » http://www.pdflib.com/products/pdflib-family/ でダウンロード可能ですが、 商用する場合はライセンスを購入する必要があります。

古いバージョンの PDFlib に関する問題

2000 年 3 月 9 日以降のバージョンの PHP 4 では、3.0 より古いバージョンの PDFlib をサポートしていません。

PDFlib 4.0 以降は、PHP 4.3.0 以降でサポートされています。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。 この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/pdflib.

PHP < 4.3.9 で以下の関数が動作するようにするには、 --with-pdflib[=DIR] を指定して PHP をコンパイルする必要があります。DIR は PDFlib のベースインストールディレクトリで、デフォルトは /usr/local です。

リソース型

PDF_new() は新しい PDFlib オブジェクトを作成します。 これはほとんどの PDF 関数で必要となります。

廃止された PDFlib 関数についての注意

PHP 4.0.5 以降、PHPlib 用の PHP 拡張モジュールは、PDFlib GmbH から 正式にサポートされています。これにより、PDFlib リファレンスマニュアルに記述された全ての関数が PHP4 で全く同じ意味、 同じパラメータでサポートされています。しかし、PDFlib バージョン 5.0.4 以降ではすべてのパラメータを指定する必要があります。互換性を保つために PDFlib サポート関数ではまだ古い関数もサポートしていますが、上記の ように新しいバージョンに置換される予定です。PDFlib GmbH は、これらの 古い関数を使用した場合に生じた際に生じた問題に関してはサポートを 行いません。このドキュメントではそれらの関数については「古い関数」と 明記しており、かわりに使用する関数について説明しています。

例

多くの関数の使用法は簡単です。最も困難なのは、最初に pdf ドキュメントを作成する場合でしょう。次の例は、入門の際の 助けとなるはずです。この例は PHP 4 を対象に開発されており、 1 ページを有するファイル test.pdf が作成されます。 ドキュメントにはフィールドの内容についての情報が定義されており、 Helvetica-Bold フォントを読み込んで "Hello world! (says PHP)" というテキストを出力します。

Example#1 PHP 4 用の PDFlib での Hello World の例

```
<?php
$p = PDF_new();

/*  新しい PDF ファイルをオープンし、ディスク上に PDF を作成するためにファイル名を挿入します */
if (PDF_begin_document($p, "", "") == 0) {
    die("Error: " . PDF_get_errmsg($p));
}

PDF_set_info($p, "Creator", "hello.php");
PDF_set_info($p, "Author", "Rainer Schaaf");
PDF_set_info($p, "Title", "Hello world (PHP)!");

PDF_begin_page_ext($p, 595, 842, "");

$font = PDF_load_font($p, "Helvetica-Bold", "winansi", "");

PDF_setfont($p, $font, 24.0);
PDF_set_text_pos($p, 50, 700);
PDF_show($p, "Hello world!");
PDF_continue_text($p, "(says PHP)");
PDF_end_page_ext($p, "");

PDF_end_document($p, "");

$buf = PDF_get_buffer($p);
$len = strlen($buf);

header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=hello.pdf");
print $buf;

PDF_delete($p);
?>
```

以下の例は PHP 5 用の PDFlib 配布物で使用するためのものです。PHP 5 の 新機能である例外処理やオブジェクトのカプセル化機能を使用しています。 この例では hello.pdf という名前の 1 ページの ファイルを作成します。 ドキュメントにはフィールドの内容についての情報が定義されており、 Helvetica-Bold フォントを読み込んで "Hello world! (says PHP)" というテキストを出力します。

Example#2 PHP 5 用の PDFlib での Hello World の例

```
<?php
```

```
try {
     $p = new PDFlib();

    /*  新しい PDF ファイルをオープンし、ディスク上に PDF を作成するためにファイル名を挿入します */
     if ($p->begin_document("", "") == 0) {
          die("Error: " . $p->get_errmsg());
    }

      $p->set_info("Creator", "hello.php");
      $p->set_info("Author", "Rainer Schaaf");
      $p->set_info("Title", "Hello world (PHP)!");

      $p->begin_page_ext(595, 842, "");

      $font = $p->load_font("Helvetica-Bold", "winansi", "");

      $p->setfont($font, 24.0);
      $p->set_text_pos(50, 700);
      $p->show("Hello world!");
      $p->continue_text("(says PHP)");
      $p->end_page_ext("");

      $p->end_document("");

      $buf = $p->get_buffer();
      $len = strlen($buf);

       header("Content-type: application/pdf");
       header("Content-Length: $len");
       header("Content-Disposition: inline; filename=hello.pdf");
      print $buf;
}
catch (PDFlibException $e) {
      die("PDFlib exception occurred in hello sample:¥n" .
          "[" . $e->get_errnum() . "] " . $e->get_apiname() . ": " .
          $e->get_errmsg() . "¥n");
}
catch (Exception $e) {
     die($e);
}
$p = 0;
?>
```

PDF_activate_item

(PECL pdflib:2.0-2.1.3)

PDF_activate_item — 構造体要素やその他の内容をアクティブにする

説明

bool **PDF_activate_item** (resource $pdfdoc , int $id)

事前に作成された構造体要素やその他の内容をアクティブにします。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_add_annotation

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_add_annotation — 注記を追加する [古い関数]

説明

この関数は非推奨です。かわりに PDF_create_annotation() を type=Text で使用してください。

PDF_add_bookmark

(PHP 4 >= 4.0.1, PECL pdflib:1.0-2.1.3)

PDF_add_bookmark — ブックマークを現在のページに追加する [古い関数]

説明

この関数は PDFlib バージョン 6 で廃止されました。かわりに PDF_create_bookmark() を使用してください。

PDF_add_launchlink

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_add_launchlink — 現在のページに起動用注記を追加する [古い関数]

説明

bool **PDF_add_launchlink** (resource $pdfdoc , float $llx , float $lly , float $urx , float $ury , string $filename)

web リソースへのリンクを追加します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに type=Launch として PDF_create_action()、そして type=Link として PDF_create_annotation() を使用してください。

PDF_add_locallink

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_add_locallink — 現在のページにリンク注記を追加する [古い関数]

説明

bool **PDF_add_locallink** (resource $pdfdoc , float $lowerleftx , float $lowerlefty , float $upperrightx , float $upperrighty , int $page , string $dest)

現在の PDF ファイルの中にターゲットへのリンク注記を追加します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに type=GoTo として PDF_create_action()、 そして type=Link として PDF_create_annotation() を使用してください。

PDF_add_nameddest

(PECL pdflib:2.0-2.1.3)

PDF_add_nameddest — 移動先を作成する

説明

bool **PDF_add_nameddest** (resource $pdfdoc , string $name , string $optlist)

現在のドキュメントの任意のページに移動先を作成します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_add_note

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_add_note — 現在のページに注記を追加する [古い関数]

説明

bool **PDF_add_note** (resource $pdfdoc , float $llx , float $lly , float $urx , float $ury , string $contents , string $title , string $icon , int $open)

現在のページに注記を追加します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに type=Text とともに PDF_create_annotation() を使用してください。

PDF_add_outline

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_add_outline — 現在のページにブックマークを追加する [古い関数]

説明

この関数は非推奨です。かわりに PDF_create_bookmark() を使用してください。

PDF_add_pdflink

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_add_pdflink — 現在のページにリンク注記を追加する [古い関数]

説明

bool **PDF_add_pdflink** (resource $pdfdoc , float $bottom_left_x , float $bottom_left_y , float $up_right_x , float $up_right_y , string $filename , int $page , string $dest)

PDF ターゲットにファイルリンク注記を追加します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに type=GoToR とともに PDF_create_action()、 および type=Link とともに PDF_create_annotation() を使用してください。

PDF_add_table_cell

(PECL pdflib:2.1.0-2.1.3)

PDF_add_table_cell — 新しいテーブル、あるいは既存のテーブルにセルを追加する

説明

int **PDF_add_table_cell** (resource $pdfdoc , int $table , int $column , int $row , string $text , string $optlist)

新しいテーブル、あるいは既存のテーブルにセルを追加します。

PDF_add_textflow

(PECL pdflib:2.1.0-2.1.3)

PDF_add_textflow — Textflow を作成するか、既存の Textflow にテキストを追加する

説明

int **PDF_add_textflow** (resource $pdfdoc , int $textflow , string $text , string $optlist)

Textflow オブジェクトを作成するか、既存の Textflow にテキストと明示的なオプションを追加します。

PDF_add_thumbnail

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_add_thumbnail — 現在のページにサムネイルを追加する

説明

bool **PDF_add_thumbnail** (resource $pdfdoc , int $image)

現在のページにサムネイルとして既存のイメージを追加します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_add_weblink

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_add_weblink — 現在のページに Web リンクを追加する [古い関数]

説明

bool **PDF_add_weblink** (resource $pdfdoc , float $lowerleftx , float $lowerlefty , float $upperrightx , float $upperrighty , string $url)

Web 上のターゲット url への Web リンクを追加します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに type=URI とともに PDF_create_action()、 および type=Link とともに PDF_create_annotation() を使用してください。

PDF_arc

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_arc — 反時計回りに円弧を描く

説明

bool **PDF_arc** (resource $p , float $x , float $y , float $r , float $alpha , float $beta)

反時計周りの円弧を追加します。

PDF_arcn

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_arcn — 時計回りに円弧を描く

説明

bool **PDF_arcn** (resource $p , float $x , float $y , float $r , float $alpha , float $beta)

描画の方向以外は、この関数の動作は pdf_arc() とまったく同じです。

PDF_attach_file

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_attach_file — 現在のページに添付ファイルを追加する [古い関数]

説明

bool **PDF_attach_file** (resource $pdfdoc , float $llx , float $lly , float $urx , float $ury , string $filename , string $description , string $author , string $mimetype , string $icon)

添付ファイル注記を追加します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに type=FileAttachment とともに PDF_create_annotation() を使用してください。

PDF_begin_document

(PECL pdflib:2.0-2.1.3)

PDF_begin_document — 新しい PDF ファイルを作成する

説明

int **PDF_begin_document** (resource $pdfdoc , string $filename , string $optlist)

さまざまなオプションをもとにして、新しい PDF ファイルを作成します。

PDF_begin_font

(PECL pdflib:2.0-2.1.3)

PDF_begin_font — Type 3 フォント定義を開始する

説明

bool **PDF_begin_font** (resource $pdfdoc , string $filename , float $a , float $b , float $c , float $d , float $e , float $f , string $optlist)

Type 3 フォント定義を開始します。

PDF_begin_glyph

(PECL pdflib:2.0-2.1.3)

PDF_begin_glyph — Type 3 フォントのグリフ定義を開始する

説明

bool **PDF_begin_glyph** (resource $pdfdoc , string $glyphname , float $wx , float $llx , float $lly , float $urx , float $ury)

Type 3 フォントのグリフ定義を開始します。

PDF_begin_item

(PECL pdflib:2.0-2.1.3)

PDF_begin_item — 構造体要素あるいはその他の内容をオープンする

説明

int **PDF_begin_item** (resource $pdfdoc , string $tag , string $optlist)

オプションで指定した属性を使用して、 構造体要素あるいはその他の内容をオープンします。

PDF_begin_layer

(PECL pdflib:2.0-2.1.3)

PDF_begin_layer — レイヤーを開始する

説明

bool ***PDF_begin_layer*** *(resource $pdfdoc , int $layer)*

その後のページ出力で使用するレイヤーを開始します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

この関数は、PDF 1.5 で使用可能です。

PDF_begin_page_ext

(PECL pdflib:2.0-2.1.3)

PDF_begin_page_ext — 新規ページを開始する

説明

bool ***PDF_begin_page_ext*** *(resource $pdfdoc , float $width , float $height , string $optlist)*

ドキュメントに新しいページを追加し、さまざまなオプションを設定します。 パラメータ width および height は、新しいページの大きさをポイント数で指定します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

一般的なページサイズのポイント数

名前	サイズ
A0	2380✗3368
A1	1684✗2380
A2	1190✗1684
A3	842✗1190
A4	595✗842
A5	421✗595
A6	297✗421
B5	501✗709
letter (8.5"✗11")	612✗792
legal (8.5"✗14")	612✗1008
ledger (17"✗11")	1224✗792
11"✗17"	792✗1224

PDF_begin_page

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_begin_page — 新規ページを開始する [古い関数]

説明

bool ***PDF_begin_page*** *(resource $pdfdoc , float $width , float $height)*

新しいページをドキュメントに追加します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

この関数は PDFlib バージョン 6 で廃止されました。かわりに PDF_begin_page_ext() を使用してください。

PDF_begin_pattern

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_begin_pattern — パターン定義を開始する

説明

int ***PDF_begin_pattern*** *(resource $pdfdoc , float $width , float $height , float $xstep , float $ystep , int $painttype)*

新規パターンの定義を開始します。

PDF_begin_template_ext

(PECL pdflib:2.1.0-2.1.3)

PDF_begin_template_ext — テンプレート定義を開始する

説明

int ***PDF_begin_template_ext*** *(resource $pdfdoc , float $width , float $height , string $optlist)*

新規テンプレートの定義を開始します。

PDF_begin_template

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_begin_template — テンプレート定義を開始する [古い関数]

説明

int ***PDF_begin_template*** *(resource $pdfdoc , float $width , float $height)*

新規テンプレートの定義を開始します。

この関数は PDFlib バージョン 7 で廃止されました。 かわりに PDF_begin_template_ext() を使用してください。

PDF_circle

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_circle — 円を描く

説明

bool ***PDF_circle*** *(resource $pdfdoc , float $x , float $y , float $r)*

円を追加します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

PDF_clip

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_clip — 現在のパスに切り取る

説明

bool ***PDF_clip*** *(resource $p)*

現在のパスをクリッピングパスとして使用し、パスを終了します。 成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

PDF_close_image

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_close_image — 画像を閉じる

説明

bool ***PDF_close_image*** *(resource $p , int $image)*

PDF_open_image() 関数から取得された image を閉じます。

PDF_close_pdi_page

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_close_pdi_page — ページハンドルを閉じる

説明

bool ***PDF_close_pdi_page*** *(resource $p , int $page)*

ページハンドルを閉じ、そのページに関係する全てのリソースを開放します。 成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

PDF_close_pdi

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_close_pdi — PDF ドキュメント入力を閉じる [古い関数]

説明

bool **PDF_close_pdi** (resource $p , int $doc)

全てのオープンされたページハンドルをクローズし、PDF ドキュメント入力 (PDI) を閉じます。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 7 で廃止されました。 かわりに **PDF_close_pdi_document()** を使用してください。

PDF_close

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_close — pdf ドキュメントを閉じる [古い関数]

説明

bool **PDF_close** (resource $p)

生成された PDF ファイルを閉じ、ドキュメントの関係する全てのリソースを 開放します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに PDF_end_document() を使用してください。

PDF_closepath_fill_stroke

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_closepath_fill_stroke — 現在のパスを閉じ、塗りつぶし、輪郭を描く

説明

bool **PDF_closepath_fill_stroke** (resource $p)

パスを閉じ、塗りつぶし、輪郭を描きます。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_closepath_stroke

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_closepath_stroke — パスを閉じ、パスに沿って線を描く

説明

bool **PDF_closepath_stroke** (resource $p)

パスを閉じ、その輪郭を描きます。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_closepath

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_closepath — 現在のパスを閉じる

説明

bool **PDF_closepath** (resource $p)

現在のパスを閉じます。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_concat

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_concat — 行列を CTM に追加する

説明

bool **PDF_concat** (resource $p , float $a , float $b , float $c , float $d , float $e , float $f)

行列を現在の変換行列 (Current Transformation Matrix: CTM) に追加します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_continue_text

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_continue_text — 次の行にテキストを出力する

説明

bool **PDF_continue_text** (resource $p , string $text)

次の行に text を出力します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_create_3dview

(PECL pdflib:2.1.0-2.1.3)

PDF_create_3dview — 3D ビューを作成する

説明

int **PDF_create_3dview** (resource $pdfdoc , string $username , string $optlist)

3D ビューを作成します。

この関数は PDF 1.6 を必要とします。

PDF_create_action

(PECL pdflib:2.0-2.1.3)

PDF_create_action — オブジェクトやイベントに対するアクションを作成する

説明

int **PDF_create_action** (resource $pdfdoc , string $type , string $optlist)

さまざまなオブジェクトやイベントに適用可能なアクションを作成します。

PDF_create_annotation

(PECL pdflib:2.0-2.1.3)

PDF_create_annotation — 矩形の注記を作成する

説明

bool **PDF_create_annotation** (resource $pdfdoc , float $llx , float $lly , float $urx , float $ury , string $type , string $optlist)

現在のページに矩形の注記を作成します。

PDF_create_bookmark

(PECL pdflib:2.0-2.1.3)

PDF_create_bookmark — ブックマークを作成する

説明

int **PDF_create_bookmark** (resource $pdfdoc , string $text , string $optlist)

さまざまなオプションをもとにしてブックマークを作成します。

PDF_create_field

(PECL pdflib:2.0-2.1.3)

PDF_create_field — フォームフィールドを作成する

説明

bool **PDF_create_field** (resource $pdfdoc , float $llx , float $lly , float $urx , float $ury , string $name , string $type , string $optlist)

さまざまなオプションをもとにして、現在のページ上にフォームフィールドを作成します。

PDF_create_fieldgroup

(PECL pdflib:2.0-2.1.3)

PDF_create_fieldgroup — フォームフィールドグループを作成する

説明

bool **PDF_create_fieldgroup** (resource $pdfdoc , string $name , string $optlist)

さまざまなオプションをもとにして、 フォームフィールドグループを作成します。

PDF_create_gstate

(PECL pdflib:2.0-2.1.3)

PDF_create_gstate — 画像状態オブジェクトを作成する

説明

int **PDF_create_gstate** (resource $pdfdoc , string $optlist)

さまざまなオプションをもとにして、画像状態オブジェクトを作成します。

PDF_create_pvf

(PECL pdflib:2.0-2.1.3)

PDF_create_pvf — PDFlib 仮想ファイルを作成する

説明

bool **PDF_create_pvf** (resource $pdfdoc , string $filename , string $data , string $optlist)

メモリ内のデータから、読み込み専用の仮想ファイルを作成します。

PDF_create_textflow

(PECL pdflib:2.0-2.1.3)

PDF_create_textflow — textflow オブジェクトを作成する

説明

int **PDF_create_textflow** (resource $pdfdoc , string $text , string $optlist)

後の書式設定で使用するテキストの前処理を行い、textflow オブジェクトを作成します。

PDF_curveto

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_curveto — ベジエ曲線を描く

説明

bool **PDF_curveto** (resource $p , float $x1 , float $y1 , float $x2 , float $y2 , float $x3 , float $y3)

3 つの制御点を使用して現在の点からベジエ曲線を描画します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_define_layer

(PECL pdflib:2.0-2.1.3)

PDF_define_layer — レイヤー定義を作成する

説明

int **PDF_define_layer** (resource $pdfdoc , string $name , string $optlist)

新しいレイヤー定義を作成します。

この関数は、PDF 1.5 で使用可能です。

PDF_delete_pvf

(PECL pdflib:2.0-2.1.3)

PDF_delete_pvf ― PDFlib 仮想ファイルを削除する

説明

*int **PDF_delete_pvf** (resource $pdfdoc , string $filename)*

仮想ファイルを削除し、そのデータ構造体を開放します（ファイルの中身は削除しません）。

PDF_delete_table

(PECL pdflib:2.1.0-2.1.3)

PDF_delete_table ― テーブルオブジェクトを削除する

説明

*bool **PDF_delete_table** (resource $pdfdoc , int $table , string $optlist)*

テーブルおよびそれに関連付けられたすべてのデータ構造を削除します。

PDF_delete_textflow

(PECL pdflib:2.0-2.1.3)

PDF_delete_textflow ― textflow オブジェクトを削除する

説明

*bool **PDF_delete_textflow** (resource $pdfdoc , int $textflow)*

textflow および関連付けられたデータ構造体を削除します。

PDF_delete

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_delete ― PDFlib オブジェクトを削除する

説明

*bool **PDF_delete** (resource $pdfdoc)*

*PDFlib オブジェクトを削除し、内部リソースを全て開放します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

PDF_encoding_set_char

(PECL pdflib:2.0-2.1.3)

PDF_encoding_set_char ― グリフ名や Unicode 値を追加する

説明

*bool **PDF_encoding_set_char** (resource $pdfdoc , string $encoding , int $slot , string $glyphname , int $uv)*

グリフ名や Unicode 値を、カスタムエンコーディングに追加します。

PDF_end_document

(PECL pdflib:2.0-2.1.3)

PDF_end_document ― PDF ファイルを閉じる

説明

*bool **PDF_end_document** (resource $pdfdoc , string $optlist)*

作成された PDF ファイルを閉じ、さまざまなオプションを適用します。

PDF_end_font

(PECL pdflib:2.0-2.1.3)

PDF_end_font — Type 3 フォント定義を終了する

説明

bool **PDF_end_font** (resource $pdfdoc)

Type 3 フォント定義を終了します。

PDF_end_glyph

(PECL pdflib:2.0-2.1.3)

PDF_end_glyph — Type 3 フォントのグリフ定義を終了する

説明

bool **PDF_end_glyph** (resource $pdfdoc)

Type 3 フォントのグリフ定義を終了します。

PDF_end_item

(PECL pdflib:2.0-2.1.3)

PDF_end_item — 構造体要素やその他の内容を閉じる

説明

bool **PDF_end_item** (resource $pdfdoc , int $id)

構造体要素やその他の内容を閉じます。

PDF_end_layer

(PECL pdflib:2.0-2.1.3)

PDF_end_layer — すべてのアクティブなレイヤーを無効にする

説明

bool **PDF_end_layer** (resource $pdfdoc)

すべてのアクティブなレイヤーを無効にします。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は、PDF 1.5 で使用可能です。

PDF_end_page_ext

(PECL pdflib:2.0-2.1.3)

PDF_end_page_ext — ページを終了する

説明

bool **PDF_end_page_ext** (resource $pdfdoc , string $optlist)

ページを終了し、さまざまなオプションを適用します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_end_page

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_end_page — ページを終了する

説明

bool **PDF_end_page** (resource $p)

ページを終了します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_end_pattern

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_end_pattern — パターンを終了する

説明

bool **PDF_end_pattern** (resource $p)

パターンの定義を終了します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_end_template

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_end_template — テンプレートを終了する

説明

bool **PDF_end_template** (resource $p)

テンプレートの定義を終了します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_endpath

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_endpath — 現在のパスを終了する

説明

bool **PDF_endpath** (resource $p)

塗りつぶしや描画を行わず、現在のパスを終了します。

PDF_fill_imageblock

(PECL pdflib:2.0-2.1.3)

PDF_fill_imageblock — 画像ブロックをさまざまなデータで塗りつぶす

説明

int **PDF_fill_imageblock** (resource $pdfdoc , int $page , string $blockname , int $image , string $optlist)

画像ブロックを、そのプロパティに基づいたさまざまなデータで塗りつぶします。

この関数は、PDFlib Personalization Server (PPS) でのみ使用可能です。

PDF_fill_pdfblock

(PECL pdflib:2.0-2.1.3)

PDF_fill_pdfblock — PDF ブロックをさまざまなデータで塗りつぶす

説明

int **PDF_fill_pdfblock** (resource $pdfdoc , int $page , string $blockname , int $contents , string $optlist)

PDF ブロックを、そのプロパティに基づいたさまざまなデータで塗りつぶします。

この関数は、PDFlib Personalization Server (PPS) でのみ使用可能です。

PDF_fill_stroke

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_fill_stroke — パスを塗りつぶし、パスの輪郭を描く

説明

bool **PDF_fill_stroke** (resource $p)

現在のパスの内部を塗りつぶし色で塗りつぶし、輪郭を 輪郭色で描画します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_fill_textblock

(PECL pdflib:2.0-2.1.3)

PDF_fill_textblock — テキストブロックをさまざまなデータで塗りつぶす

説明

int **PDF_fill_textblock** (resource $pdfdoc , int $page , string $blockname , string $text , string $optlist)

テキストブロックを、そのプロパティに基づいたさまざまなデータで塗りつぶします。

この関数は、PDFlib Personalization Server (PPS) でのみ使用可能です。

PDF_fill

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_fill — 現在のパスを塗りつぶす

説明

bool **PDF_fill** (resource $p)

現在のパスの内部を現在の塗りつぶし色で塗りつぶします。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_findfont

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_findfont — 後で使用するフォントを準備する [古い関数]

説明

int **PDF_findfont** (resource $p , string $fontname , string $encoding , int $embed)

後で PDF_setfont() で使用するフォントを検索し、準備します。 このフォントのメトリックがロードされ、embed がゼロでない場合はフォントファイルがチェックされますが、 ロードはまだ行われません。encoding は、 builtin、macroman、 winansi、host、 ユーザ定義のエンコーディング名あるいは CMap の名前のどれかとなります。 パラメータ embed は、PHP 4.3.5 以前あるいは PDFlib バージョン 5 未満ではオプションです。

この関数は PDFlib バージョン 5 で廃止されました。かわりに PDF_load_font() を使用してください。

PDF_fit_image

(PECL pdflib:2.0-2.1.3)

PDF_fit_image — 画像やテンプレートを配置する

説明

bool **PDF_fit_image** (resource $pdfdoc , int $image , float $x , float $y , string $optlist)

画像やテンプレートを、さまざまなオプションに基づいてページ上に配置します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_fit_pdi_page

(PECL pdflib:2.0-2.1.3)

PDF_fit_pdi_page — インポートした PDF ページを配置する

説明

bool **PDF_fit_pdi_page** (resource $pdfdoc , int $page , float $x , float $y , string $optlist)

さまざまなオプションに基づいて、インポートした PDF ページを配置します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_fit_table

(PECL pdflib:2.1.0-2.1.3)

PDF_fit_table — テーブルをページに配置する

説明

string **PDF_fit_table** (resource $pdfdoc , int $table , float $llx , float $lly , float $urx , float $ury , string $optlist)

テーブルを、ページ全体あるいは部分的に配置します。

PDF_fit_textflow

(PECL pdflib:2.0-2.1.3)

PDF_fit_textflow — textflow を矩形領域に配置する

説明

string **PDF_fit_textflow** (resource $pdfdoc , int $textflow , float $llx , float $lly , float $urx , float $ury , string $optlist)

textflow の次の部分を、矩形領域に配置します。

PDF_fit_textline

(PECL pdflib:2.0-2.1.3)

PDF_fit_textline — 1 行分のテキストを配置する

説明

bool **PDF_fit_textline** (resource $pdfdoc , string $text , float $x , float $y , string $optlist)

1 行分のテキストを、さまざまなオプションに基づいてページ上に配置します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_get_apiname

(PECL pdflib:2.0-2.1.3)

PDF_get_apiname — 成功しなかった API 関数の名前を取得する

説明

string **PDF_get_apiname** (resource $pdfdoc)

直近で例外を発生させたり失敗したりした API 関数の名前を取得します。

PDF_get_buffer

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_get_buffer — PDF 出力バッファを取得する

説明

string **PDF_get_buffer** (resource $p)

生成された PDF データを含むバッファの内容を取得します。

PDF_get_errmsg

(PECL pdflib:2.0-2.1.3)

PDF_get_errmsg — エラーテキストを取得する

説明

string **PDF_get_errmsg** (resource $pdfdoc)

直近で発生した例外のテキストや、関数コールが失敗した理由の テキストを取得します。

PDF_get_errnum

(PECL pdflib:2.0-2.1.3)

PDF_get_errnum — エラー番号を取得する

説明

int **PDF_get_errnum** (resource $pdfdoc)

直近で発生した例外の番号や、失敗した関数コールの理由を表す番号を取得します。

PDF_get_font

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_get_font — フォントを取得する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。 かわりにパラメータ font を用いて PDF_get_value() を使用してください。

PDF_get_fontname

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_get_fontname — フォント名を取得する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。 かわりにパラメータ fontname を用いて PDF_get_parameter() を使用してください。

PDF_get_fontsize

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_get_fontsize — フォント処理 [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。 かわりにパラメータ fontsize を用いて PDF_get_value() を使用してください。

PDF_get_image_height

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_get_image_height — 画像の高さを取得する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。 かわりにパラメータ imageheight を用いて PDF_get_value() を使用してください。

PDF_get_image_width

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_get_image_width — 画像の幅を取得する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。 かわりにパラメータ imagewidth を用いて PDF_get_value() を使用してください。

PDF_get_majorversion

(PHP 4 >= 4.2.0, PECL pdflib:1.0-2.1.3)

PDF_get_majorversion — メジャーバージョン番号を取得する [古い関数]

説明

int **PDF_get_majorversion** (void)

この関数は PDFlib バージョン 5 で廃止されました。 かわりにパラメータ major を用いて PDF_get_value() を使用してください。

PDF_get_minorversion

(PHP 4 >= 4.2.0, PECL pdflib:1.0-2.1.3)

PDF_get_minorversion — マイナーバージョン番号を取得する [古い関数]

説明

int **PDF_get_minorversion** (void)

PDFlib のマイナーバージョン番号を返します。

この関数は PDFlib バージョン 5 で廃止されました。 かわりにパラメータ minor を用いて PDF_get_value() を使用してください。

PDF_get_parameter

(PHP 4 >= 4.0.1, PECL pdflib:1.0-2.1.3)

PDF_get_parameter — 文字列パラメータを取得する

説明

string **PDF_get_parameter** (resource $p , string $key , float $modifier)

いくつかの PDFlib パラメータの内容を文字列として取得します。

PDF_get_pdi_parameter

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_get_pdi_parameter — PDI 文字列パラメータを取得する [古い関数]

説明

string **PDF_get_pdi_parameter** (resource $p , string $key , int $doc , int $page , int $reserved)

文字列型の PDI ドキュメントパラメータの内容を取得します。

この関数は PDFlib バージョン 7 で廃止されました。 かわりに PDF_pcos_get_string() を使用してください。

PDF_get_pdi_value

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_get_pdi_value — 数値型の PDI パラメータを取得する [古い関数]

説明

float **PDF_get_pdi_value** (resource $p , string $key , int $doc , int $page , int $reserved)

数値型を有する PDI ドキュメントパラメータの内容を取得します。

この関数は PDFlib バージョン 7 で廃止されました。 かわりに PDF_pcos_get_number() を使用してください。

PDF_get_value

(PHP 4 >= 4.0.1, PECL pdflib:1.0-2.1.3)

PDF_get_value — 数値型のパラメータを取得する

説明

float **PDF_get_value** (resource $p , string $key , float $modifier)

いくつかの PDFlib パラメータを数値型で取得します。

PDF_info_font

(PECL pdflib:2.1.0-2.1.3)

PDF_info_font — 読み込まれたフォントについての詳細情報を問い合わせる

説明

float **PDF_info-font** (resource $pdfdoc , int $font , string $keyword , string $optlist)

読み込まれたフォントについての詳細情報を問い合わせます。

PDF_info_matchbox

(PECL pdflib:2.1.0-2.1.3)

PDF_info_matchbox — マッチボックスの情報を問い合わせる

説明

float **PDF_info_matchbox** (resource $pdfdoc , string $boxname , int $num , string $keyword)

現在のページ上のマッチボックスについての情報を問い合わせます。

PDF_info_table

(PECL pdflib:2.1.0-2.1.3)

PDF_info_table — テーブルの情報を取得する

説明

float **PDF_info_table** (resource $pdfdoc , int $table , string $keyword)

直近に配置されたテーブルインスタンスについてのテーブルの情報を取得します。

PDF_info_textflow

(PECL pdflib:2.0-2.1.3)

PDF_info_textflow — textflow の状態を問い合わせる

説明

float **PDF_info_textflow** (resource $pdfdoc , int $textflow , string $keyword)

textflow の現在の状態を問い合わせます。

PDF_info_textline

(PECL pdflib:2.1.0-2.1.3)

PDF_info_textline — テキストの行のフォーマットを行い、メトリクスを問い合わせる

説明

float **PDF_info_textline** (resource $pdfdoc , string $text , string $keyword , string $optlist)

テキストの行のフォーマットを行い、その結果のメトリクスを問い合わせます。

PDF_initgraphics

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_initgraphics — 描画状態をリセットする

説明

bool **PDF_initgraphics** (resource $p)

色および描画状態パラメータを全てデフォルト値にリセットします。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_lineto

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_lineto — 線を描く

説明

bool **PDF_lineto** (resource $p , float $x , float $y)

現在の点から別の点まで線を描画します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_load_3ddata

(PECL pdflib:2.1.0-2.1.3)

PDF_load_3ddata — 3D モデルを読み込む

説明

int ***PDF_load_3ddata*** *(resource $pdfdoc , string $filename , string $optlist)*

3D モデルを、ディスク上あるいは仮想ファイルから読み込みます。

この関数は PDF 1.6 を必要とします。

PDF_load_font

(PECL pdflib:2.0-2.1.3)

PDF_load_font — フォントを検索し、準備する

説明

int ***PDF_load_font*** *(resource $pdfdoc , string $fontname , string $encoding , string $optlist)*

フォントを検索し、後に使用するために準備します。

PDF_load_iccprofile

(PECL pdflib:2.0-2.1.3)

PDF_load_iccprofile — ICC プロファイルを検索し、準備する

説明

int ***PDF_load_iccprofile*** *(resource $pdfdoc , string $profilename , string $optlist)*

ICC プロファイルを検索し、後に使用するために準備します。

PDF_load_image

(PECL pdflib:2.0-2.1.3)

PDF_load_image — 画像ファイルをオープンする

説明

int ***PDF_load_image*** *(resource $pdfdoc , string $imagetype , string $filename , string $optlist)*

さまざまなオプションをもとに、 ディスク上の画像ファイルや仮想画像ファイルをオープンします。

PDF_makespotcolor

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_makespotcolor — スポット色を作成する

説明

int ***PDF_makespotcolor*** *(resource $p , string $spotname)*

組み込みのスポット色名を検索するか、 現在の塗りつぶし色から名前付きのスポット色を作成します。 成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

PDF_moveto

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_moveto — 現在の位置を設定する

説明

bool ***PDF_moveto*** *(resource $p , float $x , float $y)*

描画出力のための現在の点を設定します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

PDF_new

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_new — PDFlib オブジェクトを作成する

説明

resource **PDF_new** ($)

新規に PDFlib オブジェクトを作成します。

PDF_open_ccitt

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_open_ccitt — raw CCITT イメージをオープンする [古い関数]

説明

int **PDF_open_ccitt** (resource $pdfdoc , string $filename , int $width , int $height , int $BitReverse , int $k , int $Blackls1)

raw CCITT イメージをオープンします。

この関数は PDFlib バージョン 5 で廃止されました。かわりに PDF_load_image() を使用してください。

PDF_open_file

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_open_file — PDF ファイルを作成する [古い関数]

説明

bool **PDF_open_file** (resource $p , string $filename)

新規の PDF ファイルを指定したファイル名で作成します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに PDF_begin_document() を使用してください。

PDF_open_gif

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_open_gif — GIF イメージをオープンする [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_load_image() を使用してください。

PDF_open_image_file

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_open_image_file — ファイルからイメージを読み込む [古い関数]

説明

int **PDF_open_image_file** (resource $p , string $imagetype , string $filename , string $stringparam , int $intparam)

イメージファイルをオープンします。

この関数は PDFlib バージョン 5 で廃止されました。かわりに colorize、ignoremask、invert、mask、masked および page を指定して PDF_load_image() を使用してください。

PDF_open_image

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_open_image — イメージデータを使用する [古い関数]

説明

int **PDF_open_image** (resource $p , string $imagetype , string $source , string $data , int $length , int $width , int $height , int $components , int $bpc , string $params)

さまざまなデータソースからのイメージデータを使用します。

この関数は PDFlib バージョン 5 で廃止されました。かわりに 仮想ファイルおよび PDF_load_image() を使用してください。

PDF_open_jpeg

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_open_jpeg — JPEG イメージをオープンする [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_load_image() を使用してください。

PDF_open_memory_image

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_open_memory_image — PHP のイメージ関数で作成されたイメージをオープンする [未サポート]

説明

int **PDF_open_memory_image** (resource $p , resource $image)

この関数は PDFlib GmbH でサポートされていません。

PDF_open_pdi_page

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_open_pdi_page — ページを準備する

説明

int **PDF_open_pdi_page** (resource $p , int $doc , int $pagenumber , string $optlist)

PDF_fit_pdi_image() で後で使用するようにページを準備します。

PDF_open_pdi

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_open_pdi — PDF ファイルをオープンする [古い関数]

説明

int **PDF_open_pdi** (resource $pdfdoc , string $filename , string $optlist , int $len)

後で使用するため、ディスクベースかあるいは仮想 PDF ドキュメントをオープンします。

この関数は PDFlib バージョン 7 で廃止されました。 かわりに **PDF_open_pdi_document()** を使用してください。

PDF_open_tiff

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_open_tiff — TIFF イメージをオープンする [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_load_image() を使用してください。

PDF_pcos_get_number

(PECL pdflib:2.1.0-2.1.3)

PDF_pcos_get_number — number あるいは boolean 型の pCOS パスの値を取得する

説明

float **PDF_pcos_get_number** (resource $p , int $doc , string $path)

number あるいは boolean 型の pCOS パスの値を取得します。

PDF_pcos_get_stream

(PECL pdflib:2.1.0-2.1.3)

PDF_pcos_get_stream — stream、fstream あるいは string 型の pCOS パスの内容を取得する

説明

string **PDF_pcos_get_stream** (resource $p , int $doc , string $optlist , string $path)

stream、fstream あるいは string 型の pCOS パスの内容を取得します。

PDF_pcos_get_string

(PECL pdflib:2.1.0-2.1.3)

PDF_pcos_get_string — name、string あるいは boolean 型の pCOS パスの値を取得する

説明

string **PDF_pcos_get_string** (resource $p , int $doc , string $path)

name, string あるいは boolean 型の pCOS パスの値を取得します。

PDF_place_image

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_place_image — イメージをページ上に置く [古い関数]

説明

bool **PDF_place_image** (resource $pdfdoc , int $image , float $x , float $y , float $scale)

イメージを置き、サイズを変更します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 5 で廃止されました。かわりに PDF_fit_image() を使用してください。

PDF_place_pdi_page

(PHP 4 >= 4.0.6, PECL pdflib:1.0-2.1.3)

PDF_place_pdi_page — PDF ページを置く [古い関数]

説明

bool **PDF_place_pdi_page** (resource $pdfdoc , int $page , float $x , float $y , float $sx , float $sy)

PDI ページを配置し、サイズを変更します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 5 で廃止されました。かわりに PDF_fit_pdi_page() を使用してください。

PDF_process_pdi

(PECL pdflib:2.0-2.1.3)

PDF_process_pdi — インポートされた PDF ドキュメントを処理する

説明

int **PDF_process_pdi** (resource $pdfdoc , int $doc , int $page , string $optlist)

インポートされた PDF ドキュメントの要素を処理します。

PDF_rect

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_rect — 矩形を描く

説明

bool **PDF_rect** (resource $p , float $x , float $y , float $width , float $height)

矩形を描画します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_restore

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_restore — 描画状態を復元する

説明

bool **PDF_restore** (resource $p)

直近に保存された描画状態を復元します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_resume_page

(PECL pdflib:2.0-2.1.3)

PDF_resume_page — ページを再開する

説明

bool **PDF_resume_page** (resource $pdfdoc , string $optlist)

ページを再開し、さらに内容を追加します。

PDF_rotate

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_rotate — 座標系を回転する

説明

bool **PDF_rotate** (resource $p , float $phi)

座標系を回転します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_save

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_save — 描画状態を保存する

説明

bool **PDF_save** (resource $p)

現在の描画状態を保存します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_scale

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_scale — スケールを設定する

説明

bool **PDF_scale** (resource $p , float $sx , float $sy)

座表系のスケールを設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_set_border_color

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_border_color — 注記の周りの境界色を設定する [古い関数]

説明

bool **PDF_set_border_color** (resource $p , float $red , float $green , float $blue)

全ての種類の注記の境界色を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに PDF_create_annotation() でオプション annotcolor を使用してください。

PDF_set_border_dash

(PHP 4 >= 4.0.1, PECL pdflib:1.0-2.1.3)

PDF_set_border_dash — 注記の周りの境界の破線形式を設定する [古い関数]

説明

bool **PDF_set_border_dash** (resource $pdfdoc , float $black , float $white)

全ての種類の注記の境界の破線形式を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに PDF_create_annotation() でオプション dasharray を使用してください。

PDF_set_border_style

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_border_style — 注記の周りの境界の形式を設定する [古い関数]

説明

bool **PDF_set_border_style** (resource $pdfdoc , string $style , float $width)

全ての種類の注記の境界の形式を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 6 で廃止されました。かわりに PDF_create_annotation() でオプション borderstyle および linewidth を使用してください。

PDF_set_char_spacing

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_char_spacing — 文字間隔を設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_value() でパラメータ charspacing を使用してください。

PDF_set_duration

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_duration — ページ間隔を設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに duration オプションを PDF_begin_page_ext() あるいは PDF_end_page_ext() で使用してください。

PDF_set_gstate

(PECL pdflib:2.0-2.1.3)

PDF_set_gstate — 画像状態オブジェクトをアクティブにする

説明

bool **PDF_set_gstate** (resource $pdfdoc , int $gstate)

画像状態オブジェクトをアクティブにします。

PDF_set_horiz_scaling

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_horiz_scaling — テキストの横方向倍率を設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_value() でパラメータ horizscaling を使用してください。

PDF_set_info_author

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_info_author — ドキュメントの author フィールドを設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_info() を使用してください。

PDF_set_info_creator

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_info_creator — ドキュメントの creator フィールドを設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_info() を使用してください。

PDF_set_info_keywords

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_info_keywords — ドキュメントの keyword フィールドを設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_info() を使用してください。

PDF_set_info_subject

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_info_subject — ドキュメントの subject フィールドを設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_info() を使用してください。

PDF_set_info_title

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_info_title — ドキュメントの title フィールドを設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_info() を使用してください。

PDF_set_info

(PHP 4 >= 4.0.1, PECL pdflib:1.0-2.1.3)

PDF_set_info — ドキュメント情報のフィールドを設定する

説明

bool **PDF_set_info** (resource $p , string $key , string $value)

ドキュメント情報フィールド key に value を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_set_layer_dependency

(PECL pdflib:2.0-2.1.3)

PDF_set_layer_dependency — レイヤー間の関係を定義する

説明

bool ***PDF_set_layer_dependency*** *(resource $pdfdoc , string $type , string $optlist)*

レイヤー間の階層やグループ関係を定義します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

この関数は、PDF 1.5 で使用可能です。

PDF_set_leading

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_leading — テキストの行間を設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_value() で パラメータ leading を使用してください。

PDF_set_parameter

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_parameter — 文字列パラメータを設定する

説明

bool ***PDF_set_parameter*** *(resource $p , string $key , string $value)*

PDFlib パラメータを文字列型で設定します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

PDF_set_text_matrix

(PHP 4 <= 4.0.4)

PDF_set_text_matrix — テキストの行列を設定する [古い関数]

説明

PDFlib バージョン 3 以降、この関数は非推奨です。かわりに PDF_scale()、PDF_translate()、 PDF_rotate() あるいは PDF_skew() を使用してください。

PDF_set_text_pos

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_text_pos — テキストの位置を設定する

説明

bool ***PDF_set_text_pos*** *(resource $p , float $x , float $y)*

テキストの出力位置を設定します。成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

PDF_set_text_rendering

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_text_rendering — テキストの描画方法を設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_value() でパラメータ textrendering を使用してください。

PDF_set_text_rise

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_text_rise — テキストの傾きを設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_value() でパラメータ textrise を使用してください。

PDF_set_value

(PHP 4 >= 4.0.1, PECL pdflib:1.0-2.1.3)

PDF_set_value — 数値パラメータを設定する

説明

bool **PDF_set_value** (resource $p , string $key , float $value)

PDFlib パラメータの値を数値型で設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_set_word_spacing

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_set_word_spacing — 単語間の空白を設定する [古い関数]

説明

この関数は PDFlib バージョン 3 で廃止されました。かわりに PDF_set_value() でパラメータ wordspacing を使用してください。

PDF_setcolor

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_setcolor — 塗りつぶし色および輪郭色を設定する

説明

bool **PDF_setcolor** (resource $p , string $fstype , string $colorspace , float $c1 , float $c2 , float $c3 , float $c4)

現在の色空間と色を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setdash

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setdash — 破線パターンを設定する

説明

bool **PDF_setdash** (resource $pdfdoc , float $b , float $w)

現在のダッシュパターンを黒 b および白 w 単位で設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setdashpattern

(PECL pdflib:2.0-2.1.3)

PDF_setdashpattern — 破線パターンを設定する

説明

bool **PDF_setdashpattern** (resource $pdfdoc , string $optlist)

オプションリストで定義した破線パターンを設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setflat

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setflat — 平面度を設定する

説明

bool **PDF_setflat** (resource $pdfdoc , float $flatness)

平面度を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setfont

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_setfont — フォントを設定する

説明

bool **PDF_setfont** (resource $pdfdoc , int $font , float $fontsize)

PDF_findfont() から返された font ハンドルを使用し、指定した size で現在のフォントを設定します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setgray_fill

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setgray_fill — 塗りつぶし色をグレーに設定する [古い関数]

説明

bool **PDF_setgray_fill** (resource $p , float $g)

現在の塗りつぶし色を 0 から 1 までのグレー値に設定します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 4 で廃止されました。かわりに PDF_setcolor() を使用してください。

PDF_setgray_stroke

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setgray_stroke — 描画色をグレーに設定する [古い関数]

説明

bool **PDF_setgray_stroke** (resource $p , float $g)

現在の輪郭描画色を 0 から 1 までのグレー値に設定します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 4 で廃止されました。かわりに PDF_setcolor() を使用してください。

PDF_setgray

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setgray — 色をグレーに設定する [古い関数]

説明

bool **PDF_setgray** (resource $p , float $g)

現在の塗りつぶしおよび輪郭描画色を 0 から 1 までのグレー値に設定します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 4 で廃止されました。かわりに PDF_setcolor() を使用してください。

PDF_setlinecap

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setlinecap — linecap パラメータを設定する

説明

bool **PDF_setlinecap** (resource $p , int $linecap)

linecap を設定し、 ストロークを考慮したうえでパスの終端の形状を制御します。

PDF_setlinejoin

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setlinejoin — linejoin パラメータを設定する

説明

bool **PDF_setlinejoin** (resource $p , int $value)

linejoin パラメータを設定し、 パスを描画する際の角の形状を指定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setlinewidth

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setlinewidth — 線幅を設定する

説明

bool **PDF_setlinewidth** (resource $p , float $width)

現在の線幅を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setmatrix

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_setmatrix — 現在の変換行列を設定する

説明

bool **PDF_setmatrix** (resource $p , float $a , float $b , float $c , float $d , float $e , float $f)

現在の変換行列を明示的に設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setmiterlimit

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setmiterlimit — miter limit を設定する

説明

bool **PDF_setmiterlimit** (resource $pdfdoc , float $miter)

miter limit を設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_setpolydash

(PHP 4 >= 4.0.5, PECL pdflib:1.0-2.1.3)

PDF_setpolydash — 複雑な破線パターンを設定する [古い関数]

説明

この関数は PDFlib バージョン 5 で廃止されました。かわりに PDF_setdashpattern() を使用してください。

PDF_setrgbcolor_fill

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setrgbcolor_fill — 塗りつぶし RGB 色の値を設定する

説明

bool **PDF_setrgbcolor_fill** (resource $p , float $red , float $green , float $blue)

現在の塗りつぶし色を指定した RGB 値に設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 4 で廃止されました。かわりに PDF_setcolor() を使用してください。

PDF_setrgbcolor_stroke

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setrgbcolor_stroke — 描画 RGB 色を設定する [古い関数]

説明

bool **PDF_setrgbcolor_stroke** (resource $p , float $red , float $green , float $blue)

現在の輪郭色を指定した RGB 値に設定します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 4 で廃止されました。かわりに PDF_setcolor() を使用してください。

PDF_setrgbcolor

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_setrgbcolor — 描画および塗りつぶし RGB 色を設定する [古い関数]

説明

bool **PDF_setrgbcolor** (resource $p , float $red , float $green , float $blue)

現在の塗りつぶし色と輪郭色を指定した RGB 値に設定します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

この関数は PDFlib バージョン 4 で廃止されました。かわりに PDF_setcolor() を使用してください。

PDF_shading_pattern

(PECL pdflib:2.0-2.1.3)

PDF_shading_pattern — シェーディングパターンを定義する

説明

int **PDF_shading_pattern** (resource $pdfdoc , int $shading , string $optlist)

シェーディングオブジェクトを使用して、シェーディングパターンを定義します。

この関数は、PDF 1.4 以降で使用可能です。

PDF_shading

(PECL pdflib:2.0-2.1.3)

PDF_shading — 混色を定義する

説明

int **PDF_shading** (resource $pdfdoc , string $shtype , float $x0 , float $y0 , float $x1 , float $y1 , float $c1 , float $c2 , float $c3 , float $c4 , string $optlist)

現在の塗りつぶし色と他の色との混色を定義します。

この関数は、PDF 1.4 以降で使用可能です。

PDF_shfill

(PECL pdflib:2.0-2.1.3)

PDF_shfill — シェーディングで領域を塗りつぶす

説明

bool **PDF_shfill** (resource $pdfdoc , int $shading)

シェーディングオブジェクトに基づいたシェーディングで、 領域を塗りつぶします。

この関数は、PDF 1.4 以降で使用可能です。

PDF_show_boxed

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_show_boxed — ボックスにテキストを出力する [古い関数]

説明

int **PDF_show_boxed** (resource $p , string $text , float $left , float $top , float $width , float $height , string $mode , string $feature)

この関数は PDFlib バージョン 6 で廃止されました。かわりに 単一行の場合は PDF_fit_textline()、 複数行の場合は **PDF_*_textflow()** 関数を 使用してください。

PDF_show_xy

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_show_xy — 指定した位置にテキストを出力する

説明

bool **PDF_show_xy** (resource $p , string $text , float $x , float $y)

現在のフォントで text を出力します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_show

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_show — 現在の位置にテキストを出力する

説明

bool **PDF_show** (resource $pdfdoc , string $text)

現在の位置に現在のフォントとサイズで text を出力します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_skew

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_skew — 座標系を歪ませる

説明

bool **PDF_skew** (resource $p , float $alpha , float $beta)

座標系を x 方向および y 方向に それぞれ alpha 度、beta 度歪ませます。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_stringwidth

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_stringwidth — テキストの幅を返す

説明

float **PDF_stringwidth** (resource $p , string $text , int $font , float $fontsize)

任意のフォントにおける text の幅を返します。

PDF_stroke

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_stroke — パスを描く

説明

bool **PDF_stroke** (resource $p)

現在の色および線幅でパスの輪郭を描画し、パスをクリアします。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDF_suspend_page

(PECL pdflib:2.0-2.1.3)

PDF_suspend_page — ページを停止する

説明

bool **PDF_suspend_page** (resource $pdfdoc , string $optlist)

現在のページを一時停止します。後で PDF_resume_page() で再開することが可能です。

PDF_translate

(PHP 4, PECL pdflib:1.0-2.1.3)

PDF_translate — 座標系の原点を設定する

説明

bool **PDF_translate** (resource $p , float $tx , float $ty)

座標系の原点を変更します。

PDF_utf16_to_utf8

(PECL pdflib:2.0.3-2.1.3)

PDF_utf16_to_utf8 — 文字列を UTF-16 から UTF-8 に変換する

説明

string **PDF_utf16_to_utf8** (resource $pdfdoc , string $utf16string)

文字列を UTF-16 フォーマットから UTF-8 に変換します。

PDF_utf32_to_utf16

(No version information available, might be only in CVS)

PDF_utf32_to_utf16 — 文字列を UTF-32 から UTF-16 に変換する

説明

string **PDF_utf32_to_utf16** (resource $pdfdoc , string $utf32string , string $ordering)

文字列を UTF-32 フォーマットから UTF-16 に変換します。

PDF_utf8_to_utf16

(PECL pdflib:2.0.3-2.1.3)

PDF_utf8_to_utf16 — 文字列を UTF-8 から UTF-16 に変換する

説明

string **PDF_utf8_to_utf16** (resource $pdfdoc , string $utf8string , string $ordering)

文字列を UTF-8 フォーマットから UTF-16 に変換します。

目次

- PDF_begin_template_ext — テンプレート定義を開始する
- PDF_begin_template — テンプレート定義を開始する [古い関数]
- PDF_circle — 円を描く
- PDF_clip — 現在のパスに切り取る
- PDF_close_image — 画像を閉じる
- PDF_close_pdi_page — ページハンドルを閉じる
- PDF_close_pdi — PDF ドキュメント入力を閉じる [古い関数]
- PDF_close — pdf ドキュメントを閉じる [古い関数]
- PDF_closepath_fill_stroke — 現在のパスを閉じ、塗りつぶし、輪郭を描く
- PDF_closepath_stroke — パスを閉じ、パスに沿って線を描く
- PDF_closepath — 現在のパスを閉じる
- PDF_concat — 行列を CTM に追加する
- PDF_continue_text — 次の行にテキストを出力する
- PDF_create_3dview — 3D ビューを作成する
- PDF_create_action — オブジェクトやイベントに対するアクションを作成する
- PDF_create_annotation — 矩形の注記を作成する
- PDF_create_bookmark — ブックマークを作成する
- PDF_create_field — フォームフィールドを作成する
- PDF_create_fieldgroup — フォームフィールドグループを作成する
- PDF_create_gstate — 画像状態オブジェクトを作成する
- PDF_create_pvf — PDFlib 仮想ファイルを作成する
- PDF_create_textflow — textflow オブジェクトを作成する
- PDF_curveto — ベジエ曲線を描く
- PDF_define_layer — レイヤー定義を作成する
- PDF_delete_pvf — PDFlib 仮想ファイルを削除する
- PDF_delete_table — テーブルオブジェクトを削除する
- PDF_delete_textflow — textflow オブジェクトを削除する
- PDF_delete — PDFlib オブジェクトを削除する
- PDF_encoding_set_char — グリフ名や Unicode 値を追加する
- PDF_end_document — PDF ファイルを閉じる
- PDF_end_font — Type 3 フォント定義を終了する
- PDF_end_glyph — Type 3 フォントのグリフ定義を終了する
- PDF_end_item — 構造体要素やその他の内容を閉じる
- PDF_end_layer — すべてのアクティブなレイヤーを無効にする
- PDF_end_page_ext — ページを終了する
- PDF_end_page — ページを終了する
- PDF_end_pattern — パターンを終了する
- PDF_end_template — テンプレートを終了する
- PDF_endpath — 現在のパスを終了する
- PDF_fill_imageblock — 画像ブロックをさまざまなデータで塗りつぶす
- PDF_fill_pdfblock — PDF ブロックをさまざまなデータで塗りつぶす
- PDF_fill_stroke — パスを塗りつぶし、パスの輪郭を描く
- PDF_fill_textblock — テキストブロックをさまざまなデータで塗りつぶす
- PDF_fill — 現在のパスを塗りつぶす
- PDF_findfont — 後で使用するフォントを準備する [古い関数]
- PDF_fit_image — 画像やテンプレートを配置する
- PDF_fit_pdi_page — インポートした PDF ページを配置する
- PDF_fit_table — テーブルをページに配置する
- PDF_fit_textflow — textflow を矩形領域に配置する
- PDF_fit_textline — 1 行分のテキストを配置する
- PDF_get_apiname — 成功しなかった API 関数の名前を取得する
- PDF_get_buffer — PDF 出力バッファを取得する
- PDF_get_errmsg — エラーテキストを取得する
- PDF_get_errnum — エラー番号を取得する
- PDF_get_font — フォントを取得する [古い関数]
- PDF_get_fontname — フォント名を取得する [古い関数]
- PDF_get_fontsize — フォント処理 [古い関数]
- PDF_get_image_height — 画像の高さを取得する [古い関数]
- PDF_get_image_width — 画像の幅を取得する [古い関数]
- PDF_get_majorversion — メジャーバージョン番号を取得する [古い関数]
- PDF_get_minorversion — マイナーバージョン番号を取得する [古い関数]
- PDF_get_parameter — 文字列パラメータを取得する
- PDF_get_pdi_parameter — PDI 文字列パラメータを取得する [古い関数]
- PDF_get_pdi_value — 数値型の PDI パラメータを取得する [古い関数]

- PDF_get_value — 数値型のパラメータを取得する
- PDF_info_font — 読み込まれたフォントについての詳細情報を問い合わせる
- PDF_info_matchbox — マッチボックスの情報を問い合わせる
- PDF_info_table — テーブルの情報を取得する
- PDF_info_textflow — textflow の状態を問い合わせる
- PDF_info_textline — テキストの行のフォーマットを行い、メトリクスを問い合わせる
- PDF_initgraphics — 描画状態をリセットする
- PDF_lineto — 線を描く
- PDF_load_3ddata — 3D モデルを読み込む
- PDF_load_font — フォントを検索し、準備する
- PDF_load_iccprofile — ICC プロファイルを検索し、準備する
- PDF_load_image — 画像ファイルをオープンする
- PDF_makespotcolor — スポット色を作成する
- PDF_moveto — 現在の位置を設定する
- PDF_new — PDFlib オブジェクトを作成する
- PDF_open_ccitt — raw CCITT イメージをオープンする [古い関数]
- PDF_open_file — PDF ファイルを作成する [古い関数]
- PDF_open_gif — GIF イメージをオープンする [古い関数]
- PDF_open_image_file — ファイルからイメージを読み込む [古い関数]
- PDF_open_image — イメージデータを使用する [古い関数]
- PDF_open_jpeg — JPEG イメージをオープンする [古い関数]
- PDF_open_memory_image — PHP のイメージ関数で作成されたイメージをオープンする [未サポート]
- PDF_open_pdi_page — ページを準備する
- PDF_open_pdi — PDF ファイルをオープンする [古い関数]
- PDF_open_tiff — TIFF イメージをオープンする [古い関数]
- PDF_pcos_get_number — number あるいは boolean 型の pCOS パスの値を取得する
- PDF_pcos_get_stream — stream、fstream あるいは string 型の pCOS パスの内容を取得する
- PDF_pcos_get_string — name、string あるいは boolean 型の pCOS パスの値を取得する
- PDF_place_image — イメージをページ上に置く [古い関数]
- PDF_place_pdi_page — PDF ページを置く [古い関数]
- PDF_process_pdi — インポートされた PDF ドキュメントを処理する
- PDF_rect — 矩形を描く
- PDF_restore — 描画状態を復元する
- PDF_resume_page — ページを再開する
- PDF_rotate — 座標系を回転する
- PDF_save — 描画状態を保存する
- PDF_scale — スケールを設定する
- PDF_set_border_color — 注記の周りの境界色を設定する [古い関数]
- PDF_set_border_dash — 注記の周りの境界の破線形式を設定する [古い関数]
- PDF_set_border_style — 注記の周りの境界の形式を設定する [古い関数]
- PDF_set_char_spacing — 文字間隔を設定する [古い関数]
- PDF_set_duration — ページ間隔を設定する [古い関数]
- PDF_set_gstate — 画像状態オブジェクトをアクティブにする
- PDF_set_horiz_scaling — テキストの横方向倍率を設定する [古い関数]
- PDF_set_info_author — ドキュメントの author フィールドを設定する [古い関数]
- PDF_set_info_creator — ドキュメントの creator フィールドを設定する [古い関数]
- PDF_set_info_keywords — ドキュメントの keyword フィールドを設定する [古い関数]
- PDF_set_info_subject — ドキュメントの subject フィールドを設定する [古い関数]
- PDF_set_info_title — ドキュメントの title フィールドを設定する [古い関数]
- PDF_set_info — ドキュメント情報のフィールドを設定する
- PDF_set_layer_dependency — レイヤー間の関係を定義する
- PDF_set_leading — テキストの行間を設定する [古い関数]
- PDF_set_parameter — 文字列パラメータを設定する
- PDF_set_text_matrix — テキストの行列を設定する [古い関数]
- PDF_set_text_pos — テキストの位置を設定する
- PDF_set_text_rendering — テキストの描画方法を設定する [古い関数]
- PDF_set_text_rise — テキストの傾きを設定する [古い関数]
- PDF_set_value — 数値パラメータを設定する
- PDF_set_word_spacing — 単語間の空白を設定する [古い関数]
- PDF_setcolor — 塗りつぶし色および輪郭色を設定する
- PDF_setdash — 破線パターンを設定する
- PDF_setdashpattern — 破線パターンを設定する
- PDF_setflat — 平面度を設定する
- PDF_setfont — フォントを設定する

- PDF_setgray_fill — 塗りつぶし色をグレーに設定する [古い関数]
- PDF_setgray_stroke — 描画色をグレーに設定する [古い関数]
- PDF_setgray — 色をグレーに設定する [古い関数]
- PDF_setlinecap — linecap パラメータを設定する
- PDF_setlinejoin — linejoin パラメータを設定する
- PDF_setlinewidth — 線幅を設定する
- PDF_setmatrix — 現在の変換行列を設定する
- PDF_setmiterlimit — miter limit を設定する
- PDF_setpolydash — 複雑な破線パターンを設定する [古い関数]
- PDF_setrgbcolor_fill — 塗りつぶし RGB 色の値を設定する
- PDF_setrgbcolor_stroke — 描画 RGB 色を設定する [古い関数]
- PDF_setrgbcolor — 描画および塗りつぶし RGB 色を設定する [古い関数]
- PDF_shading_pattern — シェーディングパターンを定義する
- PDF_shading — 混色を定義する
- PDF_shfill — シェーディングで領域を塗りつぶす
- PDF_show_boxed — ボックスにテキストを出力する [古い関数]
- PDF_show_xy — 指定した位置にテキストを出力する
- PDF_show — 現在の位置にテキストを出力する
- PDF_skew — 座標系を歪ませる
- PDF_stringwidth — テキストの幅を返す
- PDF_stroke — パスを描く
- PDF_suspend_page — ページを停止する
- PDF_translate — 座標系の原点を設定する
- PDF_utf16_to_utf8 — 文字列を UTF-16 から UTF-8 に変換する
- PDF_utf32_to_utf16 — 文字列を UTF-32 から UTF-16 に変換する
- PDF_utf8_to_utf16 — 文字列を UTF-8 から UTF-16 に変換する

PDO 関数

導入

PHP Data Objects (PDO) 拡張モジュールは、 PHP の中からデータベースにアクセスするための軽量で高性能な インターフェイスを定義します。 PDO インターフェイスを実装する各データベースドライバは、 正規表現関数のようなデータベース固有の機能を提供することができます。 PDO 拡張モジュールによりそのデータベースの全てのデータベース関数を 実行できるわけではないことに注意してください。 データベースサーバにアクセスするには、 データベース固有の PDO ドライバ を使用する必要があります。

PDO は、データアクセスの抽象化レイヤを提供します。 つまり、使用しているデータベースが何であるかにかかわらず、同じ 関数を使用してクエリの発行やデータの取得が行えるということです。 PDO は、データベースの抽象化を行うのでは ありません。つまり、SQL を書き直したり 存在しない機能をエミュレートしたりはしないということです。 もしそのような機能が必要なら、全体を網羅する (full-blown) 別の抽象化レイヤを使用すべきです。

PDO は PHP 5.1 以降にバンドルされており、PHP 5.0 では PECL 拡張モジュールとして使用可能です。PDO は PHP 5 の新機能である オブジェクト指向機能を使用しており、それより前のバージョンの PHP では動作しません。

インストール手順

Unix システム上での PHP 5.1 以降

1. PHP 5.1 を動作させているのならば、PDO および PDO_SQLITE はその配布物の中に含まれています。configure を実行した際に PDO は自動的に有効になります。 PDO は、共有モジュールとしてビルドすることを推奨します。 なぜなら、PECL からアップデート版を導入する際にそのほうが有利だからです。 PHP を PDO サポート込みでビルドする際の推奨設定は、 zlib サポートを有効にする (pecl インストーラを使用するため) ことです。 また、あなたが選択したデータベースについての PDO ドライバを有効にする必要もあります。 詳細については データベース固有の PDO ドライバ を参照ください。PDO を共有モジュールとしてビルドした場合は、 PDO ドライバも共有モジュールとしてビルドする必要があることに注意しましょう。 SQLite 拡張モジュールは PDO に依存しています。そのため、PDO を共有モジュールとしてビルドした場合には SQLite も同じようにビルドしなければなりません。

   ```
   ./configure  --with-zlib  --enable-pdo=shared  --with-pdo-sqlite=shared  --with-sqlite=shared
   ```

2. PDO を共有モジュールとしてインストールした後は、PHP の実行時に PDO 拡張モジュールが自動的に読み込まれるよう、php.ini ファイルを編集する必要があります。 また同様に、データベース固有のドライバについてもここで有効にする必要があります。 この設定は pdo.so の行より後で記述するようにしましょう。なぜなら、 データベース固有のドライバが読み込まれるためには、それ以前に PDO が初期化されていなければならないからです。 PDO およびデータベース固有のドライバを静的にビルドしたのならば、 この設定は不要です。

   ```
   extension=pdo.so
   ```

3. PDO を共有モジュールとしてビルドすると、新しいバージョンの PDO が公開された際に **pecl upgrade pdo** コマンドで更新することができるようになります。 その際に PHP 自体を再ビルドする必要はありません。注意すべき点として、 PDO をこの方法で更新した際はデータベース固有のドライバも同時に更新するようにしてください。

Unix システム上での PHP 5.0.0 以降

1. PDO は、 » http://pecl.php.net/package/pdo から PECL 拡張モジュールとして使用可能です。 **pecl** ツールを使用してインストールを行います。 このツールは、PHP の configure 時に自動的に使用可能になります。 **pecl** が圧縮されたパッケージを取り扱えるよう、 PHP が --with-zlib を含めて confiure されていることを 確認しておきましょう。

2. 以下のコマンドにより、最新の安定版の PDO をダウンロードしてビルドを行い、そしてインストールします。

```
pecl install pdo
```

3. **pecl** コマンドは、PDO モジュールを自動的に PHP のエクステンションディレクトリにインストールします。 Linux や Unix 上で PDO を有効にするには、以下の行を php.ini に追加する必要があります。

```
extension=pdo.so
```

PECL パッケージのビルド方法についてのより詳細な情報は、マニュアルの PECL 拡張モジュールのインストール を参照ください。

PHP 5.1.0 以降を使用している Windows ユーザ

1. PDO および主要データベースのドライバは、共有モジュールとして PHP に同梱されています。これを使用するには、単に php.ini ファイルを編集するだけです。

```
extension=php_pdo.dll
```

2. 次に、その他のデータベース固有の DLL ファイルを選択します。 実行時に dl() によりロードするか、または、 php.ini の php_pdo.dll の下で有効にしてください。 例えば、以下のようになります。

```
extension=php_pdo.dll
extension=php_pdo_firebird.dll
extension=php_pdo_informix.dll
extension=php_pdo_mssql.dll
extension=php_pdo_mysql.dll
extension=php_pdo_oci.dll
extension=php_pdo_oci8.dll
extension=php_pdo_odbc.dll
extension=php_pdo_pgsql.dll
extension=php_pdo_sqlite.dll
```

これらの DLL は、システムの extension_dir で指定した場所に存在する必要があります。 PDO_INFORMIX は PECL 拡張モジュールにのみ存在することに注意しましょう。

実行時設定

php.ini の設定により動作が変化します。

PDO 設定オプション

名前	デフォルト	変更の可否	変更履歴
pdo.dsn.*		*php.ini* only	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

*pdo.dsn.** string

DSN の別名を定義します。完全な説明については PDO->__construct() を参照ください。

PDO ドライバ

以下のドライバが現在 PDO インターフェイスを実装しています。

ドライバ名	サポートされるデータベース
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird/Interbase 6
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2、unixODBC そして win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 と SQLite 2

接続、および接続の管理

PDO 基底クラスのインスタンスを作成することにより、接続が確立されます。 どのドライバを使用するのかにかかわらず、常に PDO クラスを指定します。 コンストラクタに渡す引数により、データソース（いわゆる DSN）の指定や（もしあれば、オプションで）ユーザ名およびパスワードの指定を行います。

Example#1 MySQL への接続

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
?>
```

接続時になんらかのエラーが発生した場合、PDOException オブジェクトがスローされます。エラー処理を行いたい場合はこの例外を キャッチします。あるいはこれを無視して、 set_exception_handler() で設定した グローバル例外ハンドラに処理を任せることもできます。

Example#2 接続エラーの処理

```php
<?php
try {
    $dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
    foreach ($dbh->query('SELECT * from FOO') as $row) {
       print_r($row);
    }
    $dbh = null;
} catch (PDOException $e) {
    print "エラー!: " . $e->getMessage() . "<br/>";
    die();
}
?>
```

警告

PDO コンストラクタからの例外をアプリケーション内でキャッチしない場合、 zend エンジンはスクリプトの実行を終了し、バックトレースを表示します。 このバックトレースを見れば、データベースへの接続の詳細がわかってしまいます。 その中にはユーザ名やパスワードも含まれます。 (catch 文を使用して) 明示的に例外をキャッチするか、 あるいは set_exception_handler() を使用して 暗黙的に例外をキャッチするようにしましょう。

データベースへの接続に成功すると、PDO クラスのインスタンスが スクリプトに返されます。この PDO オブジェクトが存在する間、 接続がアクティブであり続けます。接続を閉じるには、他から 参照されていないことを保障することでオブジェクトを破棄する 必要があります。それには、オブジェクトを保持している変数に対して NULL を代入します。 明示的にこれを行わなかった場合は、スクリプトの終了時に自動的に 接続が閉じられます。

Example#3 接続を閉じる

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
// ここで接続を使用します


// 使用を終了したので、閉じます
$dbh = null;
?>
```

データベースサーバへの持続的な接続による恩恵をこうむる web アプリケーションは多いでしょう。持続的な接続は、スクリプトが 終了しても閉じられずにキャッシュされ、他のスクリプトが同じ内容の 接続を要求してきた際にそれが再利用されます。持続的接続の キャッシュにより、スクリプトがデータベースを使用するたびに 新しい接続を確立するオーバーヘッドを避けることができます。 それにより、結果として web アプリケーションを高速化できるように なります。

Example#4 持続的な接続

```php
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass, array(
  PDO::ATTR_PERSISTENT => true
));
?>
```

> **注意**: 持続的な接続を使用したい場合は、ドライバのオプションを表す配列に **PDO::ATTR_PERSISTENT** を設定して PDO のコンストラクタに渡す必要があります。この属性を PDO->setAttribute() を用いてインスタンス作成後に設定した場合は、 そのドライバは持続的な接続を使用しません。
>
> **注意**: PDO ODBC ドライバを使用しており、ODBC ライブラリが ODBC 接続プーリングをサポートしている場合 (unixODBC および Windows はこれをサポートしています。他にもあるかもしれません) は、 PDO の持続的接続を使用せずに ODBC の接続プーリングに接続キャッシュ処理を任せることを推奨します。 ODBC の接続プールは、プロセス内で他のモジュールと共有されています。 PDO が接続をキャッシュしてしまうと、その接続は ODBC の 接続プールに返されなくなり、他のモジュールによって新たな接続が 作成されてしまうようになります。

トランザクションおよび自動コミット

さあ、PDO を使用してデータベースに接続することができました。 クエリを発行する前に、PDO がトランザクションをどのように扱うのかを 理解しなければなりません。トランザクションについてよくわからない方の ために説明すると、これは以下の 4 つの機能、つまり 原子性 - Atomicity、一貫性 - Consistency、独立性 - Isolation および 永続性 - Durability (ACID) を提供するものです。 一般的な言葉で言うと、トランザクション内で実行された作業は たとえ段階的に行われたものであってもデータベースに安全に反映される ことが保証されています。トランザクションのコミット時に他の接続の 干渉を受けることはありません。また、トランザクション内での作業は (まだコミットされていなければ) いつでも自動的に取り消すことが できます。これにより、スクリプト内でのエラー処理がより楽になります。

トランザクションの一般的な実装は、変更内容を一時的に「蓄えて」おき、 それを一気に適用するようになっています。この実装には、更新処理の 性能を劇的に向上させるという効果もあります。つまり、 トランザクションによってあなたの書くスクリプトはより高速になり、 またより堅牢になるということです (これらの恩恵をうけるためには、トランザクションを正しく使用する 必要があります)。

残念ながら、すべてのデータベースがトランザクションをサポートしていると いうわけではありません。そのため、PDO で最初に接続をオープンした際には、 いわゆる「自動コミット」モードで動作します。自動コミットモードとは、 もしデータベースがトランザクションをサポートしていたら個々のクエリが 暗黙的なトランザクションのもとで実行され、サポートしていなかったら トランザクションを使用せずに実行されることを意味します。トランザクションを使用する場合は、 PDO->beginTransaction() メソッドを使用して トランザクションを初期化する必要があります。使用しているドライバが トランザクションをサポートしていない場合は PDOException が スローされます (これは深刻な状態であるため、エラー処理の設定にかかわらず常にスローされます)。初期化した後でトランザクションを 終了させるには、トランザクション内でのコードが成功したか否かに応じて PDO->commit() あるいは PDO->rollBack() を使用します。

スクリプトが終了したり接続が閉じられようとした際に、もし処理が 完了していないトランザクションがあれば PDO が自動的に ロールバックします。これは、スクリプトが予期せぬ状態で終了した場合に データの不整合が発生するのを避けるための安全装置です。もし 明示的にコミットしていなければ、おそらく何かおかしなことが 起こったのだろうと推測されます。そのため、データを守るために ロールバックが行われるのです。

警告

自動的にロールバックが行われるのは、トランザクションを PDO->beginTransaction() で開始した場合のみです。 トランザクションを開始するクエリを手動で発行した場合、 PDO はそれを知ることができません。そのため、何か問題が発生しても ロールバックすることはできないのです。

Example#5 トランザクション内で一括処理を行う

以下の例では、新しい従業員のデータを作成しているものとします。 この従業員には ID 番号 23 を割り当てます。この人物についての 基礎データを入力するだけでなく、その給与についても登録する 必要があります。以下では単純に 2 つの別々の更新を行っていますが、 それらは PDO->beginTransaction() および PDO->commit() のコールで囲まれています。 これにより、変更が完了するまでは他からは一切変更内容が 見えないことが保証されます。もし何か問題が発生すれば、 catch ブロック内でトランザクション開始以降のすべての変更が ロールバックされます。そしてエラーメッセージを表示します。

```
<?php
try {
  $dbh = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2',
       array(PDO::ATTR_PERSISTENT => true));
  echo "接続しました\n";
  $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $dbh->beginTransaction();
  $dbh->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
  $dbh->exec("insert into salarychange (id, amount, changedate)
       values (23, 50000, NOW())");
  $dbh->commit();

} catch (Exception $e) {
  $dbh->rollBack();
  echo "失敗しました。" . $e->getMessage();
}
?>
```

トランザクション内で行える処理は、更新には限りません。たとえば 何か複雑なクエリを発行してデータを抽出し、その情報をもとに データの更新をしたり別のクエリを実行したりすることも可能です。 トランザクションがアクティブな間は、作業中のデータについては 他から一切変更が加えられないことが保証されます。 実際のところこの説明は 100% 正確というわけではありませんが、 もしあなたがいままでトランザクションのことを知らなかったのであれば このように理解しておけば十分でしょう。

プリペアドステートメントおよびストアドプロシージャ

より成熟したデータベースの多くは、プリペアドステートメントという 概念をサポートしています。プリペアドステートメントとはいったい何の ことでしょう? これは、実行したい SQL をコンパイルした 一種のテンプレートのようなものです。パラメータ変数を使用することで SQL をカスタマイズすることが可能です。プリペアドステートメントには 2 つの大きな利点があります。

- クエリのパース（あるいは準備）が必要なのは最初の一回だけで、 同じパラメータ（あるいは別のパラメータ）を指定して何度でも クエリを実行することができます。クエリを実行するには、準備として クエリの解析やコンパイル、そして実行プランの最適化が行われます。 クエリが複雑になると、この処理には時間がかかるようになります。 同じクエリを異なったパラメータで何度も実行すると、アプリケーションの 動作は目に見えて遅くなるでしょう。 プリペアドステートメントを使用すると、この 解析/コンパイル/最適化 の繰り返しを避けることができます。端的に言うと、プリペアドステートメントは使用するリソースが少なくいため 高速に動作するということです。
- プリペアドステートメントに渡すパラメータは、引用符で括る必要は ありません。それはドライバが自動的に行います。 アプリケーションで明示的にプリペアドステートメントを使用するように すれば、SQL インジェクションは決して発生しません（しかし、もし信頼できない入力をもとにクエリの他の部分を構築している のならば、その部分に対するリスクを負うことになります）。

プリペアドステートメントは非常に有用な機能なので、もしドライバが サポートしていなくても、例外的に PDO がこの機能をエミュレートします。これにより、データベースの機能にかかわらず同じ仕組みで データベースへのアクセスができることが保証されます。

Example#6 プリペアドステートメントを使用して、繰り返し挿入処理を行う

この例は、name および value を名前つきプレースホルダで置き換えて INSERT クエリを実行します。

```
<?php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);

// 行を挿入します
$name = 'one';
$value = 1;
$stmt->execute();

// パラメータを変更し、別の行を挿入します
$name = 'two';
$value = 2;
$stmt->execute();
?>
```

Example#7 プリペアドステートメントを使用して、繰り返し挿入処理を行う

この例は、name および value をプレースホルダ ? で置き換えて INSERT クエリを実行します。

```
<?php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (?, ?)");
$stmt->bindParam(1, $name);
$stmt->bindParam(2, $value);

// 行を挿入します
$name = 'one';
$value = 1;
$stmt->execute();

// パラメータを変更し、別の行を挿入します
$name = 'two';
$value = 2;
$stmt->execute();
?>
```

Example#8 プリペアドステートメントを使用してデータを取得する

この例では、フォームで入力したキーの値に応じたデータを取得します。 ユーザの入力内容は自動的に引用符で括られるので、SQL インジェクション攻撃の 恐れはありません。

```
<?php
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name = ?");
if ($stmt->execute(array($_GET['name']))) {
  while ($row = $stmt->fetch()) {
```

```
    print_r($row);
  }
}
?>
```

データベースドライバがサポートしていれば、入力パラメータだけでなく 出力パラメータもバインドすることが可能です。出力パラメータは、 一般にストアドプロシージャから値を受け取るために使用します。この場合、 返される値の大きさがどの程度になるのかをバインド時に知っておく必要が あります。指定した大きさよりも大きな値が返されると、エラーが発生します。

Example#9 出力パラメータを指定してストアドプロシージャをコールする

```
<?php
$stmt = $dbh->prepare("CALL sp_returns_string(?)");
$stmt->bindParam(1, $return_value, PDO::PARAM_STR, 4000);

// ストアドプロシージャをコールします
$stmt->execute();

print "プロシージャが返した値は $return_value です\n";
?>
```

入出力の両方に使用するパラメータを指定することもできます。 このパラメータの書式は、出力パラメータと同じです。 次の例では、ストアドプロシージャに文字列 'hello' を渡しています。 プロシージャの結果が返ってくると、 この文字列はプロシージャの返す値に置き換えられます。

Example#10 入出力パラメータを指定してストアドプロシージャをコールする

```
<?php
$stmt = $dbh->prepare("CALL sp_takes_string_returns_string(?)");
$value = 'hello';
$stmt->bindParam(1, $value, PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT, 4000);

// ストアドプロシージャをコールします
$stmt->execute();

print "プロシージャが返した値は $value です\n";
?>
```

Example#11 プレースホルダの間違った使用法

```
<?php
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name LIKE '%?%'");
$stmt->execute(array($_GET['name']));

// プレースホルダは、値全体に対して使用しなければなりません
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name LIKE ?");
$stmt->execute(array("%$_GET[name]%"));
?>
```

エラーおよびエラー処理

PDO が提供するエラー処理方法は 3 通り存在し、 アプリケーションの開発形態によって使い分けることができます。

- **PDO::ERRMODE_SILENT**

  デフォルトのモードです。ステートメントおよびデータベースオブジェクトの エラーについて、PDO は単にそのエラーコードのみを設定します。 これを取得するには PDO->errorCode() および PDO->errorInfo() メソッドを使用します。 ステートメントオブジェクトへのコールによってエラーが発生した場合は、 そのオブジェクトの PDOStatement->errorCode() あるいは PDO->errorInfo() メソッドを呼び出します。 データベースオブジェクトへのコールによってエラーが発生した場合は、 その代わりにデータベースオブジェクト上の同じメソッドを呼び出します。

- **PDO::ERRMODE_WARNING**

  エラーコードを設定することに加え、PDO は 伝統的な E_WARNING メッセージも出力します。この設定はデバッグ/テストの際に有用で、 アプリケーションの動作を妨げることなしに問題点を確認できるように なります。

- **PDO::ERRMODE_EXCEPTION**

  エラーコードを設定することに加え、PDO は PDOException をスローします。エラーコードや 関連情報が、クラスのプロパティとして設定されます。 この設定もまたデバッグ時に有用で、エラーが発生した時点で スクリプトの実行を停止させることによりコード内の問題点を 見つけやすくなります (例外によりスクリプトが終了した際には、トランザクションは自動的に ロールバックされることを覚えておきましょう)。

  このモードが有用である理由のひとつとして、伝統的な PHP 形式の警告よりも より明確にエラー処理コードが書けることがあります。例外を発生させず、 データベースへのコールのたびに毎回明示的に返り値をチェックすることに 比べると、コードの量やネストを減らすことができます。

  PHP の例外についての詳細な情報は、 例外 を参照ください。

PDO のエラーコードは、SQL-92 の SQLSTATE エラーコード文字列に 標準化されています。 ネイティブのコードを適切な SQLSTATE コードに変換するのは、個々の PDO ドライバの仕事となります。 PDO->errorCode() メソッドは SQLSTATE コードを返します。 エラーについての詳細な銃尾法が知りたい場合、PDO では PDO->errorInfo() メソッドも提供しており、 これは SQLSTATE コード、ドライバ固有のエラーコードおよびドライバ固有の エラーメッセージを含む配列を返します。

ラージオブジェクト (LOB)

アプリケーション内で、データベースに「大きな」データを格納する 必要を感じることがあるかもしれません。「大きな」とは、一般的には 「4kb 程度以上」を指しますが、データベースによっては 32kb くらいまでは 「大きい」と判断されずにすむこともあります。ラージオブジェクトは テキストあるいはバイナリの両方の形式をとり得ます。 PDO でこのラージデータ型を扱うには、 PDOStatement->bindParam() や PDOStatement->bindColumn() のコール時に 型コードとして **PDO::PARAM_LOB** を使用します。 **PDO::PARAM_LOB** を指定すると、PDO は データをストリームにマップします。これにより、 PHP ストリーム API を使用してデータを扱えるようになります。

Example#12 データベース内の画像を表示する

この例では $lob という名前の変数に LOB をバインドし、 fpassthru() を使用してそれをブラウザに送信します LOB はストリームで表されるので、 fgets()、fread() および stream_get_contents() といった関数を 使用することができます。

```
<?php
$db = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2');
$stmt = $db->prepare("select contenttype, imagedata from images where id=?");
```

```
$stmt->execute(array($_GET['id']));
$stmt->bindColumn(1, $type, PDO::PARAM_STR, 256);
$stmt->bindColumn(2, $lob, PDO::PARAM_LOB);
$stmt->fetch(PDO::FETCH_BOUND);

header("Content-Type:  $type");
fpassthru($lob);
?>
```

Example#13 画像をデータベースに挿入する

この例では、ファイルをオープンしてそのハンドルを PDO に渡し、 LOB としてデータベースに挿入します。PDO は、データベースに応じた もっとも適切な方法でデータを取得します。

```
<?php
$db = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2');
$stmt = $db->prepare("insert into images (id, contenttype, imagedata) values (?, ?, ?)");
$id = get_new_id(); // 新しい ID を割り当てるための何らかの関数

// フォームからファイルをアップロードしていると仮定します。
// 詳細な情報は PHP のドキュメントを参照ください。

$fp = fopen($_FILES['file']['tmp_name'], 'rb');

$stmt->bindParam(1, $id);
$stmt->bindParam(2, $_FILES['file']['type']);
$stmt->bindParam(3, $fp, PDO::PARAM_LOB);

$db->beginTransaction();
$stmt->execute();
$db->commit();
?>
```

Example#14 画像をデータベースに挿入する: Oracle

Oracle は、ファイルから LOB を挿入する方法が他とは少し違います。 また、必ずトランザクション内で挿入しなければなりません。 それ以外の場合、新しく挿入された LOB は長さゼロとなり、クエリの 実行時に暗黙的にコミットされます。

```
<?php
$db = new PDO('oci:', 'scott', 'tiger');
$stmt = $db->prepare("insert into images (id, contenttype, imagedata) " .
  "VALUES (?, ?, EMPTY_BLOB()) RETURNING imagedata INTO ?");
$id = get_new_id(); // 新しい ID を割り当てるための何らかの関数

// フォームからファイルをアップロードしていると仮定します。
// 詳細な情報は PHP のドキュメントを参照ください。

$fp = fopen($_FILES['file']['tmp_name'], 'rb');

$stmt->bindParam(1, $id);
$stmt->bindParam(2, $_FILES['file']['type']);
$stmt->bindParam(3, $fp, PDO::PARAM_LOB);

$stmt->beginTransaction();
$stmt->execute();
$stmt->commit();
?>
```

定義済みクラス

PDO

PHP とデータベースサーバの間の接続を表します。

コンストラクタ

- PDO - PDO オブジェクトのコンストラクタ

メソッド

- beginTransaction - トランザクションを開始する
- commit - トランザクションを コミットする
- errorCode - エラーが発生した場合に、データベースからエラーコードを取得する
- errorInfo - エラーが発生した場合に、データベースからエラー情報の配列を取得する
- exec - SQL ステートメントを 発行し、作用された行の数を返す
- getAttribute - データベース接続属性を取得する
- lastInsertId - テーブルに挿入された直近の行の値を取得する
- prepare - SQL ステートメントを実行するために準備する
- rollBack - トランザクションをロールバックする
- setAttribute - データベース接続属性を設定する

PDOStatement

プリペアドステートメントを表します。ステートメント実行後は関連する結果セットを 表します。

メソッド

- bindColumn - PHP 変数を結果セットの出力カラムにバインドする
- bindParam - プリペアドステートメントのパラメータに PHP 変数をバインドする
- bindValue - プリペアドステートメントのパラメータに値をバインドする
- closeCursor - カーソルを閉じてステートメントを再実行できるようにする
- columnCount - 結果セットのカラム数を返す
- errorCode - エラーが発生した場合、ステートメントからエラーコードを取得する
- errorInfo - エラーが発生した場合、ステートメントからエラー情報の配列を取得する
- execute - プリペアドステートメントを実行する
- fetch - 結果セットから行を取得する
- fetchAll - 結果セットからすべての行を含む配列を返す
- fetchColumn 結果セットの単一カラムからデータを返す
- getAttribute - PDOStatement 属性を取得する
- getColumnMeta - 結果セットのカラムのメタデータを取得する
- nextRowset - 次の行セット（結果セット）を取得する
- rowCount - SQL ステートメントの実行により作用された行の数を返す
- setAttribute - PDOStatement 属性を設定する
- setFetchMode - PDOStatement の取得モードを設定する

PDOException

PDO が発するエラーを表します。あなた自身が書いたコードから PDOException をスローしてはいけません。 PHP の例外についての詳細な情報は、例外 を参照ください。

Example#15 PDOException クラス

```
<?php
class PDOException extends Exception
{
    public $errorInfo = null;    //  PDO::errorInfo()
                                    // あるいは PDOStatement::errorInfo() に対応します。
    protected $message;          //  テキストのエラーメッセージ。
                                    // Exception::getMessage() を使用してアクセスします。
    protected $code;             //  SQLSTATE エラーコード。
                                    // Exception::getCode() を使用してアクセスします。
}
?>
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

警告

PHP 5.1 以降、PDO はクラス定数を使用します。 それ以前のリリースでは、**PDO_PARAM_BOOL** の形式のグローバル変数を使用します。

PDO::PARAM_BOOL (integer)
ブールデータ型を表します。
PDO::PARAM_NULL (integer)
SQL NULL データ型を表します。
PDO::PARAM_INT (integer)
SQL INTEGER データ型を表します。
PDO::PARAM_STR (integer)
SQL CHAR, VARCHAR, または他の文字列データ型を表します。
PDO::PARAM_LOB (integer)
SQL ラージオブジェクト型を表します。
PDO::PARAM_STMT (integer)
PDO::PARAM_INPUT_OUTPUT (integer)
パラメータがストアドプロシージャ用の入力パラメータであることを指定します。 この値は、PDO::PARAM_* データ型とのビットORとして指定する必要があります。
PDO::FETCH_LAZY (integer)
取得する方法として、 結果セットが返すカラム名と同じ名前の変数を有するオブジェクトとして各行を返す方法を 指定します。 PDO::FETCH_LAZY は、アクセスされたものと同じ名前のオブジェクト変数を作成します。
PDO::FETCH_ASSOC (integer)
結果セットの対応するカラム名にふられているものと同じキーを付けた 連想配列として各行を返す取得方法を指定します。 もし結果セットが複数のカラムを同名で含む場合、 PDO::FETCH_ASSOC はカラム名毎に 1 つの値のみ返します。
PDO::FETCH_NAMED (integer)
結果セットの対応するカラム名にふられているものと同じキーを付けた 連想配列として各行を返す取得方法を指定します。 もし結果セットが複数のカラムを同名で含む場合、 PDO::FETCH_NAMED はカラム名毎に値の配列を返します。
PDO::FETCH_NUM (integer)
結果セットの対応するカラム番号にふられているものと同じ添字を付けた 配列として各行を返す取得方法を指定します。番号は0から始まります。
PDO::FETCH_BOTH (integer)
結果セットと同じカラム名と0から始まるカラム番号を付けた配列として各行を返す 方法を指定します。
PDO::FETCH_OBJ (integer)
結果セットが返すカラム名と同じ名前のプロパティを有する オブジェクトとして各行を返す方法を指定します。
PDO::FETCH_BOUND (integer)
結果セットのカラムの値を **PDOStatement::bindParam()** または **PDOStatement::bindColumn()** メソッドでバインドされた PHP変数に代入し、TRUEを返すという取得方法を指定します。
PDO::FETCH_COLUMN (integer)

結果セットの次の行から指定された一つのカラムのみを返す取得方法を指定します。
PDO::FETCH_CLASS (integer)
カラムをクラスのプロパティにマップしつつ、 指定されたクラスの新規インスタンスを返す取得方法を指定します。
PDO::FETCH_INTO (integer)
カラムをクラスのプロパティにマップしつつ、 指定されたクラスの既存のインスタンスを更新する取得方法を指定します。
PDO::FETCH_FUNC (integer)
PDO::FETCH_GROUP (integer)
PDO::FETCH_UNIQUE (integer)
PDO::FETCH_KEY_PAIR (integer)
最初のカラムの値をキー、それ以降のカラムの内容を値として持つ連想配列形式でデータを取得します。
PDO::FETCH_CLASSTYPE (integer)
PDO::FETCH_SERIALIZE (integer)
PDO::FETCH_INTO と同様ですが、 シリアライズした文字列としてオブジェクトを提供します。 PHP 5.1.0 以降で使用可能です。
PDO::FETCH_PROPS_LATE (integer)
PHP 5.2.0 以降で使用可能です。
PDO::ATTR_AUTOCOMMIT (integer)
この値が **FALSE** の場合、PDO は接続がトランザクションを開始できるように オートコミットを無効にしようとします。
PDO::ATTR_PREFETCH (integer)
独自アプリケーションにおけるメモリ使用量に対する速度のバランスを 調整するためのプリフェッチサイズを設定します。 全てのデータベースとドライバの組み合わせでプリフェッチサイズの設定を サポートしているわけではありません。
PDO::ATTR_TIMEOUT (integer)
データベースとの通信に対するタイムアウト値を秒で設定します。
PDO::ATTR_ERRMODE (integer)
PDO::ATTR_SERVER_VERSION (integer)
PDO::ATTR_CLIENT_VERSION (integer)
PDO::ATTR_SERVER_INFO (integer)
PDO::ATTR_CONNECTION_STATUS (integer)
PDO::ATTR_CASE (integer)
PDO::CASE_* 定数で指定されたケースにカラム名を変更します。
PDO::ATTR_CURSOR_NAME (integer)
PDO::ATTR_CURSOR (integer)
PDO::ATTR_DRIVER_NAME (string)
ドライバ名を返します。
PDO::ATTR_ORACLE_NULLS (integer)
空文字を SQL の NULL 値に変換します。
PDO::ATTR_PERSISTENT (integer)
新規接続を生成するよりもむしろ持続的接続を要求します。
PDO::ATTR_STATEMENT_CLASS (integer)
PDO::ATTR_FETCH_CATALOG_NAMES (integer)
結果セット中の各カラム名にカタログ名を追加します。 カタログ名とカラム名は、小数点 (.) で区切られます。
PDO::ATTR_FETCH_TABLE_NAMES (integer)
結果セット中の各カラム名にテーブル名を追加します。 テーブル名とカラム名は、小数点 (.) で区切られます。
PDO::ATTR_STRINGIFY_FETCHES (integer)
PDO::ATTR_MAX_COLUMN_LEN (integer)
PDO::ATTR_DEFAULT_FETCH_MODE (integer)
PHP 5.2.0 以降で使用可能です。
PDO::ATTR_EMULATE_PREPARES (integer)
PHP 5.1.3 以降で使用可能です。
PDO::ERRMODE_SILENT (integer)
エラー時にエラーもしくは例外を発生しません。 開発者の方は明示的にエラーをチェックするようにしてください。 これはデフォルトのモードです。
PDO::ERRMODE_WARNING (integer)
エラーが発生した場合、PHP の E_WARNING メッセージを発行します。
PDO::ERRMODE_EXCEPTION (integer)
エラーが発生した場合、PDOException を投げます。
PDO::CASE_NATURAL (integer)
カラム名をデータベースドライバにより返されたままにします。
PDO::CASE_LOWER (integer)
カラム名を小文字にします。
PDO::CASE_UPPER (integer)
カラム名を大文字にします。
PDO::NULL_NATURAL (integer)
PDO::NULL_EMPTY_STRING (integer)
PDO::NULL_TO_STRING (integer)
PDO::FETCH_ORI_NEXT (integer)
結果セットの次の行を取得します。スクローラブルなカーソルでのみ有効です。
PDO::FETCH_ORI_PRIOR (integer)
結果セットの前の行を取得します。スクローラブルなカーソルでのみ有効です。
PDO::FETCH_ORI_FIRST (integer)
結果セットの先頭の行を取得します。スクローラブルなカーソルでのみ有効です。
PDO::FETCH_ORI_LAST (integer)
結果セットの最後の行を取得します。スクローラブルなカーソルでのみ有効です。
PDO::FETCH_ORI_ABS (integer)
結果セットから行番号で指定した行を取得します。スクローラブルなカーソルでのみ有効です。
PDO::FETCH_ORI_REL (integer)
結果セットのカーソルの現在の位置を基準とする相対位置により指定された行を 取得します。スクローラブルなカーソルでのみ有効です。
PDO::CURSOR_FWDONLY (integer)
前進のみ可能なカーソルを有するPDOStatementオブジェクトを生成します。 これにより、アプリケーションの性能は改善しますが、 PDOStatement オブジェクトは前方にある結果セットから一度に一行を 取得するという制約を受けます。
PDO::CURSOR_SCROLL (integer)
スクローラブルカーソルを有するPDOStatementオブジェクトを作成しあす。 結果セットから取得した行を制御するためのPDO::FETCH_ORI_*定数を指定して ください。
PDO::ERR_CANT_MAP (integer)
PDO::ERR_SYNTAX (integer)
PDO::ERR_CONSTRAINT (integer)
PDO::ERR_NOT_FOUND (integer)
PDO::ERR_ALREADY_EXISTS (integer)
PDO::ERR_NOT_IMPLEMENTED (integer)
PDO::ERR_MISMATCH (integer)
PDO::ERR_TRUNCATED (integer)
PDO::ERR_DISCONNECTED (integer)
PDO::ERR_NO_PERM (integer)
PDO::ERR_NONE (string)
SQLSTATE '00000' は SQL ステートメントがエラーや警告がなく発行に成功したことを意味します。 この定数はエラーが発生したかどうかを判別するために **PDO::errorCode()** もしくは **PDOStatement::errorCode()** をチェックする際に 便利です。この場合、通常はエラー状態を発生したメソッドからの戻りコードを 検査することによって検知します。
PDO::PARAM_EVT_ALLOC (integer)
割り当てられたときに発生するイベント。
PDO::PARAM_EVT_FREE (integer)
割り当てが解除されたときに発生するイベント。
PDO::PARAM_EVT_EXEC_PRE (integer)
プリペアドステートメントの実行前に発生するイベント。

PDO::PARAM_EVT_EXEC_POST (integer)
プリペアドステートメントの実行後に発生するイベント。
PDO::PARAM_EVT_FETCH_PRE (integer)
結果セットから結果を取得する前に発生するイベント。
PDO::PARAM_EVT_FETCH_POST (integer)
結果セットから結果を取得した後に発生するイベント。
PDO::PARAM_EVT_NORMALIZE (integer)
バインドパラメータの登録時に発生するイベント。 これにより、ドライバがパラメータ名を正規化できるようになります。

PDO->beginTransaction()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->beginTransaction() — トランザクションを開始する

説明

PDO
bool **beginTransaction** (void)

オートコミットモードをオフにします。オートコミットモードがオフの間、 PDO オブジェクトを通じてデータベースに加えた変更は **PDO::commit()** をコールするまでコミットされません。 **PDO::rollback()** をコールすると、 データベースへの全ての変更をロールバックし、 オートコミットモードに設定された接続を返します。

MySQL を含むいくつかのデータベースでは、DROP TABLE や CREATE TABLE のようなデータベース定義言語 (DDL) ステートメントがトランザクション中に 発行される場合、暗黙的なコミットが自動的に発行されます。 この暗黙的なコミットにより、そのトランザクション境界で 他のあらゆる変更をロールバックすることができなくなるでしょう。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 トランザクションをロールバックする

以下の例は、トランザクションを開始し、 変更をロールバックする前にデータベースを修正する 2 つのステートメントを発行します。 しかしながら MySQL では、DROP TABLE ステートメントは 自動的にトランザクションをコミットするので、 トランザクション中のどの変更もロールバックされません。

```
<?php
/* トランザクションを開始する。オートコミットがオフになる */
$dbh->beginTransaction();

/* データベーススキーマとデータを変更する */
$sth = $dbh->exec("DROP TABLE fruit");
$sth = $dbh->exec("UPDATE dessert
      SET name = 'hamburger'");

/* ミスに気づき、変更をロールバックする */
$dbh->rollBack();

/* データベース接続はオートコミットモードに戻る */
?>
```

参考

- PDO->commit()
- PDO->rollBack()

PDO->commit()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->commit() — トランザクションをコミットする

説明

PDO
bool **commit** (void)

トランザクションをコミットし、 次にPDO->beginTransaction()で新たなトランザクションが開始されるまで、 データベース接続をオートコミットモードに戻します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 トランザクションをコミットする

```
<?php
/* トランザクションを開始する。オートコミットがオフになる */
$dbh->beginTransaction();

/* データベーススキーマを変更する */
```

```
$sth = $dbh->exec("DROP TABLE fruit");

/* 変更をコミットする */
$dbh->commit();

/* データベース接続はオートコミットモードに戻る */
?>
```

参考

- PDO->beginTransaction()
- PDO->rollBack()

PDO->__construct()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->__construct() — データベースへの接続を表す PDO インスタンスを生成する

説明

PDO
PDO **__construct** (string $dsn [, string $username [, string $password [, array $driver_options]]])

指定されたデータベースへの接続を表す PDO インスタンスを生成します。

パラメータ

dsn

データソース名(Data Source Name)またはDSN。データベースに接続するために 必要な情報が含まれます。

一般に、DSNはPDOドライバ名の後にコロンが続き、 各PDOドライバに固有の接続構文が続きます。より多くの情報は PDO driver-specific documentation にあります。

dsn パラメータは、データベースへの接続を生成する ために必要な引数を指定する方法として、3種類の方法をサポートします。

ドライバ呼び出し

dsn に完全な DSN を指定します。

URI 呼び出し

dsn は、**uri:** の後に DSN 文字列を含むファイルの位置を定義する URI が続く形式となります。 この URI には、ローカルファイルまたはリモート URL を指定することができます。

uri:file:///path/to/dsnfile

エイリアス

dsn は、DSN 文字列を定義する php.ini の pdo.dsn.name へマップする名前 name からなります。

注意: エイリアスは、.htaccess や httpd.conf ではなく、 php.ini で定義する必要があります。

username

DSN 文字列のユーザ名。このパラメータは、いくつかの PDO ドライバではオプションです。

password

パスワード。DSN 文字列で必要とされる場合に指定。

driver_options

ドライバ固有の接続オプションを指定するキー=> 値の配列。

返り値

成功時に PDO オブジェクトを返します。

エラー / 例外

PDO->__construct() は、 指定されたデータベースへの接続に失敗した場合、 PDOException を投げます。

例

Example#1 ドライバ呼び出しにより PDO インスタンスを生成する

```
<?php
/* ドライバ呼び出しを使用して ODBC データベースに接続する */
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
$user = 'dbuser';
$password = 'dbpass';

try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}

?>
```

Example#2 URI 呼び出しにより PDO インスタンスを生成する

以下の例では、ファイル /usr/local/dbconnect が存在し、 PHP がこのファイルを読み込む権限を有していることを仮定します。 このファイルには、 PDO_ODBC により DB2 データベースに接続するための PDO DSN が含まれています。

```
odbc:DSN=SAMPLE;UID=john;PWD=mypass
```

これにより、PHP スクリプトでは、単に uri: パラメータを渡し、 ファイルの URI を示すだけでデータベース接続を作成することができるようになります。

```
<?php
/* ドライバ呼び出しを使用して ODBC データベースに接続する */
$dsn = 'uri:file:///usr/local/dbconnect';
$user = '';
$password = '';

try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}

?>
```

Example#3 エイリアスにより PDO インスタンスを生成する

以下の例では、php.ini にエイリアス mydb のみで MySQL データベースに接続できるようにするための以下のエントリが含まれることを 仮定します。

```
[PDO]
pdo.dsn.mydb="mysql:dbname=testdb;host=localhost"
```

```
<?php
/* エイリアスを使用して ODBC データベースに接続する */
$dsn = 'mydb';
$user = '';
$password = '';

try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo '接続に失敗しました: ' . $e->getMessage();
}

?>
```

PDO->errorCode()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->errorCode() — データベースハンドラにおける直近の操作に関連する SQLSTATE を取得する

説明

PDO
string **errorCode** (void)

返り値

SQLSTATE を返します。これは、ANSI SQL-92 標準で定義された英数 5 文字の ID です。簡潔に言えば、SQLSTATE は 2文字のクラス値の後に 3 文字のサブクラス値が続きます。クラス値 01 はワーニングを表し、 戻り値のコード SQL_SUCCESS_WITH_INFO を伴います。 クラス 'IM' を除く '01' 以外のクラス値はエラーを表します。 クラス 'IM' は PDO 自身の実装 (もしくは ODBC ドライバを使用している場合は ODBC かも知れません) に由来するワーニングやエラーに固有の値です。 あらゆるクラスでのサブクラス値 '000' は SQLSTATE に対するサブクラスがない事を示しています。

PDO->errorCode() はデータベースハンドラに 直接行った操作に対するエラーコードのみを取得します。 もし PDO->prepare() や PDO->query() を通して PDOStatement オブジェクトを生成し、 文でエラーが発生した場合、PDOStatement->errorCode() はエラーを反映しません。 特定の文ハンドラに対して実行された操作についてのエラーコードを返すには PDOStatement->errorCode() をコールしなければなりません。

例

Example#1 SQLSTATE コードを取得する

```
<?php
/* エラーを発生させる -- BONES テーブルは存在しない */
$dbh->exec("INSERT INTO bones(skull) VALUES ('lucy')");

echo "¥nPDO::errorCode(): ";
print $dbh->errorCode();
?>
```

上の例の出力は以下となります。

```
PDO::errorCode(): 42S02
```

参考

- PDO->errorInfo()
- PDOStatement->errorCode()
- PDOStatement->errorInfo()

PDO->errorInfo()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->errorInfo() — データベースハンドラにおける直近の操作に関連する拡張エラー情報を取得する

説明

PDO
array **errorInfo** (void)

返り値

PDO->errorInfo() は、 このデータベースハンドラによって実行された直近の操作に関するエラー情報を 配列として返します。この配列は次のフィールドを含みます。

要素	情報
0	SQLSTATE エラーコード (これは、ANSI SQL 標準で定義された英数 5 文字の ID)
1	ドライバ固有のエラーコード
2	ドライバ固有のエラーメッセージ

PDO->errorInfo() はデータベースハンドラに 直接行った操作に対するエラーコードのみを取得します。 もし PDO->prepare() や PDO->query() を通して PDOStatement オブジェクトを生成し、 文でエラーが発生した場合、PDO->errorInfo() はそのエラーを反映しません。 特定の文ハンドラに対して実行された操作についてのエラーコードを返すには PDOStatement->errorInfo() をコールしなければなりません。

例

Example#1 DB2 データベースに対する PDO_ODBC 接続の errorInfo() フィールドを表示する

```
<?php
/* エラーを発生させる -- 無効な SQL シンタックス */
$stmt = $dbh->prepare('bogus sql');
if (!$stmt) {
    echo "¥nPDO::errorInfo():¥n";
    print_r($dbh->errorInfo());
}
?>
```

上の例の出力は以下となります。

```
PDO::errorInfo():
Array
(
    [0] => HY000
    [1] => 1
    [2] => near "bogus": syntax error
)
```

参考

- PDO->errorCode()
- PDOStatement->errorCode()
- PDOStatement->errorInfo()

PDO->exec()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->exec() — SQL ステートメントを実行し、作用した行数を返す

説明

PDO
int **exec** (string $statement)

PDO->exec() は、一度の関数コールで SQL 文を実行し、文によって作用した行数を返します。

PDO->exec() は SELECT 文からは結果を返しません。 プログラム中で一度だけ発行が必要になる SELECT 文に対しては、 PDO->query() の発行を検討してください。 複数回発行が必要な文については、PDO->prepare() による PDOStatement オブジェクトの準備と PDOStatement->execute() による文の発行を行ってください。

パラメータ

statement

準備、実行する SQL ステートメントを指定します。

返り値

PDO->exec() は、発行した SQL ステートメントによって更新もしくは 削除された行数を返します。 1 行も作用しなかった場合、PDO->exec() は 0 を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

以下の例は PDO->exec() の戻り値の使用法を間違っています。結果として一行も更新されなかった場合に die() がコールされてしまうからです。

```
<?php
$db->exec() or die($db->errorInfo());
?>
```

例

Example#1 DELETE 文の発行

WHERE 句を伴う DELETE 文によって削除された行数をカウントします。

```
<?php
$dbh = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');

/* FRUIT テーブルから全ての行を削除する */
$count = $dbh->exec("DELETE FROM fruit WHERE colour = 'red'");

/* 削除された行数を返す */
print("Deleted $count rows.¥n");
?>
```

上の例の出力は以下となります。

```
Deleted 1 rows.
```

参考

- PDO->prepare()
- PDO->query()
- PDOStatement->execute()

PDO->getAttribute()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDO->getAttribute() — データベース接続の属性を取得する

説明

PDO
mixed **getAttribute** (int $attribute)

この関数はデータベース接続の属性値を返します。PDOStatement 属性を取得する場合、PDOStatement->getAttribute() を参照ください。

いくつかのデータベースもしくはドライバは、 データベース接続の属性の全てのをサポートしていないかも知れないことに 注意してください。

パラメータ

attribute

PDO_ATTR_* 定数の 1 つを指定します。 データベース接続に適用される定数は以下の通りです。

- PDO::ATTR_AUTOCOMMIT
- PDO::ATTR_CASE
- PDO::ATTR_CLIENT_VERSION
- PDO::ATTR_CONNECTION_STATUS
- PDO::ATTR_DRIVER_NAME
- PDO::ATTR_ERRMODE
- PDO::ATTR_ORACLE_NULLS
- PDO::ATTR_PERSISTENT
- PDO::ATTR_PREFETCH
- PDO::ATTR_SERVER_INFO
- PDO::ATTR_SERVER_VERSION
- PDO::ATTR_TIMEOUT

返り値

コールに成功した場合は要求された PDO 属性の値を返します。 コールに失敗した場合は null を返します。

例

Example#1 データベース接続の属性を取得する

```
<?php
```

```
$conn = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');
$attributes = array(
    "AUTOCOMMIT", "ERRMODE", "CASE", "CLIENT_VERSION", "CONNECTION_STATUS",
    "ORACLE_NULLS", "PERSISTENT", "PREFETCH", "SERVER_INFO", "SERVER_VERSION",
    "TIMEOUT"
);

foreach ($attributes as $val) {
    echo "PDO::ATTR_$val:  ";
    echo $conn->getAttribute(constant("PDO::ATTR_$val")) . "¥n";
}
?>
```

参考

- PDO->setAttribute()
- PDOStatement->getAttribute()
- PDOStatement->setAttribute()

PDO->getAvailableDrivers()

(No version information available, might be only in CVS)

PDO->getAvailableDrivers() — 利用可能な PDO ドライバの配列を返す

説明

PDO
array **getAvailableDrivers** (void)

この関数は、PDO->__construct() の DSN パラメータで利用可能な全ての有効な PDO ドライバを返します。これはスタティックメソッドです。

返り値

PDO->getAvailableDrivers() は PDO ドライバ名の配列を返します。 もしドライバが何も利用できない場合、空の配列を返します。

例

Example#1 PDO->getAvailableDrivers() の例

```
<?php
print_r(PDO::getAvailableDrivers());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [0] => mysql
    [1] => sqlite
)
```

PDO->lastInsertId()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->lastInsertId() — 最後に挿入された行の ID あるいはシーケンスの値を返す

説明

PDO
string **lastInsertId** ([string $name])

最後に挿入された行の ID、 あるいはシーケンスオブジェクトから次の値をを返します。 これは、構成しているドライバに依存します。例えば PDO_PGSQL() の場合、name パラメータにシーケンスオブジェクト名を指定する必要があります。

> **注意**: このメソッドは、異なる PDO ドライバ間で意味のあるもしくは 一貫性のある結果を返さないかも知れません。 構成しているデータベースが自動インクリメントフィールド、 もしくはシーケンスの概念をサポートしていないかも知れないためです。

パラメータ

name

ID が返されるべきシーケンスオブジェクト名を指定します。

返り値

もし name パラメータにシーケンス名が指定されなかった場合、 PDO->lastInsertId() はデータベースに挿入された最後の行の行IDに相当する文字列を返します。

もし name パラメータにシーケンス名が指定された場合、 PDO->lastInsertId() は指定されたシーケンスオブジェクトから取得した最後の値に相当する 文字列を返します。

もし PDO ドライバがサポートしていない場合、 PDO->lastInsertId() は IM001 SQLSTATE を発生させます。

PDO->prepare()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->prepare() — 文を実行する準備を行い、文オブジェクトを返す

説明

PDO
PDOStatement **prepare** (string $statement [, array $driver_options])

PDOStatement->execute() メソッドによって実行される SQL ステートメントを準備します。 SQL ステートメントは、文が実行されるときに実際の値に置き換えられる 0 個もしくはそれ以上の名前 (:name) もしくは疑問符 (?) パラメータマークを含むことができます。 名前と疑問符パラメータを同一 SQL ステートメント中で使用することはできません。 どちらか一方か、他のパラメータ形式を使用してください。

PDOStatement->execute() をコールする際には、 文に渡すパラメータにはそれぞれ固有のパラメータマークを設定する必要があります。 ひとつのプリペアドステートメントの中で、同じ名前のパラメータマークを 複数使用することはできません。SQL 文の IN() 句などで、 ひとつのパラメータに複数の値を 割り当てることはできません。

異なるパラメータを用いて複数回実行されるような文に対し PDO->prepare() と PDOStatement->execute() をコールすることで、 ドライバがクライアントまたはサーバ側にクエリプランやメタ情報を キャッシュさせるよう調整するため、 アプリケーションのパフォーマンスを最適化します。また、 パラメータに手動でクオートする必要がなくなるので SQL インジェクション攻撃から保護する助けになります。

PDO は元々この機能をサポートしていないドライバに対して プリペアドステートメントとバインドパラメータをエミュレートします。 このため、ある形式をサポートしているがその他の形式をサポートしていない ドライバの場合、名前もしくは疑問符形式のパラメータを他の適当な値に 書き換えることも可能です。

パラメータ

statement

これは対象のデータベースサーバに対して有効な SQL 文でなければなりません。

driver_options

この配列は、このメソッドによって返される PDOStatement オブジェクトに対して 1 もしくはそれ以上の key=>value の組を含みます。 通常、スクロール可能なカーソルを要求するために PDO::ATTR_CURSOR に PDO::CURSOR_SCROLL を設定する場合に使用することになるでしょう。 いくつかのドライバには、準備する際に利用可能なドライバ固有の オプションがあります。

返り値

もしデータベースサーバが正常に文を準備する場合、 PDO->prepare() は PDOStatement オブジェクトを返します。 もしデータベースサーバが文を準備できなかった場合、 PDO->prepare() は **FALSE** を返します。

例

Example#1 名前付きパラメータを用いて SQL ステートメントを準備する

```php
<?php
/* 値の配列を渡してプリペアドステートメントを実行する */
$sql = 'SELECT name, colour, calories
     FROM fruit
     WHERE calories < :calories AND colour = :colour';
$sth = $dbh->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$red = $sth->fetchAll();
$sth->execute(array('calories' => 175, 'colour' => 'yellow'));
$yellow = $sth->fetchAll();
?>
```

Example#2 疑問符パラメータを用いて SQL ステートメントを準備する

```php
<?php
/* 値の配列を渡してプリペアドステートメントを実行する */
$sth = $dbh->prepare('SELECT name, colour, calories
     FROM fruit
     WHERE calories < ? AND colour = ?');
$sth->execute(array(150, 'red'));
$red = $sth->fetchAll();
$sth->execute(array(175, 'yellow'));
$yellow = $sth->fetchAll();
?>
```

参考

- PDO->exec()
- PDO->query()
- PDOStatement->execute()

PDO->query()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDO->query() — SQL ステートメントを実行し、結果セットを PDOStatement オブジェクトとして返す

説明

```
PDO
PDOStatement query ( string $statement )
PDO
PDOStatement query ( string $statement , int $PDO::FETCH_COLUMN , int $colno )
PDO
PDOStatement query ( string $statement , int $PDO::FETCH_CLASS , string $classname , array $ctorargs )
PDO
PDOStatement query ( string $statement , int $PDO::FETCH_INTO , object $object )
```

PDO->query() は、一回の関数コールの中で SQL ステートメントを実行し、このステートメントにより返された 結果セット（ある場合）を PDOStatement オブジェクトとして返します。

複数回発行する必要があるステートメントの場合、 PDO->prepare() で PDOStatement ステートメントを準備し、 PDOStatement->execute() でそのステートメントを 複数回発行する方がより良いパフォーマンスを得られると実感するでしょう。

PDO->query() を次にコールする前に 結果セット内の全てのデータを取得しない場合、そのコールは失敗します。 PDOStatement->closeCursor() をコールし、 次に PDO->query() をコールする前に PDOStatement オブジェクトに関連付けられたリソースを解放してください。

> **注意**: この関数はパラメータをひとつだけしかとらないと記述されていますが、 追加のパラメータを渡すことも可能です。これらのパラメータは、 返される結果のオブジェクトに対して PDOStatement->setFetchMode() をコールするのと同じような扱いになります。

パラメータ

statement

準備、発行する SQL ステートメント。

返り値

PDO->query() は、PDOStatement オブジェクトを返します。

例

Example#1 PDO::query の例

PDO->query()の優れた機能は、 実行に成功したSELECTステートメントにより返されたレコードセットで 反復処理が可能であることです。

```
<?php
function getFruit($conn) {
    $sql = 'SELECT name, colour, calories FROM fruit ORDER BY name';
     foreach ($conn->query($sql) as $row) {
         print $row['NAME'] . "¥t";
         print $row['COLOUR'] . "¥t";
         print $row['CALORIES'] . "¥n";
    }
}
?>
```

上の例の出力は以下となります。

```
apple    red      150
banana   yellow   250
kiwi     brown    75
lemon    yellow   25
orange   orange   300
pear     green    150
watermelon       pink     90
```

参考

- PDO->exec()
- PDO->prepare()
- PDOStatement->execute()

PDO->quote()

(PHP 5 >= 5.1.0, PECL pdo:0.2.1-1.0.3)

PDO->quote() — クエリ用の文字列をクオートする

説明

```
PDO
string quote ( string $string [, int $parameter_type ] )
```

PDO->quote() は入力文字列のまわりに引用符を付け（必要であれば）、 入力文字列にあるシングルクオートをエスケープします。その場合、 構成しているドライバに適したクオート形式が使用されます。

この関数を SQL の構築に使用する場合、 SQL ステートメントにユーザーの入力値を埋め込むための PDO->quote() を使用する代わりに、 バインドパラメータを用いて SQL を準備するための PDO->prepare() を使用することが強く推奨されます。 バインドパラメータを用いるプリペアドステートメントは、 補間されたクエリした場合に比べ、 移植性や利便性に優れ、SQL インジェクションに対する免疫力を持つだけでなく しばしばより高速で、サーバやクライアント側でコンパイル済みの形式でクエリを キャッシュする事が可能です。

全ての PDO ドライバがこのメソッドを実装しているわけではありません（たとえば PDO_ODBC などの例があります）。 代わりにプリペアドステートメントを使用することを検討してください。

パラメータ

string

クオートされる文字列を指定します。

parameter_type

クオートするスタイルを変更するため、 ドライバにデータ型のヒントを提供します。 デフォルト値は、*PDO::PARAM_STR* です。

返り値

理論上安全なクオートされた *SQL* ステートメントの文字列を返します。 ドライバがこの方法での引用符付けをサポートしていない場合は **FALSE** を返します。

例

Example#1 通常の文字列をクオートする

```
<?php
$conn = new PDO('sqlite:/home/lynn/music.sql3');

/* 単純な文字列 */
$string = 'Nice';
print "Unquoted string: $string¥n";
print "Quoted string: " . $conn->quote($string) . "¥n";
?>
```

上の例の出力は以下となります。

```
Unquoted string: Nice
Quoted string: 'Nice'
```

Example#2 危険な文字列をクオートする

```
<?php
$conn = new PDO('sqlite:/home/lynn/music.sql3');

/* 危険な文字列 */
$string = 'Naughty ¥' string';
print "Unquoted string: $string¥n";
print "Quoted string:" . $conn->quote($string) . "¥n";
?>
```

上の例の出力は以下となります。

```
Unquoted string: Naughty ' string
Quoted string: 'Naughty '' string'
```

Example#3 複雑な文字列をクオートする

```
<?php
$conn = new PDO('sqlite:/home/lynn/music.sql3');

/* 複雑な文字列 */
$string = "Co'mpl''ex ¥"st'¥"ring";
print "Unquoted string: $string¥n";
print "Quoted string: " . $conn->quote($string) . "¥n";
?>
```

上の例の出力は以下となります。

```
Unquoted string: Co'mpl''ex "st'"ring
Quoted string: 'Co''mpl''''ex "st'"ring'
```

参考

- PDO->prepare()
- PDOStatement->execute()

PDO->rollBack()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->rollBack() — トランザクションをロールバックする

説明

PDO
bool **rollBack** (void)

PDO->beginTransaction() によって開始された 現在のトランザクションをロールバックします 有効なトランザクションがない場合にこのメソッドをコールすると エラーになります。

データベースがオートコミットモードに設定されている場合、 この関数はトランザクションをロールバックした後に オートコミットモードを元に戻します。

MySQL を含むいくつかのデータベースでは、DROP TABLE や CREATE TABLE のようなデータベース定義言語 (DDL) ステートメントがトランザクション中に 発行される場合、暗黙的なコミットが自動的に発行されます。 この暗黙的なコミットにより、そのトランザクション境界で 他のあらゆる変更をロールバックすることができなくなるでしょう。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 トランザクションをロールバックする

以下の例は、トランザクションを開始し、 変更をロールバックする前にデータベースを修正する 2 つのステートメントを発行します。 しかしながら MySQL では、DROP TABLE ステートメントは 自動的にトランザクションをコミットするので、 トランザクション中のどの変更もロールバックされません。

```
<?php
/* トランザクションを開始する。オートコミットがオフになる */
$dbh->beginTransaction();

/* データベーススキーマとデータを変更する */
$sth = $dbh->exec("DROP TABLE fruit");
$sth = $dbh->exec("UPDATE dessert
    SET name = 'hamburger'");

/* ミスに気づき、変更をロールバックする */
$dbh->rollBack();

/* データベース接続はオートコミットモードに戻る */
?>
```

参考

- PDO->beginTransaction()
- PDO->commit()

PDO->setAttribute()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDO->setAttribute() — 属性を設定する

説明

PDO
bool **setAttribute** (int $attribute , mixed $value)

データベースハンドルの属性を設定します。 利用可能な通常の属性は以下の一覧の通りです。いくつかのドライバでは、 ドライバ固有の属性を使用することが可能です。

- PDO::ATTR_CASE: 強制的にカラム名を指定したケースにする
  - PDO::CASE_LOWER: 強制的にカラム名を小文字にする
  - PDO::CASE_NATURAL: データベースドライバによって返されるカラム名をそのままにする
  - PDO::CASE_UPPER: 強制的にカラム名を大文字にする
- PDO::ATTR_ERRMODE: エラーレポート
  - PDO::ERRMODE_SILENT: エラーコードのみ設定する
  - PDO::ERRMODE_WARNING: E_WARNING を発生させる
  - PDO::ERRMODE_EXCEPTION: exceptions を投げる
- PDO::ATTR_ORACLE_NULLS (Oracle だけでなく、全てのドライバで利用可能): NULL とから文字列の変換
  - PDO::NULL_NATURAL: 変換しない
  - PDO::NULL_EMPTY_STRING: 空文字は **NULL** に変換される
  - PDO::NULL_TO_STRING: NULL は空文字に変換される
- PDO::ATTR_STRINGIFY_FETCHES: フェッチする際、数値を文字列に変換する。bool を必要とする
- PDO::ATTR_STATEMENT_CLASS: PDOStatement に由来するユーザーが提供するステートメントクラスを設定する。 永続的な PDO インスタンスは使用できない。 array(string classname, array(mixed constructor_args)) を必要とする。
- PDO::ATTR_AUTOCOMMIT (OCI, Firebird そして MySQL で利用可能): それぞれの単一文で自動コミットするかどうか。
- PDO::MYSQL_ATTR_USE_BUFFERED_QUERY (MySQL で利用可能): バッファされたクエリを使用する。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

PDOStatement->bindColumn()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->bindColumn() — カラムを PHP 変数にバインドする

説明

PDOStatement
bool **bindColumn** (mixed $column , mixed &$param [, int $type])

PDOStatement->bindColumn() は、 クエリからの結果セット中にあるカラムにバインドされた特定の値を 取得するための準備をします。 PDOStatement->fetch() もしくは PDOStatement->fetchAll() がコールされる度に、カラムにバインドされた全ての変数は更新されます。

> **注意**: カラムに関する情報はステートメントが実行されるまで常に PDO から利用できないため、移植可能なアプリケーションでは PDOStatement->execute() の 後に この関数をコールするようにしてください。

パラメータ

column

結果セット中のカラム番号（1 から始まる）を指定します。 カラム名を使用する場合、 ドライバによって返されるカラムの大文字小文字が一致する必要が あることをご承知おきください。

param

カラムがバインドされる PHP 変数名を指定します。

type

パラメータのデータ型を PDO::PARAM_* 定数で指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 バインドした結果セットを PHP 変数に出力する

PHP 変数にバインドしている結果セットのカラムは、 アプリケーションで利用可能な各行に含まれるデータを作成するための 効果的な方法です。 以下のサンプルは、どうやって PDO が様々なオプションやを理にかなったデフォルト値を用いて カラムをバインド、取得するかを例示しています。

```
<?php
function readData($dbh) {
  $sql = 'SELECT name, colour, calories FROM fruit';
  try {
    $stmt = $dbh->prepare($sql);
    $stmt->execute();

    /* カラム番号によってバインドする */
    $stmt->bindColumn(1, $name);
    $stmt->bindColumn(2, $colour);

    /* カラム名によってバインドする */
    $stmt->bindColumn('calories', $cals);

     while ($row = $stmt->fetch(PDO::FETCH_BOUND)) {
       $data = $name . "¥t" . $colour . "¥t" . $cals . "¥n";
       print $data;
     }
  }
  catch (PDOException $e) {
    print $e->getMessage();
  }
}
readData($dbh);
?>
```

上の例の出力は以下となります。

```
apple     red       150
banana    yellow    175
kiwi      green     75
orange    orange    150
mango     red       200
strawberry        red       25
```

参考

- PDOStatement->execute()
- PDOStatement->fetch()
- PDOStatement->fetchAll()
- PDOStatement->fetchColumn()

PDOStatement->bindParam()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->bindParam() — 指定された変数名にパラメータをバインドする

説明

PDOStatement
bool **bindParam** (mixed $parameter , mixed &$variable [, int $data_type [, int $length [, mixed $driver_options]]])

準備された SQL ステートメント中で、 対応する名前もしくは疑問符プレースホルダにパラメータをバインドします。 PDOStatement->bindValue() と異なり、 変数は参照としてバインドされ、 PDOStatement->execute() がコールされたときのみ評価されます。

ほとんどのパラメータは入力パラメータです。つまり、クエリを構築する際、 パラメータは読み込み専用で使用されます。 いくつかのドライバは、出力パラメータとしてデータを返す ストアドプロシージャの実行をサポートしており、 またいくつかのドライバは、データを渡し更新された値を受け取る、 といった入出力パラメータもサポートしています。

パラメータ

parameter

パラメータ ID を指定します。名前付けされたプレースホルダを使った文に 対しては、:name 形式のパラメータ名となります。 疑問符プレースホルダを使った文に対しては、1 から始まるパラメータの 位置となります。

variable

SQL ステートメントパラメータにバインドする PHP 変数名を指定します。

data_type

パラメータに対して PDO::PARAM_* 定数を使った明示的なデータ型を指定します。 デフォルトは PHP のネイティブ型です。 ストアドプロシージャからの INOUT パラメータの場合、 data_type パラメータに PDO::PARAM_INPUT_OUTPUT ビットを設定するためにビット OR を使用してください。

length

データ型の長さを指定します。パラメータがストアドプロシージャからの OUT パラメータであることを示す場合、 明示的に長さを設定しなければなりません。

driver_options

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 名前付けされたプレースホルダを用いてプリペアドステートメントを実行する

```php
<?php
/* バインドされた PHP 変数によってプリペアドステートメントを実行する */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindParam(':calories', $calories, PDO::PARAM_INT);
$sth->bindParam(':colour', $colour, PDO::PARAM_STR, 12);
$sth->execute();
?>
```

Example#2 疑問符プレースホルダを用いてプリペアドステートメントを実行する

```php
<?php
/* バインドされた PHP 変数によってプリペアドステートメントを実行する */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < ? AND colour = ?');
$sth->bindParam(1, $calories, PDO::PARAM_INT);
$sth->bindParam(2, $colour, PDO::PARAM_STR, 12);
$sth->execute();
?>
```

Example#3 INOUT パラメータを持つストアドプロシージャをコールする

```php
<?php
/* INOUT パラメータを持つストアドプロシージャをコールする */
$colour = 'red';
$sth = $dbh->prepare('CALL puree_fruit(?)');
$sth->bindParam(1, $colour, PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT, 12);
$sth->execute();
print("After pureeing fruit, the colour is: $colour");
?>
```

参考

- PDO->prepare()
- PDOStatement->execute()
- PDOStatement->bindValue()

PDOStatement->bindValue()

(PHP 5 >= 5.1.0, PECL pdo:1.0-1.0.3)

PDOStatement->bindValue() — 値をパラメータにバインドする

説明

PDOStatement
bool **bindValue** (mixed $parameter , mixed $value [, int $data_type])

プリペアドステートメントで使用する SQL 文の中で、 対応する名前あるいは疑問符のプレースホルダに値をバインドします。

パラメータ

parameter

パラメータ ID。名前つきプレースホルダを使用する プリペアドステートメントの場合は、 :name 形式のパラメータ名となります。 疑問符プレースホルダを使用するプリペアドステートメントの場合は、 1 から始まるパラメータの位置となります。

value

パラメータにバインドする値。

data_type

パラメータに対して PDO::PARAM_* 定数を使った明示的なデータ型を指定します。 デフォルトは PHP のネイティブ型です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 名前付けされたプレースホルダを用いてプリペアドステートメントを実行する

```
<?php
/* バインドされた PHP 変数によってプリペアドステートメントを実行する */
$calories  = 150;
$colour  = 'red';
$sth  = $dbh->prepare('SELECT  name,  colour,  calories
     FROM fruit
      WHERE  calories  <  :calories  AND  colour  =  :colour');
$sth->bindValue(':calories', $calories, PDO::PARAM_INT);
$sth->bindValue(':colour', $colour, PDO::PARAM_STR);
$sth->execute();
?>
```

Example#2 疑問符プレースホルダを用いてプリペアドステートメントを実行する

```
<?php
/* バインドされた PHP 変数によってプリペアドステートメントを実行する */
$calories  = 150;
$colour  = 'red';
$sth  = $dbh->prepare('SELECT  name,  colour,  calories
     FROM fruit
     WHERE calories  <  ?  AND colour  =  ?');
$sth->bindValue(1, $calories, PDO::PARAM_INT);
$sth->bindValue(2, $colour, PDO::PARAM_STR);
$sth->execute();
?>
```

参考

- PDO->prepare()
- PDOStatement->execute()
- PDOStatement->bindParam()

PDOStatement->closeCursor()

(PHP 5 >= 5.1.0, PECL pdo:0.9-1.0.3)

PDOStatement->closeCursor() — カーソルを閉じてステートメントを再実行できるようにする

説明

PDOStatement
bool **closeCursor** (void)

PDOStatement->closeCursor() は、 他の SQL ステートメントを発行できるようにサーバへの接続を解放しますが、 ステートメントは再実行可能な状態のまま残されます。

このメソッドは以前に実行された PDOStatement オブジェクトが行をまだフェッチしていない場合に PDOStatement オブジェクトの実行をサポートしていないデータベースドライバに対して有用です。 もし使用しているデータベースドライバがこの制限を受ける場合、 out-of-sequence エラーが出力されます。

PDOStatement->closeCursor() は、 オプションのドライバ固有のメソッド（最大の効率を得るため）もしくはドライバ固有の関数がインストールされていない場合の汎用的な PDO フォールバックとして実装されています。 汎用的な PDO フォールバックは、PHP スクリプト中に以下のようなコードを書くことと意味的に等価です。

```php
<?php
do {
    while ($stmt->fetch())
        ;
    if (!$stmt->nextRowset())
        break;
} while (true);
?>
```

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 PDOStatement->closeCursor() の例

以下の例では、PDOStatement オブジェクト $stmt は複数の行を返しますが、このアプリケーションは先頭行のみフェッチし、 PDOStatement オブジェクトをフェッチしていない行がある状態のままにします。 このアプリケーションがが全てのデータベースドライバで動作するよう、PDOStatement オブジェクト $otherStmt を実行する前に $stmt に対して PDOStatement->closeCursor() の呼び出しを挿入しています。

```php
<?php
/* PDOStatement オブジェクトを生成する */
$stmt = $dbh->prepare('SELECT foo FROM bar');

/* 第二の PDOStatement オブジェクトを生成する */
$otherStmt = $dbh->prepare('SELECT foobaz FROM foobar');

/* 最初の文を実行する */
$stmt->execute();

/* 結果から先頭行のみフェッチする */
$stmt->fetch();

/* 続く closeCursor() のコールはいくつかのドライバでは必要となる */
$stmt->closeCursor();

/* ここで第二の文を実行することができる */
$otherStmt->execute();
?>
```

参考

- PDOStatement->execute()

PDOStatement->columnCount()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDOStatement->columnCount() — 結果セット中のカラム数を返す

説明

PDOStatement
int **columnCount** (void)

PDOStatement オブジェクトに相当する結果セットにあるカラム数を返すために PDOStatement->columnCount() を使用します。

もし PDOStatement オブジェクトが PDO->query() から返された場合、カラム数は直ちに利用可能です。

もし PDOStatement オブジェクトが PDO->prepare() から返された場合、正確なカラム数は PDOStatement->execute() を実行するまで利用可能になりません。

返り値

PDOStatement オブジェクトに相当する結果セットにあるカラム数を返します。 もし結果セットがなければ、PDOStatement->columnCount() は 0 を返します。

例

Example#1 カラム数を数える

この例は、結果セットがある場合とない場合で、 PDOStatement->columnCount() がどのように動作するかを例示しています。

```php
<?php
$dbh = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');

$sth = $dbh->prepare("SELECT name, colour FROM fruit");

/* (存在しない) 結果セットにあるカラム数を数える */
$colcount = $sth->columnCount();
print("Before execute(), result set has $colcount columns (should be 0)¥n");

$sth->execute();

/* 結果セットにあるカラム数を数える */
$colcount = $sth->columnCount();
print("After execute(), result set has $colcount columns (should be 2)¥n");

?>
```

上の例の出力は以下となります。

```
Before execute(), result set has 0 columns (should be 0)
After execute(), result set has 2 columns (should be 2)
```

参考

- PDO->prepare()
- PDOStatement->execute()
- PDOStatement->rowCount()

PDOStatement->errorCode()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->errorCode() — 文ハンドラにおける直近の操作に関連する SQLSTATE を取得する

説明

```
PDOStatement
string errorCode ( void )
```

返り値

PDOStatement->errorCode() は PDOStatement オブジェクトを用いて実行された操作に対する エラーコードを取得することを除き、PDO->errorCode() と等価です。

例

Example#1 SQLSTATE コードを取得する

```
<?php
/* エラーを発生させる -- BONES テーブルは存在しない */
$err = $dbh->prepare('SELECT skull FROM bones');
$err->execute();

echo "¥nPDOStatement::errorCode(): ";
print $err->errorCode();
?>
```

上の例の出力は以下となります。

```
PDOStatement::errorCode(): 42S02
```

参考

- PDO->errorCode()
- PDO->errorInfo()
- PDOStatement->errorInfo()

PDOStatement->errorInfo()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->errorInfo() — 文ハンドラにおける直近の操作に関連する拡張エラー情報を取得する

説明

```
PDOStatement
array errorInfo ( void )
```

返り値

PDOStatement->errorInfo() は、 この文ハンドラによって実行された直近の操作に関するエラー情報を 配列として返します。この配列は次のフィールドを含みます。

要素	情報
0	SQLSTATE エラーコード (これは、ANSI SQL 標準で定義された英数 5 文字の ID)
1	ドライバ固有のエラーコード
2	ドライバ固有のエラーメッセージ

例

Example#1 DB2 データベースに対する PDO_ODBC 接続の errorInfo() フィールドを表示する

```
<?php
/* エラーを発生させる -- BONES テーブルは存在しない */
```

```
$sth = $dbh->prepare('SELECT skull FROM bones');
$sth->execute();

echo "¥nPDOStatement::errorInfo():¥n";
$arr = $sth->errorInfo();
print_r($arr);
?>
```

上の例の出力は以下となります。

```
PDOStatement::errorInfo():
Array
(
    [0] => 42S02
    [1] => -204
     [2] => [IBM][CLI Driver][DB2/LINUX] SQL0204N  "DANIELS.BONES" is an undefined name.  SQLSTATE=42704
)
```

参考

- PDO->errorCode()
- PDO->errorInfo()
- PDOStatement->errorCode()

PDOStatement->execute()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->execute() — プリペアドステートメントを実行する

説明

PDOStatement
bool **execute** ([array $input_parameters])

プリペアドステートメントを実行します。もし、プリペアドステートメントが パラメータマーカを含む場合、次のいずれかを行わなければなりません。

- パラメータマーカに PHP 変数をバインドするため PDOStatement->bindParam() をコールする。 関連づけされたパラメータマーカがあれば、バインドされた変数は入力値を渡す もしくは出力値を受け取ります。
- あるいは、入力専用のパラメータ値の配列を渡す

パラメータ

input_parameters

実行される SQL 文の中のバインドパラメータと同数の要素からなる、 値の配列。

ひとつのパラメータに対して複数の値をバインドすることはできません。 例えば、IN() 句の中のひとつのパラメータに対して 2 つの値をバインドすることはできません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 バインド変数を伴うプリペアドステートメントの実行

```
<?php
/* バインド変数を伴うプリペアドステートメントの実行 */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindParam(':calories', $calories, PDO::PARAM_INT);
$sth->bindParam(':colour', $colour, PDO::PARAM_STR, 12);
$sth->execute();
?>
```

Example#2 入力値の配列を伴うプリペアドステートメントの実行（名前つきパラメータ）

```
<?php
/* 入力値の配列を伴うプリペアドステートメントの実行 */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->execute(array(':calories' => $calories, ':colour' => $colour));
?>
```

Example#3 入力値の配列を伴うプリペアドステートメントの実行（プレースホルダ）

```
<?php
/* 入力値の配列を伴うプリペアドステートメントの実行 */
```

```
$calories  = 150;
$colour  = 'red';
$sth  = $dbh->prepare('SELECT  name,  colour,  calories
     FROM  fruit
     WHERE  calories  <  ?  AND  colour  =  ?');
$sth->execute(array($calories,  $colour));
?>
```

Example#4 疑問符プレースホルダを伴うプリペアドステートメントの実行

```
<?php
/* バインド変数を伴うプリペアドステートメントの実行 */
$calories  = 150;
$colour  = 'red';
$sth  = $dbh->prepare('SELECT  name,  colour,  calories
     FROM  fruit
     WHERE  calories  <  ?  AND  colour  =  ?');
$sth->bindParam(1,  $calories,  PDO::PARAM_INT);
$sth->bindParam(2,  $colour,  PDO::PARAM_STR,  12);
$sth->execute();
?>
```

参考

- PDO->prepare()
- PDOStatement->bindParam()
- PDOStatement->fetch()
- PDOStatement->fetchAll()
- PDOStatement->fetchColumn()

PDOStatement->fetch()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->fetch() — 結果セットから次の行を取得する

説明

PDOStatement
mixed **fetch** ([int $fetch_style [, int $cursor_orientation [, int $cursor_offset]]])

PDOStatementオブジェクトに関連付けられた結果セットから1行取得します。 fetch_style パラメータは、PDO がその行をどの様に返すかを決定します。

パラメータ

fetch_style

次のレコードを呼び出し元に返す方法を制御します。 この値は、PDO::FETCH_* 定数のどれかで、 PDO::FETCH_BOTHがデフォルトです。

- PDO::FETCH_ASSOC: は、結果セットに 返された際のカラム名で添字を付けた配列を返します。
- PDO::FETCH_BOTH (デフォルト): 結果セットに返された際のカラム名と 0 で始まるカラム番号で添字を付けた配列を返します。
- PDO::FETCH_BOUND: **TRUE** を返し、結果セットのカラムの値を PDOStatement->bindColumn() メソッドでバインドされた PHP 変数に代入します。
- PDO::FETCH_CLASS: 結果セットのカラムがクラス内の名前付けされたプロパティに マッピングされている、要求されたクラスの新規インスタンスを返します。 fetch_style に PDO::FETCH_CLASSTYPE が 含まれている場合 (例: PDO::FETCH_CLASS | PDO::FETCH_CLASSTYPE) は、最初のカラムの値から クラス名を決定します。
- PDO::FETCH_INTO: 結果セットのカラムがクラス内の名前付けされたプロパティに マッピングされている要求された既存インスタンスを更新します。
- PDO::FETCH_LAZY: PDO::FETCH_BOTH とPDO::FETCH_OBJの 組合せで、オブジェクト変数名を作成します。
- PDO::FETCH_NUM: 結果セットに返された際の 0 から始まるカラム番号を添字とする配列を返します。
- PDO::FETCH_OBJ: 結果セットに返された際のカラム名と同名のプロパティを有する 匿名のオブジェクトを返します。

cursor_orientation

スクロール可能なカーソルを表す PDOStatement オブジェクトの場合、 この値により呼び出し側に返される行を定義します。 この値は、PDO::FETCH_ORI_* 定数のどれかと する必要があり、 PDO::FETCH_ORI_NEXT がデフォルトとなっています。 PDOStatement に対してスクロール可能なカーソルを要求するためには、 PDO->prepare() を用いて SQL ステートメントを 準備する際、 PDO::CURSOR_SCROLL に PDO::ATTR_CURSOR 属性を設定する必要があります。

offset

スクロール可能なカーソルを表すPDOStatementオブジェクトの場合で、 cursor_orientationパラメータが PDO::FETCH_ORI_ABSに設定された場合、この値により 取得される結果セットの行の絶対位置が指定されます。

スクロール可能なカーソルを表すPDOStatementオブジェクトの場合で、 cursor_orientationパラメータが PDO::FETCH_ORI_RELに設定された場合、この値は、 PDOStatement->fetch() がコールされる前のカーソルの 位置を基準として取得する行の位置を指定します。

返り値

この関数が成功した場合の返り値は、取得形式によって異なります。 失敗した場合は常に **FALSE** を返します。

例

Example#1 異なる取得方法で行を取得する

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* Exercise PDOStatement::fetch styles */
print("PDO::FETCH_ASSOC: ");
print("Return next row as an array indexed by column name¥n");
$result = $sth->fetch(PDO::FETCH_ASSOC);
print_r($result);
print("¥n");

print("PDO::FETCH_BOTH: ");
print("Return next row as an array indexed by both column name and number¥n");
$result = $sth->fetch(PDO::FETCH_BOTH);
print_r($result);
print("¥n");

print("PDO::FETCH_LAZY: ");
print("Return next row as an anonymous object with column names as properties¥n");
$result = $sth->fetch(PDO::FETCH_LAZY);
print_r($result);
print("¥n");

print("PDO::FETCH_OBJ: ");
print("Return next row as an anonymous object with column names as properties¥n");
$result = $sth->fetch(PDO::FETCH_OBJ);
print $result->NAME;
print("¥n");
?>
```

上の例の出力は以下となります。

```
PDO::FETCH_ASSOC: Return next row as an array indexed by column name
Array
(
    [NAME] => apple
    [COLOUR] => red
)

PDO::FETCH_BOTH: Return next row as an array indexed by both column name and number
Array
(
    [NAME] => banana
    [0] => banana
    [COLOUR] => yellow
    [1] => yellow
)

PDO::FETCH_LAZY: Return next row as an anonymous object with column names as properties
PDORow Object
(
    [NAME] => orange
    [COLOUR] => orange
)

PDO::FETCH_OBJ: Return next row as an anonymous object with column names as properties
kiwi
```

Example#2 スクロール可能なカーソルで行を取得する

```
<?php
function readDataForwards($dbh) {
  $sql = 'SELECT hand, won, bet FROM mynumbers ORDER BY BET';
  try {
    $stmt = $dbh->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_SCROLL));
    $stmt->execute();
    while ($row = $stmt->fetch(PDO::FETCH_NUM, PDO::FETCH_ORI_NEXT)) {
      $data = $row[0] . "¥t" . $row[1] . "¥t" . $row[2] . "¥n";
      print $data;
    }
    $stmt = null;
  }
  catch (PDOException $e) {
    print $e->getMessage();
  }
}
function readDataBackwards($dbh) {
  $sql = 'SELECT hand, won, bet FROM mynumbers ORDER BY bet';
  try {
    $stmt = $dbh->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_SCROLL));
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_NUM, PDO::FETCH_ORI_LAST);
    do {
      $data = $row[0] . "¥t" . $row[1] . "¥t" . $row[2] . "¥n";
      print $data;
    } while ($row = $stmt->fetch(PDO::FETCH_NUM, PDO::FETCH_ORI_PRIOR));
    $stmt = null;
  }
  catch (PDOException $e) {
    print $e->getMessage();
  }
}

print "Reading forwards:¥n";
readDataForwards($conn);
```

```
print "Reading backwards:¥n";
readDataBackwards($conn);
?>
```

上の例の出力は以下となります。

```
Reading forwards:
21    10    5
16    0     5
19    20    10

Reading backwards:
19    20    10
16    0     5
21    10    5
```

参考

- PDO->prepare()
- PDOStatement->execute()
- PDOStatement->fetchAll()
- PDOStatement->fetchColumn()
- PDOStatement->fetchObject()
- PDOStatement->setFetchMode()

PDOStatement->fetchAll()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->fetchAll() — 全ての結果行を含む配列を返す

説明

```
PDOStatement
array fetchAll ([ int $fetch_style [, int $column_index [, array $ctor_args ]]] )
```

パラメータ

fetch_style

PDOStatement->fetch() に文章化されているような 返される配列の内容を制御します。デフォルトは PDO_FETCH_BOTH です。

結果セットから単一カラムの全ての値を含む配列を返す場合、 PDO::FETCH_COLUMN を指定します。 column-index パラメータにどのカラムを返すかを 指定することができます。

結果セットから単一カラムの一意な値のみ取得する場合、 PDO::FETCH_UNIQUE をビット OR した PDO::FETCH_COLUMN を指定します。

指定したカラムの値によってグループ化した連想配列を返す場合、 PDO::FETCH_GROUP をビット OR した PDO::FETCH_COLUMN を指定します。

column_index

fetch_style が PDO::FETCH_COLUMN の場合に、 ここで指定した (0 からはじまる) カラム番号の値を返します。 デフォルトは 0 です。

ctor_args

独自のクラスコンストラクタへの引数。

返り値

PDOStatement->fetchAll() は、 結果セットに残っている全ての行を含む配列を返します。 この配列は、カラム値の配列 もしくは各カラム名に対応するプロパティを持つオブジェクトをして 各行を表します。

大きな結果セットをフェッチするためにこのメソッドを使用することは、 システムとネットワークリソースに大量の要求を行うことになります。 PHP で全てのデータ処理と操作を行うよりも、データベースサーバ側で 結果セットを操作することを検討してください。例えば、PHP で処理を行う前に SQL で WHERE 句や SORT BY 句を使用し、結果を制限することです。

例

Example#1 結果セットに残っている全ての行をフェッチする

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* 結果セットに残っている全ての行をフェッチする */
print("Fetch all of the remaining rows in the result set:¥n");
$result = $sth->fetchAll();
print_r($result);
?>
```

上の例の出力は以下となります。

```
Fetch all of the remaining rows in the result set:
Array
(
    [0] => Array
```

```
        (
              [NAME] => pear
              [0] => pear
              [COLOUR] => green
              [1] => green
        )

    [1] => Array
        (
              [NAME] => watermelon
              [0] => watermelon
              [COLOUR] => pink
              [1] => pink
        )

)
```

Example#2 結果セットから単一カラムの全ての値を取得する

以下の例は、 SQL ステートメント自身が行毎に複数のカラムを返す場合において、 どのように結果セットから単一カラムの全ての値を取得するかを例示しています。

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* 最初のカラムの全ての値を取得する */
$result = $sth->fetchAll(PDO::FETCH_COLUMN, 0);
var_dump($result);
?>
```

上の例の出力は以下となります。

```
Array(3)
(
    [0] =>
     string(5) => apple
    [1] =>
     string(4) => pear
    [2] =>
     string(10) => watermelon
)
```

Example#3 単一カラムによる全ての値のグループ化

以下の例は、どのように結果セット中の特定のカラムの値によって グループ化された連想配列を返すかを例示しています。 その配列は 3 つのキーを有します。値 apple、 pear は異なる 2 つの異なる色を有する配列として返され、 一方 watermelon は 1 つの色のみ有する配列として返されます。

```
<?php
$insert = $dbh->prepare("INSERT INTO fruit(name, colour) VALUES (?, ?)");
$insert->execute('apple', 'green');
$insert->execute('pear', 'yellow');

$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* 最初のカラムの値によってグループ化する */
var_dump($sth->fetchAll(PDO::FETCH_COLUMN|PDO::FETCH_GROUP));
?>
```

上の例の出力は以下となります。

```
array(3) {
  ["apple"]=>
  array(2) {
    [0]=>
     string(5) "green"
    [1]=>
     string(3) "red"
  }
   ["pear"]=>
  array(2) {
    [0]=>
     string(5) "green"
    [1]=>
     string(6) "yellow"
  }
   ["watermelon"]=>
  array(1) {
    [0]=>
     string(5) "green"
  }
}
```

参考

- PDO->query()
- PDOStatement->fetch()

- PDOStatement->fetchColumn()
- PDO->prepare()
- PDOStatement->setFetchMode()

PDOStatement->fetchColumn()

(PHP 5 >= 5.1.0, PECL pdo:0.9-1.0.3)

PDOStatement->fetchColumn() — 結果セットの次行から単一カラムを返す

説明

PDOStatement
string **fetchColumn** ([int $column_number])

結果セットの次行から単一カラムを返します。 行がもうない場合には **FALSE** を返します。

パラメータ

column_number

行から処理したい 0 で始まるカラム番号を指定します。 何も値が与えられない場合、 PDOStatement->fetchColumn() は最初のカラムをフェッチします。

返り値

PDOStatement->fetchColumn() は結果セットの次行から単一カラムを返します。

警告

PDOStatement->fetchColumn() を使用してデータを処理する場合、同一行から 他のカラムを返す方法はありません。

例

Example#1 次行の最初のカラムを返す

```
<?php
$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* 結果セットの次行から最初のカラムをフェッチする */
print("Fetch the first column from the next row in the result set:¥n");
$result = $sth->fetchColumn();
print("name = $result¥n");

print("Fetch the second column from the next row in the result set:¥n");
$result = $sth->fetchColumn(1);
print("colour = $result¥n");
?>
```

上の例の出力は以下となります。

```
Fetch the first column from the next row in the result set:
name = lemon
Fetch the second column from the next row in the result set:
colour = red
```

参考

- PDO->query()
- PDOStatement->fetch()
- PDOStatement->fetchAll()
- PDO->prepare()
- PDOStatement->setFetchMode()

PDOStatement->fetchObject()

(PHP 5 >= 5.1.0, PECL pdo:0.2.4-1.0.3)

PDOStatement->fetchObject() — 次の行を取得し、それをオブジェクトとして返す

説明

PDOStatement
mixed **fetchObject** ([string $class_name [, array $ctor_args]])

次の行を取得し、それをオブジェクトとして返します。この関数は、 PDOStatement->fetch() で **PDO::FETCH_CLASS** あるいは **PDO::FETCH_OBJ** を指定することの代替関数となります。

パラメータ

class_name

作成されるクラスの名前。デフォルトは stdClass。

ctor_args

この配列の要素がコンストラクタに渡されます。

返り値

*カラム名に対応するプロパティを保持する、要求されたクラスの インスタンスを返します。エラーが発生した場合は **FALSE** を返します。*

参考

- *PDOStatement->fetch()*

PDOStatement->getAttribute()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDOStatement->getAttribute() — 文の属性を取得する

説明

PDOStatement
mixed ***getAttribute*** *(int $attribute)*

文の属性を取得します。現時点で共通の属性は存在しませんが、 ドライバ固有の属性のみ存在します。

- *PDO::ATTR_CURSOR_NAME (Firebird と ODBC 固有): UPDATE ... WHERE CURRENT OF に対するカーソル名を取得する*

返り値

属性の値を返します。

参考

- *PDO->getAttribute()*
- *PDO->setAttribute()*
- *PDOStatement->setAttribute()*

PDOStatement->getColumnMeta()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDOStatement->getColumnMeta() — 結果セットのカラムに対するメタデータを返す

説明

PDOStatement
array ***getColumnMeta*** *(int $column)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

結果セットの 0 から始まるカラムに対するメタデータを連想配列で取得します。

警告

PDO ドライバの全てが PDOStatement->getColumnMeta() をサポートしているわけではありません。

パラメータ

column

結果セットの 0 から始まるカラムを指定します。

返り値

1つのカラムについてのメタデータに相当する次の値を含んだ連想配列を返します。

カラムのメタデータ

名前	値
native_type	カラム値を表現するために使用される PHP のネイティブ型
driver:decl_type	データベースにおけるカラム値を表現するために使用される SQL 型。 もし、結果セットのカラムが関数から返される場合、 この値は PDOStatement->getColumnMeta() よって返されません。
flags	このカラムにセットされているあらゆるフラグ
name	データベースによって返されるこのカラムの名前
table	データベースによって返されるこのカラムのテーブル名
len	カラム長。浮動小数点数以外の型については通常 *-1* となる。
precision	カラムの数値精度。浮動小数点数以外の型については通常 *0* となる。
pdo_type	*PDO::PARAM_** 定数によって表現されるカラムの型

*要求されたカラムが結果セットに存在しない、 もしくは結果セットが存在しない場合、**FALSE** を返します。*

変更履歴

バージョン	説明
5.2.3	*table* フィールド

例

Example#1 カラムのメタデータを取得する

以下の例は、PDO_SQLITE ドライバでの関数 (COUNT) によって生成された単一のカラムに対するメタデータの 取得結果を表示します。

```
<?php
$select = $DB->query('SELECT COUNT(*) FROM fruit');
$meta = $select->getColumnMeta(0);
var_dump($meta);
?>
```

上の例の出力は以下となります。

```
array(6) {
  ["native_type"]=>
  string(7) "integer"
  ["flags"]=>
  array(0) {
  }
  ["name"]=>
  string(8) "COUNT(*)"
  ["len"]=>
  int(-1)
  ["precision"]=>
  int(0)
  ["pdo_type"]=>
  int(2)
}
```

参考

- PDOStatement->columnCount()
- PDOStatement->rowCount()

PDOStatement->nextRowset()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDOStatement->nextRowset() — 複数の行セットを返す文ハンドラで次の行セットに移動する

説明

```
PDOStatement
bool nextRowset ( void )
```

*いくつかのデータベースサーバは、1つ以上の行セット（結果セットとしても知られる）を返すストアドプロシージャをサポートしています。 **PDOStatement::nextRowSet()** により、2 番目以降の PDOStatement オブジェクトに関連する行セットにアクセスすることができます。 それぞれの行セットは、 前の行セットと異なるカラムセットを含むことができます。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 ストアドプロシージャから複数の行セットをフェッチする

以下のサンプルは、3つの行セットを返すストアドプロシージャの コールの仕方を示しています。 PDOStatement->nextRowset() を併用した do / while ループを使用しており、行セットが返されなくなったとき false を返しループを終了します。

```
<?php
$sql = 'CALL multiple_rowsets()';
$stmt = $conn->query($sql);
$i = 1;
do {
    $rowset = $stmt->fetchAll(PDO::FETCH_NUM);
    if ($rowset) {
        printResultSet($rowset, $i);
    }
    $i++;
} while ($stmt->nextRowset());

function printResultSet(&$rowset, $i) {
    print "Result set $i:¥n";
    foreach ($rowset as $row) {
        foreach ($row as $col) {
            print $col . "¥t";
        }
        print "¥n";
    }
    print "¥n";
}
?>
```

上の例の出力は以下となります。

```
Result set 1:
apple     red
banana    yellow

Result set 2:
orange    orange     150
banana    yellow     175

Result set 3:
lime      green
apple     red
banana    yellow
```

参考

- PDOStatement->columnCount()
- PDOStatement->execute()
- PDOStatement->getColumnMeta()
- PDO->query()

PDOStatement->rowCount()

(PHP 5 >= 5.1.0, PECL pdo:0.1-1.0.3)

PDOStatement->rowCount() — 直近の SQL ステートメントによって作用した行数を返す

説明

PDOStatement
int **rowCount** (void)

PDOStatement->rowCount() は 相当する PDOStatement オブジェクトによって実行された 直近の DELETE, INSERT, UPDATE 文によって作用した行数を返します。

関連する PDOStatement によって実行された直近の SQL ステートメントが SELECT 文の場合、いくつかのデータベースは文によって返された 行数を返すかも知れません。しかしながら、 この振る舞いは全てのデータベースで保証されていません。 さまざまな場所で使用するアプリケーションでは、 これに頼ってはいけません。

返り値

行の数を返します。

例

Example#1 削除された行数を返す

PDOStatement->rowCount() は DELETE, INSERT, UPDATE 文によって作用した行数を返します。

```
<?php
/* FRUIT テーブルから全ての行を削除する */
$del = $dbh->prepare('DELETE FROM fruit');
$del->execute();

/* 削除された行数を返す */
print("Return number of rows that were deleted:¥n");
$count = $del->rowCount();
print("Deleted $count rows.¥n");
```

```
?>
```

上の例の出力は以下となります。

```
Deleted 9 rows.
```

Example#2 SELECT 文によって返された行をカウントする

ほとんどのデータベースでは、PDOStatement->rowCount() は SELECT 文によって作用した行数を返しません。代わりに、PDO->query() を使って 意図する SELECT 文として同様の述部を持つ SELECT COUNT(*) 文を発行し、PDOStatement->fetchColumn() を使って返される行数を取得することができます。 そうすることで、アプリケーションは正しい動作をすることができます。

```
<?php
$sql = "SELECT COUNT(*) FROM fruit WHERE calories > 100";
if ($res = $conn->query($sql)) {

    /* SELECT 文にマッチする行数をチェックする */
  if ($res->fetchColumn() > 0) {

        /* 実際の SELECT 文を発行し、結果を処理する */
         $sql = "SELECT name FROM fruit WHERE calories > 100";
        foreach ($conn->query($sql) as $row) {
            print "Name: " . $row['NAME'] . "¥n";
         }
    }
   /* 行がマッチしなかった場合 -- 他に何かをする */
  else {
      print "No rows matched the query.";
    }
}

$res = null;
$conn = null;
?>
```

上の例の出力は以下となります。

```
apple
banana
orange
pear
```

参考

- PDOStatement->columnCount()
- PDOStatement->fetchColumn()
- PDO->query()

PDOStatement->setAttribute()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDOStatement->setAttribute() — 文の属性を設定する

説明

PDOStatement
bool **setAttribute** (int $attribute , mixed $value)

文の属性を設定します。現時点で共通の属性は存在しませんが、 ドライバ固有の属性のみ存在します。

- PDO::ATTR_CURSOR_NAME (Firebird と ODBC 固有): UPDATE ... WHERE CURRENT OF に対するカーソル名を取得する

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- PDO->getAttribute()
- PDO->setAttribute()
- PDOStatement->getAttribute()

PDOStatement->setFetchMode()

(PHP 5 >= 5.1.0, PECL pdo:0.2-1.0.3)

PDOStatement->setFetchMode() — この文に対するデフォルトのフェッチモードを設定する

説明

```
PDOStatement
bool setFetchMode ( int $mode )
PDOStatement
bool setFetchMode ( int $PDO::FETCH_COLUMN , int $colno )
PDOStatement
bool setFetchMode ( int $PDO::FETCH_CLASS , string $classname , array $ctorargs )
PDOStatement
bool setFetchMode ( int $PDO::FETCH_INTO , object $object )
```

パラメータ

mode

フェッチモードは、PDO::FETCH_* 定数の 1 つでなければなりません。

返り値

成功時は 1、失敗時は **FALSE** を返します。

例

Example#1 フェッチモードを設定する

以下の例は PDOStatement->setFetchMode() がどの様に PDOStatementオブジェクトに対するデフォルトのフェッチモードを 変更するかを例示しています。

```
<?php
$sql = 'SELECT name, colour, calories FROM fruit';
try {
  $stmt = $dbh->query($sql);
  $result = $stmt->setFetchMode(PDO::FETCH_NUM);
  while ($row = $stmt->fetch()) {
    print $row[0] . "¥t" . $row[1] . "¥t" . $row[2] . "¥n";
  }
}
catch (PDOException $e) {
  print $e->getMessage();
}
?>
```

上の例の出力は以下となります。

```
apple    red      150
banana   yellow   250
orange   orange   300
kiwi     brown    75
lemon    yellow   25
pear     green    150
watermelon       pink     90
```

目次

- PDOStatement->getAttribute() — 文の属性を取得する
- PDOStatement->getColumnMeta() — 結果セットのカラムに対するメタデータを返す
- PDOStatement->nextRowset() — 複数の行セットを返す文ハンドラで次の行セットに移動する
- PDOStatement->rowCount() — 直近の SQL ステートメントによって作用した行数を返す
- PDOStatement->setAttribute() — 文の属性を設定する
- PDOStatement->setFetchMode() — この文に対するデフォルトのフェッチモードを設定する

Phar アーカイブストリームおよびクラス

導入

phar 拡張モジュールは、 phar ストリームラッパーおよび Phar クラスを提供し、 PHP アーカイブ (phar) ファイルを操作することができます。 Phar クラスを使用すると、phar ファイルの作成や展開、 そしてその中身を順に処理することなどができます。

PHP アーカイブファイル (Phar) は複数のファイルをまとめたもので、 外部からその中のファイルを透過的に実行することができます。 Java の jar アーカイブファイルと似たものです。 phar アーカイブを使用すると、 PHP アプリケーションをひとつのファイルにまとめて配布できるようになります。 このファイルは、変更したり展開したりすることなくそのまま外部から実行できます。 phar アーカイブは、tar や zip のアーカイブと同様に、 ファイルを保存するために使用することもできます。 Phar は圧縮もサポートしています。 zlib 拡張モジュールが存在する場合には gzip を、そして bz2 拡張モジュールが存在する場合には bzip2 を使用します。 さらに、SPL 拡張モジュールが存在する場合には、 順次処理やその他の機能が使用可能となります。 md5 や sha1 による Phar の署名の検証は ネイティブでサポートされており、アーカイブの整合性を確かめることができます。

Phar アーカイブのネイティブな実装は PEAR パッケージ » PHP_Archive であり、この拡張モジュールの実装はこれと非常によく似ています。 しかし Phar 拡張モジュールのほうが、より多くの機能をサポートしています。 PHP_Archive はより柔軟に Phar を作成することができます。また PHP_Archive_Manager クラスのような便利なデバッグツールも用意されています。 一方 Phar 拡張モジュールは、Phar の中身の順次処理や 配列形式でのアクセス、そして中身の変更などが シンプルなインターフェイスでできるようになります。 PHP_Archive は、Phar 拡張モジュールあるいは PHP_Archive のいずれかでシームレスに処理できる Phar アーカイブの作成をサポートしています。 一方、Phar 拡張モジュールは、 Phar 拡張モジュールで動作するアーカイブを作成するよう設計されています。 さらに Phar 拡張モジュールは、INI 設定 allow_url_include あるいは allow_url_fopen が無効になっていても動作します。一方 PHP_Archive で (Phar 拡張モジュールを使用せずに) 作成したアーカイブは動作しません。

要件

Phar を使用するには PHP 5.2.0 以降が必要です。また、 Phar ファイルの中身に対して配列形式でアクセスしたり順に処理したりするには SPL 拡張モジュールが PHP に組み込まれている必要があります。 phar ストリームを使用するには、 これは不要です。

オプションとして、zlib 拡張モジュールおよび bzip2 拡張モジュールを有効にしておくと、 圧縮された phar をサポートすることができます。

インストール手順

Windows 用のバイナリは » http://snaps.php.net/ にあります。インストールするには、php_phar.dll をダウンロードして、php.ini の extension_dir ディレクティブで指定されたフォルダに配置します。 その後、php.ini に extension=php_phar.dll を追加してこれを有効にし、ウェブサーバを再起動します。

```
extension_dir=c:/php5/exts/
extension=php_phar.dll
```

Linux、BSD その他 *nix 系 の場合は、次の手順でコンパイルします。

- 次の手順か、
  - pecl install phar で、PECL/phar のインストーラを実行します。
  - phar.so を、ビルド時に表示されたディレクトリから php.ini の extension_dir ディレクトリにコピーします。
  - extension=phar.so を php.ini に追加します。

  あるいはこちらの手順を実行します。
  - 次のようにして、php.ini へのパスを設定します。 pecl config-set php_ini /path/to/php.ini
  - pecl install phar で、PECL/phar のインストーラを実行します。
- ウェブサーバを再起動し、php.ini の設定を読み込ませます。

  注意: **開発バージョン** 現在は、PECL/phar の 安定版 はまだありません。 アルファ版 の PECL/phar をインストールするには pecl install phar-alpha を実行します。

ヒント

PEAR コマンドを使用しない PECL/phar のコンパイル

pecl install phar を使用すると、 PECL/phar を自動的にダウンロードし、インストールします。しかし、 tar ボールを » PECL からダウンロードすることもできます。tar ボールを展開したルートディレクトリで phpize && ./configure --enable-phar && make を実行すると phar.so が出来上がります。 ビルドしたものを、上のようにしてインストールします。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/phar.

実行時設定

php.ini の設定により動作が変化します。

ファイルシステムおよびストリームの設定オプション

名前	デフォルト	変更の可否	変更履歴
phar.readonly	"1"	PHP_INI_ALL	
phar.require_hash	"1"	PHP_INI_ALL	
phar.extract_list	""	PHP_INI_ALL	phar 1.0.0 以降で使用可能

以下に設定ディレクティブに関する簡単な説明を示します。

phar.readonly boolean

このオプションを使用すると、phar ストリームや Phar オブジェクトによる Phar アーカイブの作成や変更ができなくなります。 この設定は、実運用環境では常に有効にしておくべきです。 phar 拡張モジュールのこの機能は、 サーバ上に他のセキュリティ上の脆弱性があった場合に php ベースのウィルスを直接作成されてしまうことにつながります。

> **注意**: この設定は、セキュリティ上の理由により php.ini でしか解除できません。 phar.readonly を php.ini で無効にした場合は、 ユーザがスクリプト内で phar.readonly の有効/無効を切りかえることができます。 phar.readonly を php.ini で有効にした場合は、 スクリプト内でこれを "再度有効にする" ことはできますが、無効にすることはできません。

phar.require_hash boolean

このオプションを使用すると、署名つき (現在サポートしているのは MD5 および SHA1) の Phar アーカイブのみをオープンするようになります。 署名を含まない Phar アーカイブの処理はできません。

> **注意**: この設定は、セキュリティ上の理由により php.ini でしか解除できません。 phar.require_hash を php.ini で無効にした場合は、 ユーザがスクリプト内で phar.require_hash の有効/無効を切りかえることができます。 phar.require_hash を php.ini で有効にした場合は、 スクリプト内でこれを "再度有効にする" ことはできますが、無効にすることはできません。

phar.extract_list string

phar アーカイブへのフルパスあるいはそのエイリアスと、 展開されたファイルの場所との関連付けを行います。 パラメータの書式は name=archive,name2=archive2 のようになります。 phar ファイルをディスクに展開して、 展開したファイル群へのマッパーとして phar を扱えるようになります。 これを行うのは、たとえばパフォーマンスを向上させたい場合や phar のデバッグを行いたい場合などです。

Example#1 phar.extract_list の使用例

```
php.ini で、次のように記述します
phar.extract_list = archive=/full/path/to/archive/,arch2=/full/path/to/arch2
<?php
include "phar://archive/content.php";
include "phar://arch2/foo.php";
?>
```

リソース型

Phar 拡張モジュールで使用するリソースは phar ストリームで、これにより、phar 内のファイルへの透過的なアクセスが可能となります。 Phar ファイルのフォーマットについては ここ で説明されています。

リソース型は定義されていません。

定義済みクラス

- Phar
- PharFileInfo
- PharException

Phar アーカイブの使用法: 導入

Phar アーカイブは Java の JAR アーカイブと似た概念のものですが、 PHP アプリケーションで使用する際に必要な機能をより柔軟に使用できるよう改良しています。 Phar アーカイブを使用すると、PHP アプリケーションやライブラリを ひとつのファイルにまとめて配布できるようになります。 Java における JAR アーカイブの実装とは異なり、 PHP の Phar アーカイブを処理したり実行したりするには外部のツールは不要です。 Phar アーカイブ形式のアプリケーションは、その他の PHP アプリケーションとまったく同様に扱えます。

```
php coolapplication.phar
```

Phar アーカイブ形式のライブラリを使用する方法も、 その他の通常の PHP ライブラリとまったく同じです。

```
<?php
include 'coollibrary.phar';
?>
```

Phar アーカイブを非常に便利なものとしている機能が phar ストリームラッパーです。 詳細については こちら で説明します。このストリームラッパーを使用すると、 phar 内の個々のファイルに対して、 まるで通常のファイルシステム上にあるのと同じような感覚でアクセスできます。 phar ストリームラッパーは、 ファイルに対する読み書きや、ディレクトリに対する opendir() をすべてサポートしています。

```
<?php
include 'phar://coollibrary.phar/internal/file.php';
header('Content-type: image/jpeg');
// phar にアクセスするには、フルパスあるいはエイリアスを使用します
echo file_get_contents('phar:///fullpath/to/coollibrary.phar/images/wow.jpg');
?>
```

Phar 拡張モジュールで提供されているもうひとつの機能が Phar クラスです。 これにより、Phar アーカイブ内のファイルに対して まるでそれが連想配列であるかのようにアクセスできるようになります。 またその他の機能も用意されています。Phar クラスについての説明は こちら をご覧ください。

```
<?php
try {
    // 既存の phar をオープンします
    $p = new Phar('coollibrary.phar');
     foreach ($p as $file) {
        // $file は PharFileInfo クラスで、これは SplFileInfo を継承しています
         echo $file->getFileName() . "¥n";
         echo $file . "¥n"; //  内容を表示します
    }
     if (isset($p['internal/file.php'])) {
         var_dump($p['internal/file.php']->getMetaData());
    }

    // 新しい phar の作成 - php.ini で phar.readonly を 0 にしておく必要があります。
     // phar.readonly は、セキュリティ上の理由によってデフォルトで有効になっています。
     // 実際の運用サーバでは、決して Phar を作成する必要はないはずで、
     // 単に実行できるだけでよいはずです。
     if (Phar::canWrite()) {
         $p = new Phar(dirname(__FILE__) . '/newphar.phar', 0, 'newphar.phar');
        // トランザクションを作成します - commit() がコールされるまで、
         // newphar.phar には何も書き込まれません。しかし一時ストレージが必要となります
         $p->begin();
        // 新しいファイルを追加し、その内容を設定します
         $p['file1.txt'] = 'Information';
         $fp = fopen('hugefile.dat', 'rb');
        // ストリームからコピーします
         $p['data/hugefile.dat'] = $fp;
         if (Phar::canCompress()) {
             $p['data/hugefile.dat']->setCompressedGZ();
         }
         $p['images/wow.jpg'] = file_get_contents('images/wow.jpg');
        // ファイル固有のメタデータとして、任意のデータを保存できます
         $p['images/wow.jpg']->setMetaData(array('mime-type' => 'image/jpeg'));
         $p['index.php'] = file_get_contents('index.php');
         $p->setMetaData(array('bootstrap' => 'index.php'));
        // ローダースタブを設定します
         $p->setStub('<?php
$p = new Phar(__FILE__);
$m = $p->getMetaData();
require "phar://" . __FILE__ . "/" . $m["bootstrap"];
__HALT_COMPILER();');
        // phar アーカイブをディスクに保存します
         $p->commit();
     }
} catch (Exception $e) {
     echo 'Phar をオープンできません: ', $e;
}
?>
```

Phar アーカイブの使用法: phar ストリームラッパー

Phar ストリームラッパーは、fopen() による読み込みや 書き込みそして追記、unlink()、stat()、 fstat()、fseek()、rename() そしてディレクトリに対する opendir() といった機能に完全に対応しています。 Phar ストリームラッパーはディレクトリの作成や削除には対応していません。 ファイルは単なるファイルとしてしか格納されず、 ディレクトリを抽象化した概念は存在しないからです。

Phar アーカイブ内の各ファイルの圧縮やファイル単位のメタデータの操作も、 ストリーム上で可能です。

```
<?php
$context = stream_context_create(array('phar' =>
                                    array('compression' => Phar::GZ)),
                                    array('metadata' => array('user' => 'cellog'))));
file_put_contents('phar://my.phar/somefile.php', 0, $context);
?>
```

phar ストリームラッパーは、 リモートファイルを操作することはできません。つまり、INI 設定 allow_url_fopen および allow_url_include が無効になっている場合でも使用できます。

ストリーム操作だけで新しい phar アーカイブをゼロから作成することも可能ですが、 そんな場合は Phar クラスの組み込み機能を使用するほうが便利です。 ストリームラッパーが有用なのは、読み込み操作を行う場合です。

Phar アーカイブの使用法: Phar クラス

Phar クラスは Phar アーカイブの読み込みや操作をサポートしています。 また » RecursiveDirectoryIterator クラスを継承しているため、順次処理も可能です。 ArrayAccess インターフェイスをサポートしているので、 Phar アーカイブ内のファイルに対して、 それがまるで連想配列であるかのようにアクセスすることができます。

注意すべき点は、Phar アーカイブを作成する際には Phar のコンストラクタに フルパスを渡さなければならないということです。 相対パスでは初期化に失敗します。

$p が、次のように作成した Phar オブジェクトであるとしましょう。

```
<?php
$p = new Phar('/path/to/myphar.phar', 0, 'myphar.phar');
?>
```

空の Phar アーカイブが /path/to/myphar.phar に作成されます。もし /path/to/myphar.phar が既に存在する場合は、それを再度オープンします。 リテラル myphar.phar は、エイリアスを表します。 これを用いると、URL で /path/to/myphar.phar を参照する際に次のようにできます。

```
<?php
// これらのふたつの file_get_contents() コールが同等となるのは、
// /path/to/myphar.phar のマニフェストでエイリアス "myphar.phar"
// が明示的に指定されている場合か、先ほどの例のように
// Phar オブジェクトを初期化した場合です。
```

```
$f = file_get_contents('phar:///path/to/myphar.phar/whatever.txt');
$f = file_get_contents('phar://myphar.phar/whatever.txt');
?>
```

新しく作成した Phar オブジェクト $p に対して、次のような処理が可能となります。

- $a = $p['file.php'] とすると、 phar://myphar.phar/file.php の中身を参照する PharFileInfo クラスが作成されます。
- $p['file.php'] = $v とすると、 myphar.phar の中に新しいファイル (phar://myphar.phar/file.php) を作成するか、あるいは同名のファイルを上書きします。 $v には、文字列あるいはファイルポインタのいずれかを指定できます。 ファイルポインタを指定した場合は、その中身全体をもとにして新しいファイルを作成します。
- isset($p['file.php']) とすると、phar://myphar.phar/file.php が myphar.phar の中に存在するかどうかがわかります。
- unset($p['file.php']) とすると、 phar://myphar.phar/file.php を myphar.phar から削除します。

さらに、Phar 固有のメタデータにアクセスするためには Phar オブジェクトを使用することが唯一の方法となります。そのためには Phar->getMetaData() を使用します。また、Phar アーカイブの PHP ローダスタブを設定したり取得したりするための唯一の方法が Phar->getStub() および Phar->setStub() です。 また、Phar アーカイブ全体の圧縮を行うには Phar クラスが必要となります。

Phar オブジェクトの全機能の一覧については、以下で説明します。

PharFileInfo クラスは » SplFileInfo クラスを継承しており、Phar 内のファイルについての Phar 固有の情報 (圧縮情報やメタデータなど) を扱うためのメソッドが追加されています。

Phar ファイルフォーマット

Phar ファイルは、三つあるいは四つの部分から構成されています。

1. スタブ
2. 内容を説明するマニフェスト
3. ファイルの内容
4. [オプション] Phar の整合性を検証するためのシグネチャ

Phar ファイルのスタブ

Phar のスタブは、単純な PHP ファイルです。必要最小限のスタブは、次のようになります。

```
<?php __HALT_COMPILER();
```

スタブには、少なくとも __HALT_COMPILER(); トークンが必要です。 典型的なスタブは、次のように読み込み機能を含んでいます。

```
<?php
Phar::mapPhar();
include 'phar://myphar.phar/index.php';
__HALT_COMPILER();
```

Phar スタブの内容には特に制限はありません。唯一の制約は、最後が __HALT_COMPILER(); でなければならないということです。 PHP の終了タグ

```
?>
```

は含めても含めなくてもかまいません。 しかし、; と終了タグ

```
?>
```

の間にはひとつ以上の空白を含めてはいけません。 もしそうしてしまうと、phar 拡張モジュールは その Phar アーカイブのマニフェストを処理できなくなります。

Phar マニフェストの書式

Phar マニフェストは高度に最適化された書式で、 ファイル単位で圧縮やパーミッションの情報を指定することができ、 さらにファイルのユーザやグループなど、独自に定義したメタデータも含めることができます。 1 バイトをこえる大きさの値はリトルエンディアン形式のバイト順で保存されます。 ただし API バージョンだけは例外です。これは 3 ニブルのデータですが、 歴史的な理由によりビッグエンディアン形式のバイト順で保存されます。

未使用のフラグはすべて将来の使用に備えて予約されています。 したがって、独自の情報を保存するためにそれを使用してはいけません。 特定のファイルについて独自の情報を保存するには、 ファイル単位のメタデータ機能を使用します。

Phar アーカイブマニフェストの基本的なファイルフォーマットは、次のようになります。

グローバル Phar マニフェスト書式

バイト数	説明
4 バイト	マニフェスト全体のバイト長 (最大 1 MB)。
4 バイト	Phar 内のファイル数。
2 バイト	Phar マニフェストの API バージョン (現在は 1.0.0)。
4 バイト	グローバルな Phar ビットマップフラグ。
4 バイト	Phar のエイリアスの長さ。
??	Phar のエイリアス (先ほどの長さに基づきます)。
4 バイト	Phar のメタデータの長さ (存在しない場合は 0)。
??	シリアライズ化された Phar メタデータ。serialize() 形式で格納される。
最低でも 24 * エントリ数ぶんのバイト	各ファイルのエントリ

グローバルな Phar ビットマップフラグ

これが、Phar 拡張モジュールが現在 Phar フラットビットマップとして認識するフラグです。

認識するビットマップ値

値	説明
0x00010000	設定されている場合、この Phar には検証用シグネチャが含まれます。
0x00001000	設定されている場合、この Phar には zlib 圧縮されたファイルが少なくともひとつ含まれます。
0x00002000	設定されている場合、この Phar には bzip 圧縮されたファイルが少なくともひとつ含まれます。

Phar マニフェストのファイルエントリの定義

マニフェスト内の各ファイルについて、次のような情報が含まれます。

Phar マニフェストのファイルエントリ

バイト数	説明
4 バイト	ファイル名の長さを表すバイト数。
??	ファイル名 (先ほど指定した長さになります)。
4 バイト	圧縮前のファイルサイズを表すバイト数。
4 バイト	ファイルの Unix タイムスタンプ。
4 バイト	圧縮後のファイルサイズを表すバイト数。
4 バイト	圧縮前のファイルの CRC32 チェックサム。
4 バイト	ファイル固有のビットマップフラグ。
4 バイト	シリアライズされたファイルのメタデータの長さ (存在しない場合は *0*)。
??	シリアライズされたファイルのメタデータ。serialize() の形式で格納される。

ファイル固有のビットマップ値として認識される値は次のとおりです。

認識されるビットマップ値

値	説明
0x000001FF	これらのビットは、ファイルの特定のパーミッションを定義するために予約されています。 このパーミッションは fstat() で用いられ、 ファイルを展開する際に特定のパーミッションを指定することができます。
0x00001000	設定されている場合、このファイルは zlib で圧縮されています。
0x00002000	設定されている場合、このファイルは bzip で圧縮されています。

Phar のシグネチャの書式

シグネチャを含む Phar は、常にシグネチャを保持しています。 シグネチャの場所は、 ローダ、マニフェスト、実際のファイルの内容に続く Phar アーカイブの最後の部分です。 現在サポートしているシグネチャのフォーマットは MD5 および SHA1 の二種類です。

シグネチャのフォーマット

バイト長	説明
16 あるいは 20 バイト	実際のシグネチャ。SHA1 の場合は 20 バイト、 MD5 の場合は 16 バイトとなります。
4 バイト	シグネチャのフラグ。*0x0001* は MD5 シグネチャ、そして *0x0002* は SHA1 シグネチャを表します。
4 バイト	*GBMB* という固定値で、 シグネチャが存在することを表します。

Phar::apiVersion

(PECL phar:1.0.0-1.2.1)

Phar::apiVersion — API のバージョンを返す

説明

```
string Phar::apiVersion ( void )
```

phar ファイルフォーマットの API バージョンを返します。 phar を作成する際にこのバージョンが使用されます。 Phar 拡張モジュールが読み込みをサポートしているのは、 バージョン 1.0.0 以降の API です。

パラメータ

返り値

API のバージョンを、"1.0.0" のような文字列で返します。

例

Example#1 Phar::apiVersion() の例

```
<?php
echo Phar::apiVersion();
?>
```

上の例の出力は以下となります。

```
1.0.0
```

Phar::canCompress

(PECL phar:1.0.0-1.2.1)

Phar::canCompress — phar 拡張モジュールが zlib あるいは bzip2 による圧縮をサポートしているかどうかを返す

説明

bool **Phar::canCompress** ([int $type])

これは、圧縮されたファイルを含む phar を読み込む前に、 圧縮が可能かどうかを調べるために使用します。

パラメータ

type

Phar::GZ あるいは Phar::BZ2 のいずれかを指定し、指定した圧縮アルゴリズムでの圧縮が可能かどうかを調べます。

返り値

圧縮/展開 が可能な場合に **TRUE**、そうでない場合に **FALSE** を返します。

例

Example#1 Phar::canCompress() の例

```
<?php
if (Phar::canCompress()) {
    echo file_get_contents('phar://compressedphar.phar/internal/file.txt');
} else {
    echo '圧縮はできません';
}
?>
```

参考

- PharFileInfo::isCompressed()

Phar::canWrite

(PECL phar:1.0.0-1.2.1)

Phar::canWrite — phar 拡張モジュールが phar の書き込みや作成をサポートしているかどうかを返す

説明

bool **Phar::canWrite** (void)

このメソッドは、書き込みアクセスが php.ini の phar.readonly で無効にされているかどうかを調べます。

パラメータ

返り値

書き込みが可能な場合に **TRUE**、そうでない場合に **FALSE** を返します。

例

Example#1 Phar::canWrite() の例

```
<?php
if (Phar::canWrite()) {
    file_put_contents('phar://myphar.phar/file.txt', 'hi there');
}
?>
```

参考

- phar.readonly

Phar->compressAllFilesBZIP2

(PECL phar:1.0.0-1.2.1)

Phar->compressAllFilesBZIP2 — 現在の Phar アーカイブ内のすべてのファイルを Bzip2 で圧縮する

説明

bool **Phar->compressAllFilesBZIP2** (void)

このメソッドは、Phar アーカイブ内のすべてのファイルを bzip2 で圧縮します。この機能を使用するには bzip2 拡張モジュールが有効になっていなければなりません。 また、すでに gzip で圧縮されているファイルを処理するためには、 まず gzip を伸張してから bzip2 で再圧縮するため、zlib 拡張モジュールが有効になっていなければなりません。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

エラー / 例外

INI 設定 phar.readonly が on の場合、bzip2 拡張モジュールが使用できない場合、あるいは gzip で圧縮されたファイルがあるのに zlib 拡張モジュールが使用できない場合に BadMethodCallException をスローします。

例

Example#1 Phar->compressAllFilesBZIP2() の例

```
<?php
$p = new Phar('/path/to/my.phar', 0, 'my.phar');
$p['myfile.txt'] = 'hi';
$p['myfile2.txt'] = 'hi';
foreach ($p as $file) {
    var_dump($file->getFileName());
    var_dump($file->isCompressed());
    var_dump($file->isCompressedBZIP2());
    var_dump($file->isCompressedGZ());
}
$p->compressAllFilesBZIP2();
foreach ($p as $file) {
    var_dump($file->getFileName());
    var_dump($file->isCompressed());
    var_dump($file->isCompressedBZIP2());
    var_dump($file->isCompressedGZ());
}
?>
```

上の例の出力は以下となります。

```
string(10) "myfile.txt"
bool(false)
bool(false)
bool(false)
string(11) "myfile2.txt"
bool(false)
bool(false)
bool(false)
string(10) "myfile.txt"
bool(true)
bool(true)
bool(false)
string(11) "myfile2.txt"
bool(true)
bool(true)
bool(false)
```

参考

- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- Phar->uncompressAllFiles()
- Phar->compressAllFilesGZ()

Phar->compressAllFilesGZ

(PECL phar:1.0.0-1.2.1)

Phar->compressAllFilesGZ — 現在の Phar アーカイブ内のすべてのファイルを Gzip で圧縮する

説明

bool **Phar->compressAllFilesGZ** (void)

このメソッドは、Phar アーカイブ内のすべてのファイルを gzip で圧縮します。この機能を使用するには zlib 拡張モジュールが有効になっていな

ければなりません。 また、すでに bzip2 で圧縮されているファイルを処理するためには、 まず bzip2 を伸張してから gzip で再圧縮するため、bzip2 拡張モジュールが有効になっていなければなりません。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

エラー / 例外

INI 設定 phar.readonly が on の場合、zlib 拡張モジュールが使用できない場合、あるいは bzip2 で圧縮されたファイルがあるのに bzip2 拡張モジュールが使用できない場合に BadMethodCallException をスローします。

例

Example#1 Phar->compressAllFilesGZ() の例

```
<?php
$p = new Phar('/path/to/my.phar', 0, 'my.phar');
$p['myfile.txt'] = 'hi';
$p['myfile2.txt'] = 'hi';
foreach ($p as $file) {
    var_dump($file->getFileName());
    var_dump($file->isCompressed());
    var_dump($file->isCompressedBZIP2());
    var_dump($file->isCompressedGZ());
}
$p->compressAllFilesGZ();
foreach ($p as $file) {
    var_dump($file->getFileName());
    var_dump($file->isCompressed());
    var_dump($file->isCompressedBZIP2());
    var_dump($file->isCompressedGZ());
}
?>
```

上の例の出力は以下となります。

```
string(10) "myfile.txt"
bool(false)
bool(false)
bool(false)
string(11) "myfile2.txt"
bool(false)
bool(false)
bool(false)
string(10) "myfile.txt"
bool(true)
bool(false)
bool(true)
string(11) "myfile2.txt"
bool(true)
bool(false)
bool(true)
```

参考

- PharFileInfo->getCompressedSize()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- Phar->uncompressAllFiles()
- Phar->compressAllFilesBZIP2()

Phar::__construct

(PECL phar:1.0.0-1.2.1)

Phar::__construct — Phar アーカイブオブジェクトを作成する

説明

void **Phar::__construct** (string $fname [, int $flags [, string $alias]])

パラメータ

fname

既存の Phar アーカイブへのパス。

flags

親クラス RecursiveDirectoryIterator に渡すフラグ。 » SPL RecursiveDirectoryIterator のドキュメント を参照ください。

alias

ストリーム機能をコールする場合に Phar アーカイブが参照するエイリアス。

エラー / 例外

二度コールされた場合に BadMethodCallException、 phar アーカイブがオープンできなかった場合に UnexpectedValueException がスローされます。

例

Example#1 Phar::__construct() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', CURRENT_AS_FILEINFO | KEY_AS_FILENAME,
                  'my.phar');
} catch (UnexpectedValueException $e) {
    die('my.phar をオープンできません');
} catch (BadMethodCallException $e) {
    echo 'これはありえません';
}
// これは動作します
echo file_get_contents('phar://my.phar/example.txt');
// これも動作します
echo file_get_contents('phar:///path/to/my.phar/example.txt');
?>
```

Phar->count

(PECL phar:1.0.0-1.2.1)

Phar->count — Phar アーカイブ内のエントリ（ファイル）の数を返す

説明

int **Phar->count** (void)

パラメータ

返り値

この phar ファイルに含まれるファイルの数を返します。 存在しない場合は 0（数字のゼロ）を返します。

例

Example#1 Phar->count() の例

```
<?php
// 存在しないことを確実にしておきます
@unlink('brandnewphar.phar');
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
} catch (Exception $e) {
    echo 'phar を作成できません', $e;
}
echo 'phar のエントリ数は ' . $p->count() . " です\n";
$p['file.txt'] = 'hi';
echo 'phar のエントリ数は ' . $p->count() . " です\n";
?>
```

上の例の出力は以下となります。

```
phar のエントリ数は 0 です
phar のエントリ数は 1 です
```

Phar->getMetaData

(PECL phar:1.0.0-1.2.1)

Phar->getMetaData — phar アーカイブのメタデータを返す

説明

int **Phar->getMetaData** (void)

パラメータ

返り値

Phar アーカイブのメタデータとして保存されている、シリアライズ可能な任意の PHP 変数を返します。 メタデータが保存されていない場合は **NULL** を返します。

例

Example#1 Phar->getMetaData() の例

```php
<?php
// 確実に消しておきます
@unlink('brandnewphar.phar');
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
} catch (Exception $e) {
    echo 'phar を作成できません:', $e;
}
$p['file.php'] = '<?php echo "hello";';
$p->setMetaData(array('bootstrap' => 'file.php'));
var_dump($p->getMetaData());
?>
```

上の例の出力は以下となります。

```
array(1) {
  ["bootstrap"]=>
  string(8) "file.php"
}
```

参考

- PharFileInfo->setMetaData()

Phar->getModified

(PECL phar:1.0.0-1.2.1)

Phar->getModified — phar が変更されているかどうかを返す

説明

bool **Phar->getModified** (void)

このメソッドを使用すると、phar 内のファイルが 削除されたり変更されたりしているかどうかを判断できます。

パラメータ

返り値

オープンした後でファイルが変更されている場合に **TRUE**、 そうでない場合に **FALSE** を返します。

Phar->getSignature

(PECL phar:1.0.0-1.2.1)

Phar->getSignature — Phar アーカイブの MD5/SHA1 シグネチャを返す

説明

array **Phar->getSignature** (void)

phar アーカイブの検証用シグネチャを、 十六進文字列で返します。

パラメータ

返り値

開いているアーカイブのシグネチャを配列で返します。 キー "hash" にはシグネチャ自体を、そしてキー "hash_type" には "md5" あるいは "sha1" を格納します。 このシグネチャは phar の中身全体から計算したハッシュで、 アーカイブの整合性を検証する際に使用します。 INI 設定 phar.require_hash が true の場合は、すべての phar が有効なシグネチャを持っている必要があります。

Phar->getStub

(PECL phar:1.0.0-1.2.1)

Phar->getStub — Phar アーカイブの PHP ローダーあるいは起動スタブを返す

説明

string **Phar->getStub** (void)

Phar アーカイブが tar や zip のようなその他のアーカイブフォーマットと違う点のひとつに、 アーカイブを展開せずに直接中身を実行できるような仕組みを提供しており、 アプリケーションやライブラリを配布することを考えて設計されているということがあります。 これを実現するために、 Phar アーカイブは起動用ローダーあるいは スタブ を含んでいます。 これは PHP で書かれており、任意の動作を行えます。

警告

スタブの最後は、必ず __HALT_COMPILER(); で終わらなければなりません。 それ以外は有効な Phar アーカイブとして認められません。

返り値

現在の Phar アーカイブの起動用ローダー（スタブ）の内容を文字列で返します。

エラー / 例外

スタブを Phar アーカイブから読み込めない場合に RuntimeException をスローします。

例

Example#1 Phar->getStub() の例

```
$p = new Phar('/path/to/my.phar', 0, 'my.phar');
echo $p->getStub();
echo "==NEXT==¥n";
$p->setStub("<?php
function __autoload($class)
{
    include 'phar://' . str_replace('_', '/', $class);
}
Phar::mapPhar('myphar.phar');
include 'phar://myphar.phar/startup.php';
__HALT_COMPILER();");
echo $p->getStub();
```

上の例の出力は以下となります。

```
<?php __HALT_COMPILER();
==NEXT==
<?php
function __autoload($class)
{
    include 'phar://' . str_replace('_', '/', $class);
}
Phar::mapPhar('myphar.phar');
include 'phar://myphar.phar/startup.php';
__HALT_COMPILER();
```

参考

- Phar->setStub()

Phar->getVersion

(PECL phar:1.0.0-1.2.1)

Phar->getVersion — Phar アーカイブのバージョン情報を返す

説明

string **Phar->getVersion** (void)

オープンした Phar アーカイブの API バージョンを返します。

パラメータ

返り値

オープンしたアーカイブの API バージョンを返します。 これは、読み込んだ phar 拡張モジュールが新しい phar を作成する際に使用する API バージョンとは異なります。各 Phar アーカイブは、 マニフェスト内に API バージョンをハードコードして保持しています。 詳細は Phar ファイルフォーマット のドキュメントを参照ください。

参考

- Phar::apiVersion()

Phar->isBuffering

(PECL phar:1.0.0-1.2.1)

Phar->isBuffering — Phar の書き込み操作がバッファリングされるか、あるいは直接ディスクに書き込まれるかを調べる

説明

bool **Phar->isBuffering** (void)

このメソッドを使用すると、その Phar が変更を即時にディスクに書き込むのか それとも Phar->stopBuffering() をコールしないと変更が保存されないのかがわかります。

Phar の書き込みバッファリングはアーカイブ単位で処理されます。 Phar アーカイブ foo.phar でバッファリングが有効であったとしても、それは Phar アーカイブ bar.phar の変更には影響しません。

例

Example#1 Phar->isBuffering() の例

```
<?php
$p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
$p2 = new Phar('existingphar.phar');
$p['file1.txt'] = 'hi';
var_dump($p->isBuffering());
var_dump($p2->isBuffering());
?>
=2=
<?php
$p->startBuffering();
var_dump($p->isBuffering());
var_dump($p2->isBuffering());
$p->stopBuffering();
?>
=3=
<?php
var_dump($p->isBuffering());
var_dump($p2->isBuffering());
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(false)
=2=
bool(true)
bool(false)
=3=
bool(false)
bool(false)
```

参考

- Phar->startBuffering()
- Phar->stopBuffering()

Phar::loadPhar

(PECL phar:1.0.0-1.2.1)

Phar::loadPhar — 任意の phar アーカイブを、エイリアスを指定して読み込む

説明

mixed **Phar::loadPhar** (string $filename [, string $alias])

これを使用して、外部の Phar アーカイブの内容を読み込むことができます。 phar へのエイリアスを指定することで、その後 phar を使用する場合に短い名前を使用できるので便利です。 また、データのみが含まれており、実行したり インクルードしたりする必要のない PHP スクリプトからなる Phar アーカイブを読み込む際にも便利です。

パラメータ

filename

オープンする phar アーカイブへの完全パスあるいは相対パス。

alias

この phar アーカイブをさす際に使用するエイリアス。 多くの場合は phar アーカイブ内で明示的なエイリアスを指定しており、 このような場合に新しいエイリアスを指定すると、 PharException がスローされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

すでに明示的なエイリアスを持っている phar アーカイブに対してエイリアスを渡すと、 PharException がスローされます。

例

Example#1 Phar::loadPhar() の例

Phar::loadPhar を使用すると、任意の場所にある外部の Phar アーカイブを読み込めます。 一方 Phar::mapPhar は Phar ローダのスタブ内で使用します。

```
<?php
try {
    Phar::loadPhar('/path/to/phar.phar', 'my.phar');
    echo file_get_contents('phar://my.phar/file.txt');
} catch (PharException $e) {
    echo $e;
}
?>
```

参考

- Phar::mapPhar()

Phar::mapPhar

(PECL phar:1.0.0-1.2.1)

Phar::mapPhar — 現在実行されている (phar 形式の) ファイルを読み込み、その内容を登録する

説明

mixed **Phar::mapPhar** ([string $alias [, int $dataoffset]])

この静的メソッドは、直接実行されたときや 他のスクリプトからインクルードされたときに Phar アーカイブのローダスタブ内で使用され、 phar を初期化します。

パラメータ

alias

この phar を参照するエイリアス。

dataoffset

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

PHP から直接コールされなかった場合、 ソースファイル内に __HALT_COMPILER(); トークンがなかった場合、 あるいはファイルを読み込みモードでオープンできなかった場合に PharException をスローします。

例

Example#1 Phar::mapPhar() の例

mapPhar は、phar のローダスタブ内でのみ使用します。 外部の phar をメモリに読み込むには、loadPhar を使用します。

mapPhar を使用する Phar ローダスタブの例は、このようになります。

```
<?php
function __autoload($class)
{
    include 'phar://me.phar/' . str_replace('_', '/', $class) . '.php';
}
try {
    Phar::mapPhar('me.phar');
    include 'phar://me.phar/startup.php';
} catch (PharException $e) {
    echo $e->getMessage();
    die('Phar を初期化できません');
}
__HALT_COMPILER();
```

参考

- Phar::loadPhar()

Phar::offsetExists

(PECL phar:1.0.0-1.2.1)

Phar::offsetExists — ファイルが phar 内に存在するかどうかを調べる

説明

bool **Phar::offsetExists** (string $offset)

これは ArrayAccess インターフェイスを実装したものです。 これにより、Phar アーカイブの内容に対して配列形式の角括弧を使用したアクセスが可能となります。

offsetExists() は、isset() がコールされる際にはいつもコールされます。

パラメータ

offset

Phar 内で探すファイル名 (相対パス)。

返り値

ファイルが phar 内に存在する場合に **TRUE**、しない場合に **FALSE** を返します。

例

Example#1 Phar::offsetExists() の例

```
<?php
$p = new Phar(dirname(__FILE__) . '/my.phar', 0, 'my.phar');
$p['firstfile.txt'] = 'first file';
$p['secondfile.txt'] = 'second file';
// 以下の行では、offsetExists() が間接的にコールされます
var_dump(isset($p['firstfile.txt']));
var_dump(isset($p['nothere.txt']));
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(false)
```

参考

- Phar::offsetGet()
- Phar::offsetSet()
- Phar::offsetUnset()

Phar::offsetGet

(PECL phar:1.0.0-1.2.1)

Phar::offsetGet — 指定したファイルの PharFileInfo オブジェクトを取得する

説明

int **Phar::offsetGet** (string $offset)

これは ArrayAccess インターフェイスを実装したものです。 これにより、Phar アーカイブの内容に対して配列形式の角括弧を使用したアクセスが可能となります。 offsetGet を使用して、Phar アーカイブからファイルを取得します。

パラメータ

offset

Phar 内で探すファイル名 (相対パス)。

返り値

PharFileInfo オブジェクトを返します。 これを使用して、ファイルの内容を順に処理したりファイルの情報を取得したりします。

エラー / 例外

そのファイルが Phar アーカイブ内に存在しない場合に、このメソッドは BadMethodCallException をスローします。

例

Example#1 Phar::offsetGet() の例

ArrayAccess インターフェイスを実装した他のクラスと同様、 offsetGet は [] 演算子を使用した場合に自動的にコールされます。

```
<?php
$p = new Phar(dirname(__FILE__) . '/myphar.phar', 0, 'myphar.phar');
$p['exists.txt'] = "file exists¥n";
try {
    // 自動的に offsetGet() をコールします
    echo $p['exists.txt'];
    echo $p['doesnotexist.txt'];
} catch (BadMethodCallException $e) {
    echo $e;
}
?>
```

上の例の出力は以下となります。

```
file exists
Entry doesnotexist.txt does not exist
```

参考

- Phar::offsetExists()
- Phar::offsetSet()
- Phar::offsetUnset()

Phar::offsetSet

(PECL phar:1.0.0-1.2.1)

Phar::offsetSet — 内部ファイルに、外部ファイルの内容を設定する

説明

void **Phar::offsetSet** (string $offset , string $value)

これは ArrayAccess インターフェイスを実装したものです。 これにより、Phar アーカイブの内容に対して配列形式の角括弧を使用したアクセスが可能となります。 offsetSet を使用して、Phar アーカイブの既存ファイルの変更や 新しいファイルの追加を行います。

パラメータ

offset

Phar 内で変更するファイル名（相対パス）。

value

ファイルの内容。

返り値

値を返しません。

エラー / 例外

phar.readonly に 1 の場合は BadMethodCallException がスローされます。 Phar の変更ができるのは phar.readonly が 0 のときだけだからです。Phar アーカイブへの変更をディスクに書き込む際に何らかのエラーが発生すると PharException がスローされます。

例

Example#1 Phar::offsetSet() の例

offsetSet は直接使用してはいけません。 配列に [] 演算子でアクセスする際に使用されます。

```
<?php
$p = new Phar('/path/to/my.phar', 0, 'my.phar');
try {
    // offsetSet がコールされます
    $p['file.txt'] = 'Hi there';
} catch (Exception $e) {
    echo 'ファイルを変更できません: ', $e;
}
?>
```

参考

- Phar::offsetExists()
- Phar::offsetGet()
- Phar::offsetUnset()

Phar::offsetUnset

(PECL phar:1.0.0-1.2.1)

Phar::offsetUnset — ファイルを phar から削除する

説明

bool **Phar::offsetUnset** (string $offset)

これは ArrayAccess インターフェイスを実装したものです。 これにより、Phar アーカイブの内容に対して配列形式の角括弧を使用したアクセスが可能となります。 offsetUnset を使用して、既存のファイルを削除します。また unset() が使用された場合にもコールされます。

パラメータ

offset

Phar 内で変更するファイル名（相対パス）。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

phar.readonly に 1 の場合は BadMethodCallException がスローされます。 Phar の変更ができるのは phar.readonly が 0 のときだけだからです。Phar アーカイブへの変更をディスクに書き込む際に何らかのエラーが発生すると PharException がスローされます。

例

Example#1 Phar::offsetUnset() の例

```php
<?php
$p = new Phar('/path/to/my.phar', 0, 'my.phar');
try {
    // offsetUnset がコールされ、file.txt が my.phar から削除されます
    unset($p['file.txt']);
} catch (Exception $e) {
    echo 'file.txt を削除できません: ', $e;
}
?>
```

参考

- Phar::offsetExists()
- Phar::offsetGet()
- Phar::offsetSet()

Phar->setMetaData

(PECL phar:1.0.0-1.2.1)

Phar->setMetaData — phar アーカイブのメタデータを設定する

説明

void **Phar->setMetaData** (mixed $metadata)

setMetaData() を使用するのは、 phar アーカイブ全体に関する独自の情報を保存する場合のみに限るべきです。 ファイル固有のメタデータを保存するには PharFileInfo->setMetaData() を使用します。メタデータを使用すると、phar アーカイブの読み込み時のパフォーマンスが劇的に低下します。これは、 メタデータのサイズが大きい場合に顕著になります。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

メタデータの使用例として考えられるのは、 アーカイブの中で最初に実行するファイルはどれかを指定したり、 » PEAR の package.xml のようなマニフェストの場所を指定したりすることです。 しかし、phar アーカイブに関する有用なデータなら何を保存してもかまいません。

パラメータ

metadata

phar アーカイブについての情報を含む、PHP の変数。

例

Example#1 Phar->setMetaData() の例

```php
<?php
// 確実に消しておきます
@unlink('brandnewphar.phar');
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
    $p['file.php'] = '<?php echo "hello"';
    $p->setMetaData(array('bootstrap' => 'file.php'));
    var_dump($p->getMetaData());
} catch (Exception $e) {
    echo 'Could not create and/or modify phar:', $e;
    echo 'phar の作成/変更ができません:', $e;
}
?>
```

上の例の出力は以下となります。

```
array(1) {
  ["bootstrap"]=>
  string(8) "file.php"
}
```

参考

- Phar->getMetaData()

Phar->setStub

(PECL phar:1.0.0-1.2.1)

Phar->setStub — Phar アーカイブの PHP ローダ（あるいは起動スタブ）を設定する

説明

void **Phar->setStub** (string $stub)

このメソッドを使用して、新しい Phar アーカイブに PHP 起動ローダスタブを追加します。あるいは、既存の Phar アーカイブのローダスタブを置き換えます。

Phar アーカイブのローダスタブは、 このようにアーカイブを直接インクルードした際に使用されます。

```
<?php
include 'myphar.phar';
?>
```

ストリームラッパーを使用して次のようにファイルをインクルードした際には、 ローダにはアクセスしません。

```
<?php
include 'phar://myphar.phar/somefile.php';
?>
```

エラー / 例外

php.ini で phar.readonly が有効になっている場合に UnexpectedValueException がスローされます。 変更をディスクに書き込む際に何らかの問題が発生した場合は PharException がスローされます。

例

Example#1 Phar->setStub() の例

```
<?php
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
    $p['a.php'] = '<?php var_dump("Hello");';
    $p->setStub('<?php var_dump("First"); Phar::mapPhar("brandnewphar.phar"); __HALT_COMPILER(); ?>');
     include 'phar://brandnewphar.phar/a.php';
    var_dump($p->getStub());
    $p['b.php'] = '<?php var_dump("World");';
    $p->setStub('<?php var_dump("Second"); Phar::mapPhar("brandnewphar.phar"); __HALT_COMPILER(); ?>');
     include 'phar://brandnewphar.phar/b.php';
    var_dump($p->getStub());
} catch (Exception $e) {
    echo 'brandnewphar.phar での書き込み操作に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
string(5) "Hello"
string(82) "<?php var_dump("First"); Phar::mapPhar("brandnewphar.phar"); __HALT_COMPILER(); ?>"
string(5) "World"
string(83) "<?php var_dump("Second"); Phar::mapPhar("brandnewphar.phar"); __HALT_COMPILER(); ?>"
```

参考

- Phar->getStub()

Phar->startBuffering

(PECL phar:1.0.0-1.2.1)

Phar->startBuffering — Phar の書き込み操作のバッファリングを開始するが、ディスク上の Phar オブジェクトは変更しない

説明

void **Phar->startBuffering** (void)

技術的には必須ではありませんが、**startBuffering()** メソッドを使用すると、大量のファイルを含む Phar アーカイブの作成や変更が非常に高速になります。 通常は、Phar アーカイブ内のファイルが作成あるいは変更されるたびに Phar アーカイブ全体を改めて作り直します。 これによって、アーカイブが常に最新の状態となります。

しかし、単に新しい Phar アーカイブを作りたいときや アーカイブ全体を一度に書き出したいときなどは、これは不要です。 同様に、いくつかの変更を行うときに、 すべての変更がうまくいったことを確認してから実際にディスクに書き込みたい ということもあるでしょう。 これは、ちょうどデータベースのトランザクションの概念と同じようなものです。 **startBuffering()/stopBuffering()** の両メソッドが、このような場合のために用意されています。

Phar の書き込みバッファリングはアーカイブ単位で処理されます。 Phar アーカイブ foo.phar でバッファリングが有効であったとしても、それは Phar アーカイブ bar.phar の変更には影響しません。

例

Example#1 Phar->startBuffering() の例

```
<?php
// 確実に消しておきます
@unlink('brandnewphar.phar');
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
} catch (Exception $e) {
    echo 'phar を作成できません:', $e;
}
echo '新しい phar のエントリ数は ' . $p->count() . " です\n";
$p->startBuffering();
$p['file.txt'] = 'hi';
$p['file2.txt'] = 'there';
$p['file2.txt']->setCompressedGZ();
$p['file3.txt'] = 'babyface';
$p['file3.txt']->setMetaData(42);
$p->setStub("<?php
function __autoload($class)
```

```
{
      include 'phar://myphar.phar/' . str_replace('_', '/', $class) . '.php';
}
Phar::mapPhar('myphar.phar');
include 'phar://myphar.phar/startup.php';
__HALT_COMPILER();");
$p->stopBuffering();
?>
```

参考

- Phar->stopBuffering()
- Phar->isBuffering()

Phar->stopBuffering

(PECL phar:1.0.0-1.2.1)

Phar->stopBuffering — Phar アーカイブへの書き込みリクエストのバッファリングを終了し、変更内容をディスクに書き込む

説明

void **Phar->stopBuffering** (void)

stopBuffering() は、 **startBuffering()** メソッドと組み合わせて使用します。 **startBuffering()** メソッドを使用すると、 大量のファイルを含む Phar アーカイブの作成や変更が非常に高速になります。 通常は、Phar アーカイブ内のファイルが作成あるいは変更されるたびに Phar アーカイブ全体を改めて作り直します。 これによって、アーカイブが常に最新の状態となります。

しかし、単に新しい Phar アーカイブを作りたいときや アーカイブ全体を一度に書き出したいときなどは、これは不要です。 同様に、いくつかの変更を行うときに、 すべての変更がうまくいったことを確認してから実際にディスクに書き込みたい ということもあるでしょう。 これは、ちょうどデータベースのトランザクションの概念と同じようなものです。 **startBuffering()/stopBuffering()** の両メソッドが、このような場合のために用意されています。

Phar の書き込みバッファリングはアーカイブ単位で処理されます。 Phar アーカイブ foo.phar でバッファリングが有効であったとしても、それは Phar アーカイブ bar.phar の変更には影響しません。

エラー / 例外

変更をディスクに書き出す際に何らかの問題が発生した場合は PharException がスローされます。

例

Example#1 Phar->stopBuffering() の例

```
<?php
$p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
$p['file1.txt'] = 'hi';
$p->startBuffering();
var_dump($p->getStub());
$p->setStub("<?php
function __autoload(¥$class)
{
      include 'phar://brandnewphar.phar/' . str_replace('_', '/', ¥$class) . '.php';
}
Phar::mapPhar('brandnewphar.phar');
include 'phar://brandnewphar.phar/startup.php';
__HALT_COMPILER();");
$p->stopBuffering();
var_dump($p->getStub());
?>
```

上の例の出力は以下となります。

```
string(24) "<?php __HALT_COMPILER();"
string(195) "<?php
function __autoload($class)
{
      include 'phar://' . str_replace('_', '/', $class);
}
Phar::mapPhar('brandnewphar.phar');
include 'phar://brandnewphar.phar/startup.php';
__HALT_COMPILER();"
```

参考

- Phar->startBuffering()
- Phar->isBuffering()

Phar->uncompressAllFiles

(PECL phar:1.0.0-1.2.1)

Phar->uncompressAllFiles — 現在の Phar アーカイブ内のすべてのファイルを展開する

説明

bool **Phar->uncompressAllFiles** (void)

このメソッドは、Phar アーカイブ内のすべてのファイルを展開します。 gzip で圧縮されているファイルが存在する場合は、それを展開するために zlib 拡張モジュールが有効でなければなりません。 また、bzip2 で圧縮されているファイルが存在する場合は、それを展開するために bzip2 拡張モジュールが有効でなければなりません。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

エラー / 例外

INI 設定 phar.readonly が on の場合、bzip2 で圧縮されたファイルがあるのに bzip2 拡張モジュールが使用できない場合、 あるいは gzip で圧縮されたファイルがあるのに zlib 拡張モジュールが使用できない場合に BadMethodCallException をスローします。

例

Example#1 Phar->uncompressAllFiles() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $p['myfile2.txt'] = 'hi';
    $p->compressAllFilesGZ();
    foreach ($p as $file) {
        var_dump($file->getFileName());
        var_dump($file->isCompressed());
        var_dump($file->isCompressedBZIP2());
        var_dump($file->isCompressedGZ());
    }
    $p->uncompressAllFiles();
    foreach ($p as $file) {
        var_dump($file->getFileName());
        var_dump($file->isCompressed());
        var_dump($file->isCompressedBZIP2());
        var_dump($file->isCompressedGZ());
    }
} catch (Exception $e) {
    echo 'my.phar での書き込み操作に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
string(10) "myfile.txt"
bool(true)
bool(false)
bool(true)
string(11) "myfile2.txt"
bool(true)
bool(false)
bool(true)
string(10) "myfile.txt"
bool(false)
bool(false)
bool(false)
string(11) "myfile2.txt"
bool(false)
bool(false)
bool(false)
```

参考

- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- Phar->compressAllFilesBZIP2()
- Phar->compressAllFilesGZ()

PharFileInfo->chmod

(PECL phar:1.0.0-1.2.1)

PharFileInfo->chmod — ファイル固有のパーミッションビットを設定する

説明

void **PharFileInfo->chmod** (int $permissions)

PharFileInfo->chmod() は、ファイルパーミッションのうち実行ビットと読み込みビットを設定します。 書き込みビットについては無視されます。書き込みビットは、実行時に INI 設定 phar.readonly に基づいて自動的に設定されます。

パラメータ

permissions

パーミッション (chmod() を参照ください)。

例

Example#1 PharFileInfo->chmod() の例

```
<?php
// make sure it doesn't exist
@unlink('brandnewphar.phar');
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
    $p['file.sh'] = '#!/usr/local/lib/php
     <?php echo "hi"; ?>';
    // 実行ビットを設定します
    $p['file.sh']->chmod(0555);
    var_dump($p['file.sh']->isExecutable());
} catch (Exception $e) {
    echo 'phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(true)
```

PharFileInfo::__construct

(PECL phar:1.0.0-1.2.1)

PharFileInfo::__construct — Phar エントリオブジェクトを作成する

説明

```
void PharFileInfo::__construct ( string $entry )
```

これは直接コールしてはいけません。PharFileInfo オブジェクトを作成するには、 配列へのアクセスを通じて Phar::offsetGet() をコールします。

パラメータ

entry

ファイルを取得するための完全な url。 ファイル my/file.php の情報を phar boo.phar から取得したい場合は、 このエントリは phar://boo.phar/my/file.php となります。

エラー / 例外

__construct() が二度コールされた場合に BadMethodCallException、 phar の URL がおかしかったり phar がオープンできなかったり、 あるいはファイルが phar 内で見つからなかった場合に UnexpectedValueException がスローされます。

例

Example#1 PharFileInfo::__construct() の例

```
try {
      $p = new Phar('/path/to/my.phar', 0, 'my.phar');
      $p['testfile.txt'] = "hi¥nthere¥ndude";
      $file = $p['testfile.txt'];
      foreach ($file as $line => $text) {
          echo "行番号 $line: $text";
     }
     // これも動作します
       $file = new PharFileInfo('phar:///path/to/my.phar/testfile.txt');
      foreach ($file as $line => $text) {
          echo "行番号 $line: $text";
     }
} catch (Exception $e) {
     echo 'Phar 操作に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
行番号 1: hi
行番号 2: there
行番号 3: dude
行番号 1: hi
行番号 2: there
行番号 3: dude
```

PharFileInfo->getCompressedSize

(PECL phar:1.0.0-1.2.1)

PharFileInfo->getCompressedSize — Phar アーカイブ内での実際のファイルの大きさ (圧縮された状態) を返す

説明

int **PharFileInfo->getCompressedSize** (void)

この関数は、Phar アーカイブ内でのファイルの大きさを返します。 圧縮されていないファイルの場合は、getCompressedSize は filesize() と同じ値を返します。

返り値

ディスク上の Phar アーカイブ内のファイルのバイト数を返します。

例

Example#1 PharFileInfo->getCompressedSize() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    echo $file->getCompressedSize();
} catch (Exception $e) {
    echo 'my.phar での書き込み操作に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
2
```

参考

- PharFileInfo->isCompressed()

PharFileInfo->getCRC32

(PECL phar:1.0.0-1.2.1)

PharFileInfo->getCRC32 — CRC32 コードを返すか、CRC がチェックできない場合に例外をスローする

説明

int **PharFileInfo->getCRC32** (void)

Phar アーカイブ内のファイルの crc32() チェックサムを返します。

返り値

Phar アーカイブ内のファイルの crc32() チェックサムを返します。

エラー / 例外

ファイルがまだ CRC32 で検証されていない場合に BadMethodCallException をスローします。 通常、これはありえません。CRC の検証は、 読み込みモードあるいは書き込みモードでファイルをオープンする際に行われるからです。

例

Example#1 PharFileInfo->getCRC32() の例

```
<?php
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    echo $file->getCRC32();
} catch (Exception $e) {
    echo 'my.phar での書き込み操作に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
3633523372
```

PharFileInfo->getMetaData

(PECL phar:1.0.0-1.2.1)

PharFileInfo->getMetaData — ファイルとともに保存されている、ファイル固有のメタデータを返す

説明

int **PharFileInfo->getMetaData** (void)

Phar アーカイブ内で、このファイル用のマニフェストに保存されたメタデータを返します。

パラメータ

返り値

ファイルのメタデータとして保存されている、シリアライズ可能な任意の PHP 変数を返します。 メタデータが保存されていない場合は **NULL** を返します。

例

Example#1 PharFileInfo->getMetaData() の例

```
<?php
// 確実に消しておきます
@unlink('brandnewphar.phar');
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
    $p['file.txt'] = 'hello';
    $p['file.txt']->setMetaData(array('user' => 'bill', 'mime-type' => 'text/plain'));
    var_dump($p['file.txt']->getMetaData());
} catch (Exception $e) {
    echo 'brandnewphar.phar を作成/変更できません: ', $e;
}
?>
```

上の例の出力は以下となります。

```
array(2) {
  ["user"]=>
  string(4) "bill"
  ["mime-type"]=>
  string(10) "text/plain"
}
```

参考

- PharFileInfo->setMetaData()

PharFileInfo->getPharFlags

(PECL phar:1.0.0-1.2.1)

PharFileInfo->getPharFlags — Phar ファイルエントリのフラグを返す

説明

int **PharFileInfo->getPharFlags** (void)

この関数は、Phar のマニフェスト内のフラグセットを返します。 現在の実装では、これは常に 0 を返します。

返り値

Phar フラグを返します（現在の実装では、常に 0 となります）。

例

Example#1 PharFileInfo->getPharFlags() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    var_dump($file->getPharFlags());
} catch (Exception $e) {
    echo 'my.phar を作成/変更できません: ', $e;
}
?>
```

上の例の出力は以下となります。

```
int(0)
```

PharFileInfo->isCompressed

(PECL phar:1.0.0-1.2.1)

PharFileInfo->isCompressed — エントリが圧縮されているかどうかを調べる

説明

bool **PharFileInfo->isCompressed** (void)

これは、Phar アーカイブ内のファイルが Gzip あるいは Bzip2 で圧縮されているかどうかを返します。

返り値

Phar アーカイブ内のファイルが圧縮されている場合に **TRUE**、 そうでない場合に **FALSE** を返します。

例

Example#1 PharFileInfo->isCompressed() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $p['myfile2.txt'] = 'hi';
    $p['myfile2.txt']->setCompressedGZ();
    $file = $p['myfile.txt'];
    $file2 = $p['myfile2.txt'];
    var_dump($file->isCompressed());
    var_dump($file2->isCompressed());
} catch (Exception $e) {
    echo 'my.phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(true)
```

参考

- PharFileInfo->getCompressedSize()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressedBZIP2()

PharFileInfo->isCompressedBZIP2

(PECL phar:1.0.0-1.2.1)

PharFileInfo->isCompressedBZIP2 — エントリが bzip2 で圧縮されているかどうかを調べる

説明

bool **PharFileInfo->isCompressedBZIP2** (void)

これは、Phar アーカイブ内のファイルが Bzip2 で圧縮されているかどうかを返します。

返り値

Phar アーカイブ内のファイルが Bzip2 で圧縮されている場合に **TRUE**、 そうでない場合に **FALSE** を返します。

例

Example#1 PharFileInfo->isCompressedBZIP2() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $p['myfile2.txt'] = 'hi';
    $p['myfile3.txt'] = 'hi';
    $p['myfile2.txt']->setCompressedGZ();
    $p['myfile3.txt']->setCompressedBZIP2();
    $file = $p['myfile.txt'];
    $file2 = $p['myfile2.txt'];
    $file3 = $p['myfile3.txt'];
    var_dump($file->isCompressedBZIP2());
    var_dump($file2->isCompressedBZIP2());
    var_dump($file3->isCompressedBZIP2());
} catch (Exception $e) {
    echo 'my.phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(false)
```

```
bool(true)
```

参考

- PharFileInfo->getCompressedSize()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()

PharFileInfo->isCompressedGZ

(PECL phar:1.0.0-1.2.1)

PharFileInfo->isCompressedGZ — エントリが gz で圧縮されているかどうかを調べる

説明

bool **PharFileInfo->isCompressedGZ** (void)

これは、Phar アーカイブ内のファイルが Gzip で圧縮されているかどうかを返します。

返り値

Phar アーカイブ内のファイルが Gzip で圧縮されている場合に **TRUE**、 そうでない場合に **FALSE** を返します。

例

Example#1 PharFileInfo->isCompressedGZ() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $p['myfile2.txt'] = 'hi';
    $p['myfile3.txt'] = 'hi';
    $p['myfile2.txt']->setCompressedGZ();
    $p['myfile3.txt']->setCompressedBZIP2();
    $file = $p['myfile.txt'];
    $file2 = $p['myfile2.txt'];
    $file3 = $p['myfile3.txt'];
    var_dump($file->isCompressedGZ());
    var_dump($file2->isCompressedGZ());
    var_dump($file3->isCompressedGZ());
} catch (Exception $e) {
    echo 'my.phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(true)
bool(false)
```

参考

- PharFileInfo->getCompressedSize()
- PharFileInfo->isCompressedBZIP2()
- PharFileInfo->isCompressed()

PharFileInfo->isCRCChecked

(PECL phar:1.0.0-1.2.1)

PharFileInfo->isCRCChecked — ファイルエントリの CRC が検証されているかどうかを調べる

説明

bool **PharFileInfo->isCRCChecked** (void)

これは、Phar アーカイブ内のファイルが CRC 検証されているかどうかを返します。

返り値

そのファイルの CRC が検証されている場合に **TRUE**、 そうでない場合に **FALSE** を返します。

例

Example#1 PharFileInfo->isCRCChecked() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    var_dump($file->isCRCChecked());
} catch (Exception $e) {
    echo 'my.phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(true)
```

PharFileInfo->setCompressedBZIP2

(PECL phar:1.0.0-1.2.1)

PharFileInfo->setCompressedBZIP2 — phar 内の現在の Phar エントリを、Bzip2 で圧縮する

説明

bool **PharFileInfo->setCompressedBZIP2** (void)

このメソッドは、Phar アーカイブ内のファイルを bzip2 を使用して圧縮します。この機能を使用するには、 bzip2 拡張モジュールが有効になっていなければなりません。 また、すでに gzip で圧縮されているファイルを処理するためには、 まず gzip を伸張するために zlib 拡張モジュールが有効になっていなければなりません。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

エラー / 例外

INI 設定 phar.readonly が on の場合、あるいは bzip2 拡張モジュールが使用できない場合に BadMethodCallException をスローします。

例

Example#1 PharFileInfo->setCompressedBZIP2() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    var_dump($file->isCompressedBZIP2());
    $p['myfile.txt']->setCompressedBZIP2();
    var_dump($file->isCompressedBZIP2());
} catch (Exception $e) {
    echo 'my.phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(true)
```

参考

- PharFileInfo->getCompressedSize()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- Phar->compressAllFilesBZIP2()

PharFileInfo->setCompressedGZ

(PECL phar:1.0.0-1.2.1)

PharFileInfo->setCompressedGZ — phar 内の現在の Phar エントリを、gz で圧縮する

説明

bool **PharFileInfo->setCompressedGZ** (void)

このメソッドは、Phar アーカイブ内のファイルを gzip を使用して圧縮します。この機能を使用するには、 zlib 拡張モジュールが有効になっていなければなりません。 また、すでに bzip2 で圧縮されているファイルを処理するためには、 まず bzip2 を伸張するために bzip2 拡張モジュールが有効になっていなければなりません。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

エラー / 例外

INI 設定 phar.readonly が on の場合、あるいは zlib 拡張モジュールが使用できない場合に BadMethodCallException をスローします。

例

Example#1 PharFileInfo->setCompressedGZ() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    var_dump($file->isCompressedGZ());
    $p['myfile.txt']->setCompressedGZ();
    var_dump($file->isCompressedGZ());
} catch (Exception $e) {
    echo 'my.phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(true)
```

参考

- PharFileInfo->getCompressedSize()
- PharFileInfo->isCompressedBZIP2()
- PharFileInfo->isCompressed()
- PharFileInfo->isCompressedBZIP2()
- PharFileInfo->isCompressed()
- Phar->compressAllFilesGZ()

PharFileInfo->setMetaData

(PECL phar:1.0.0-1.2.1)

PharFileInfo->setMetaData — ファイルとともに保存する、ファイル固有のメタデータを設定する

説明

void **PharFileInfo->setMetaData** (mixed $metadata)

setMetaData() を使用するのは、 既存のファイル情報としては表せない独自の情報を保存する場合のみに限るべきです。 メタデータを使用すると、phar アーカイブの読み込み時のパフォーマンスが劇的に低下します。これは、 メタデータのサイズが大きい場合やメタデータを含むファイルが大量にある場合に顕著になります。 ファイルのパーミッションについては phar 自体でネイティブにサポートしており、 PharFileInfo->chmod() で設定できることを覚えておきましょう。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

メタデータの使用例として考えられるのは、ユーザ/グループ を指定し、phar からディスクに展開する際にそれを設定することなどです。 ほかには、MIME 型として返す値を明示的に指定することなどがあります。 その他、ファイルについて説明するデータであるが ファイル自体に含むべきでないものを保存したりします。

パラメータ

metadata

ファイルとともに保存する情報を含む、PHP の変数。

例

Example#1 PharFileInfo->setMetaData() の例

```
<?php
// 確実に消しておきます
@unlink('brandnewphar.phar');
try {
    $p = new Phar(dirname(__FILE__) . '/brandnewphar.phar', 0, 'brandnewphar.phar');
    $p['file.txt'] = 'hello';
    $p['file.txt']->setMetaData(array('user' => 'bill', 'mime-type' => 'text/plain'));
    var_dump($p['file.txt']->getMetaData());
} catch (Exception $e) {
    echo 'phar を作成/変更できません:', $e;
}
?>
```

上の例の出力は以下となります。

```
array(2) {
   ["user"]=>
   string(4) "bill"
   ["mime-type"]=>
   string(10) "text/plain"
}
```

参考

- PharFileInfo->getMetaData()

PharFileInfo->setUncompressed

(PECL phar:1.0.0-1.2.1)

PharFileInfo->setUncompressed — phar 内の現在の Phar エントリが圧縮されている場合に、それを展開する

説明

bool **PharFileInfo->setUncompressed** (void)

このメソッドは、Phar アーカイブ内のファイルを展開します。 この機能を使用するには、ファイルの圧縮形式に応じて bzip2 あるいは zlib のいずれかの拡張モジュールが必要になります。 この関数は phar の内容を変更するので、使用するには INI 設定 phar.readonly が off になっていなければなりません。

エラー / 例外

INI 設定 phar.readonly が on の場合、あるいは bzip2/zlib 拡張モジュールが使用できない場合に BadMethodCallException をスローします。

例

Example#1 PharFileInfo->setUncompressed() の例

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    $file->setCompressedGZ();
    var_dump($file->isCompressed());
    $p['myfile.txt']->setUncompressed();
    var_dump($file->isCompressed());
} catch (Exception $e) {
    echo 'my.phar の作成/変更に失敗しました: ', $e;
}
?>
```

上の例の出力は以下となります。

```
bool(true)
bool(false)
```

参考

- PharFileInfo->getCompressedSize()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressedBZIP2()
- PharFileInfo->isCompressedGZ()
- PharFileInfo->isCompressed()
- Phar->uncompressAllFiles()

目次

- Phar->getVersion — Phar アーカイブのバージョン情報を返す
- Phar->isBuffering — Phar の書き込み操作がバッファリングされるか、あるいは直接ディスクに書き込まれるかを調べる
- Phar::loadPhar — 任意の phar アーカイブを、エイリアスを指定して読み込む
- Phar::mapPhar — 現在実行されている（phar 形式の）ファイルを読み込み、その内容を登録する
- Phar::offsetExists — ファイルが phar 内に存在するかどうかを調べる
- Phar::offsetGet — 指定したファイルの PharFileInfo オブジェクトを取得する
- Phar::offsetSet — 内部ファイルに、外部ファイルの内容を設定する
- Phar::offsetUnset — ファイルを phar から削除する
- Phar->setMetaData — phar アーカイブのメタデータを設定する
- Phar->setStub — Phar アーカイブの PHP ローダ（あるいは起動スタブ）を設定する
- Phar->startBuffering — Phar の書き込み操作のバッファリングを開始するが、ディスク上の Phar オブジェクトは変更しない
- Phar->stopBuffering — Phar アーカイブへの書き込みリクエストのバッファリングを終了し、変更内容をディスクに書き込む
- Phar->uncompressAllFiles — 現在の Phar アーカイブ内のすべてのファイルを展開する
- PharFileInfo->chmod — ファイル固有のパーミッションビットを設定する
- PharFileInfo::__construct — Phar エントリオブジェクトを作成する
- PharFileInfo->getCompressedSize — Phar アーカイブ内での実際のファイルの大きさ（圧縮された状態）を返す
- PharFileInfo->getCRC32 — CRC32 コードを返すか、CRC がチェックできない場合に例外をスローする
- PharFileInfo->getMetaData — ファイルとともに保存されている、ファイル固有のメタデータを返す
- PharFileInfo->getPharFlags — Phar ファイルエントリのフラグを返す
- PharFileInfo->isCompressed — エントリが圧縮されているかどうかを調べる
- PharFileInfo->isCompressedBZIP2 — エントリが bzip2 で圧縮されているかどうかを調べる
- PharFileInfo->isCompressedGZ — エントリが gz で圧縮されているかどうかを調べる
- PharFileInfo->isCRCChecked — ファイルエントリの CRC が検証されているかどうかを調べる
- PharFileInfo->setCompressedBZIP2 — phar 内の現在の Phar エントリを、Bzip2 で圧縮する
- PharFileInfo->setCompressedGZ — phar 内の現在の Phar エントリを、gz で圧縮する
- PharFileInfo->setMetaData — ファイルとともに保存する、ファイル固有のメタデータを設定する
- PharFileInfo->setUncompressed — phar 内の現在の Phar エントリが圧縮されている場合に、それを展開する

PHP オプションと情報(info)

導入

以下の関数によりPHP自体に関する多くの情報(例えば、実行時の設定、 ロードされている拡張モジュール、バージョン等)を 得ることができます。実行しているPHPのオプションを設定する関数もあ ります。おそらく最も有名な関数であるphpinfo() もここにあります。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

PHP オプション/情報設定オプション

名前	デフォルト	変更の可否	変更の履歴
assert.active	"1"	PHP_INI_ALL	
assert.bail	"0"	PHP_INI_ALL	
assert.warning	"1"	PHP_INI_ALL	
assert.callback	NULL	PHP_INI_ALL	
assert.quiet_eval	"0"	PHP_INI_ALL	
enable_dl	"1"	PHP_INI_SYSTEM	PHP 6.0.0 で削除されました。
max_execution_time	"30"	PHP_INI_ALL	
max_input_time	"-1"	PHP_INI_PERDIR	PHP 4.3.0 以降で有効
max_input_nesting_level	"64"	PHP_INI_PERDIR	4.4.8 以降で有効。PHP 5.0.0 で削除されました。
magic_quotes_gpc	"1"	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL。PHP 6.0.0 で削除されました。
magic_quotes_runtime	"0"	PHP_INI_ALL	PHP 6.0.0 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

assert.active boolean

assert() の評価を有効にします。

assert.bail boolean

assertion が失敗した時にスクリプトの実行を終了します。

assert.warning boolean

assertion が失敗する度に PHP 警告を発行します。

assert.callback string

assertion が失敗した時にコールされるユーザ関数

assert.quiet_eval boolean

assertion 式の評価時に error_reporting() の現在の 設定を使用します。有効な場合、評価時にエラーは表示されません。(暗黙のうちに error_reporting(0) とします) 無効な場合、エラーは、error_reporting() の設定に 基づき設定されます。

enable_dl boolean

このディレクティブは、Apacheモジュール版のPHPを使用した場合にのみ 有用です。PHPの動的ロード拡張機能をdl()で 仮想サーバー毎またはディレクトリ毎にオンまたはオフに変更することが できます。

動的ロード機能をオフにするのは主としてセキュリティ上の理由によります。 動的ロード機能により、 open_basedir の拘束を全て 無視することが可能になります。 デフォルトでは、セーフモードを 使用している場合以外、動的ロードが可能です。セーフモードにおいては、 dl() を使用することが常に不可能になります。

max_execution_time integer

スクリプトがパーサにより強制終了されるまでに許容される最大の 時間を秒単位で指定します。この命令は、いい加減に書かれた スクリプトがサーバーの負荷を上げることを防止するのに役立ちます。 デフォルトでは、30 に設定されています。

最大実行時間は、システムコール、ストリーム操作等の 影響を受けません。より詳細な情報については、 set_time_limit() 関数の説明を参照ください。

セーフモードで実行している場合にはこの設定を ini_set() で変更することはできません。次善策としては セーフモード をオフにするか あるいは php.ini 上で制限時間を変えるしかありません。

Web サーバによっては、別の設定項目があるかもしれません。たとえば、 Apache には Timeout ディレクティブ、IIS には CGI タイムアウト関数があり、どちらもデフォルトで 300 秒に設定されています。 これらの意味については、Web サーバのドキュメントを参照ください。

max_input_time integer

スクリプトが POST、GET そしてファイルアップロードなどの入力を パースする最大の時間を、秒単位で指定します。

max_input_nesting_level integer

外部からの入力変数 ($_GET や $_POST など) のネストの深さの最大値を設定します。

magic_quotes_gpc boolean
警告

この機能は 非推奨 であり、PHP 6.0.0 で 削除 されます。この機能を使用しないことを強く推奨します。

GPC(Get/Post/Cookie) 処理に関する magic_quotes の設定を行います。 magic_quotes が on の場合、'(シングルクオート)、" (ダブルクオート)、\(バックスラッシュ) 、NULL には全て自動的に バックスラッシュでエスケープ処理が行われます。magic_quotes_sybase も on の場合、シングルクオートは、バックスラッシュではなく シングルクオートでエスケープされます。

> **注意**: PHP 4 では、$_ENV 変数もエスケープの対象となります。
>
> **注意**: magic_quotes_sybase ディレクティブもONの場合、このオプションは、 magic_quotes_gpc を完全に上書きします。両方のオプションを有効に することにより、シングルクオートのみが '' のようにエスケープされます。 2重引用符、バックスラッシュ、NULは変更されず、エスケープされません。

get_magic_quotes_gpc() も参照してください。

magic_quotes_runtime boolean
警告

この機能は 非推奨 であり、PHP 6.0.0 で 削除 されます。この機能を使用しないことを強く推奨します。

magic_quotes_runtime が有効の場合、 データベースおよびテクストファイルを含む外部ソースから データを返す全ての関数のクオートは、バックスラッシュで エスケープされます。magic_quotes_sybase も on の場合、シングルクオートは、バックスラッシュの代わりに シングルクオートでエスケープされます。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数は、PHP コアに含まれており、常に利用可能です。

定義済の phpcredits() 定数

定数	値	説明
CREDITS_GROUP	1	コア開発者のリスト
CREDITS_GENERAL	2	一般的なクレジット: 言語設計とコンセプト、PHP と PHP SAPIモジュールの作者。
CREDITS_SAPI	4	PHPのサーバAPIモジュールとその作者の一覧。
CREDITS_MODULES	8	PHPの拡張モジュールとその作者の一覧。
CREDITS_DOCS	16	ドキュメント作成チームのクレジット
CREDITS_FULLPAGE	32	通常、他のフラグと組み合わせて使用されます。 他のフラグで示される情報を含む完全に独立したHTMLページを出力する ことを指定します。
CREDITS_QA	64	品質管理チームのクレジット
CREDITS_ALL	-1	全てのクレジット、CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGEを指定した場合と同じ。 この定数は、適当なタグを有する完全にスタンドアローンのHTMLページ を生成します。

phpinfo() の定数

定数	値	説明
INFO_GENERAL	1	configureオプション、*php.ini* の場所、構築日、Webサーバ、システ ム等。
INFO_CREDITS	2	PHP クレジット。phpcredits()も参照して下さい。
INFO_CONFIGURATION	4	PHPディレクティブの現在のローカルおよびマスター値。 ini_get()も参照してください。
INFO_MODULES	8	ロードされているモジュールとそれぞれの設定。
INFO_ENVIRONMENT	16	環境変数に関する情報で、*$_ENV*でも入手可能です。
INFO_VARIABLES	32	EGPCS(Environment, GET, POST, Cookie, Server)から定義済の変数を表 示します。
INFO_LICENSE	64	PHPライセンス情報。» license faqも参照してください。
INFO_ALL	-1	上記を全て表示します。これがデフォルト値です。

ASSERT_ACTIVE *(integer)*
ASSERT_CALLBACK *(integer)*
ASSERT_BAIL *(integer)*
ASSERT_WARNING *(integer)*
ASSERT_QUIET_EVAL *(integer)*

---

assert_options

(PHP 4, PHP 5)

assert_options — 様々な assert フラグを設定/取得する

説明

mixed ***assert_options*** *(int $what [, mixed $value])*

種々の assert() 制御オプションを設定したり、 単に現在の設定を調べたりします。

パラメータ

what

assert オプション

オプション	ini パラメータ	デフォルト	説明
ASSERT_ACTIVE	assert.active	1	assert() による評価を有効にする
ASSERT_WARNING	assert.warning	1	assersion に失敗した場合に PHP 警告を発生する
ASSERT_BAIL	assert.bail	0	assersion に失敗した場合に実行を終了する
ASSERT_QUIET_EVAL	assert.quiet_eval	0	assersion 式については error_reporting を無効にする
ASSERT_CALLBACK	assert.callback	(NULL)	assertion に失敗した場合にコールされるユーザ関数

value

オプションに指定する新しい値。

返り値

そのオプションの元の値、あるいはエラー時に ***FALSE*** *を返します。*

---

assert

(PHP 4, PHP 5)

assert — assertion が FALSE であるかどうかを調べる

説明

bool **assert** (mixed $assertion)

assert() は、指定した assertion を調べて、結果が FALSE の場合に適切な動作をします。

assertion が文字列として指定された場合、 **assert()**によりPHPコードとして評価されます。 文字列 assertion が優れているところは、assertion のチェックがオフになった場合のオーバーヘッドがより少な いことであり、assertionが失敗した場合のメッセージを式 assertion に有しています。 つまり、もし論理型の条件を assertion として渡した場合、この条件が assert_options() 関数で指定したハンドラ関数への引数とはならないということです。 条件はハンドラ関数をコールする前に文字列に変換され、論理型の FALSE は空文字列に変換されます。

assertion は、デバッグ目的にのみ使用するべきです。 assertion を常にTRUEとなる条件を調べる不具合診断に使用し、TRUE でない場合に何らかのプログラミングエラーを示したり、extension 関数または特定のシステム制限や機能といった、 特定の機能の存在をチェックするために使用することが可能です。

assersion は、入力パラメータのチェックのような通常の実行動作に 使用するべきではありません。一般的には、assertion のチェックを無効にしても そのコードが正常に動作しなければなりません。

assert() の動作は、 assert_options() またはマニュアルの関数の部分 に記述された .ini の設定により設定することが可能です。

関数 assert_options() や ASSERT_CALLBACK 設定ディレクティブにより失敗した assertion を処理するコールバック関数を設定することが可能です。

assert() のコールバックは、assertion が発生した場所に関する情報と共に assertion に渡されたコードを容易にキャプチャーできるため、 特に自動テストセットを構築する際に便利です。 この情報は他の手法でもキャプチャー可能ですが、assertion を使用することにより、より簡単かつ容易に行なうことが可能です!

コールバック関数は、3つの引数を受ける必要があります。最初の引数は、 assertionが失敗したファイルが含まれます。2番目の引数には、assertionが失敗した行が含まれ、3番目の引数には失敗した式が含まれます（もしある場合のみ。1 または "two" のようなリテラルの値は、 この引数に渡されません)。

パラメータ

assertion

assertion。

返り値

アサーションが false となった場合に FALSE、それ以外の場合に TRUE を返します。

例

Example#1 失敗した assertion をカスタムハンドラで処理する

```
<?php
// assertを有効にし、出力を抑制する
assert_options(ASSERT_ACTIVE, 1);
assert_options(ASSERT_WARNING, 0);
assert_options(ASSERT_QUIET_EVAL, 1);

// ハンドラ関数を作成する
function my_assert_handler($file, $line, $code)
{
    echo "<hr>Assertion Failed:
         File '$file'<br />
         Line '$line'<br />
         Code '$code'<br /><hr />";
}

// コールバックを設定する
assert_options(ASSERT_CALLBACK, 'my_assert_handler');

// 失敗するassertionを作成
assert('mysql_query("")');
?>
```

---

dl

(PHP 4, PHP 5)

dl — 実行時に PHP 拡張モジュールをロードする

説明

int **dl** (string $library)

library で指定された PHP 拡張モジュールを読み込みます。

その拡張モジュールが既に使用可能かどうかを調べまるには、 extension_loaded() を使用します。 これは、組み込みのモジュールと (php.ini か、あるいは **dl()** を使用して) 動的に読み込むモジュールの両方に対応しています。

パラメータ

library

このパラメータに指定できるのは拡張モジュールの ファイル名だけであり、それはプラットフォームに依存します。 例えば、Unix プラットフォームでは sockets 拡張モジュール (共有モジュールとしてコンパイルされていれば。デフォルトでは有りません!) は sockets.so と呼ばれていますし、一方 Windows プラットフォームでは php_sockets.dll と呼ばれます。

拡張モジュールを読み込むディレクトリは、プラットフォームによって異なります。

Windows - php.ini に明記されていない場合、デフォルトでは 拡張モジュールは c:¥php4¥extensions¥ から ロードされます。

Unix - php.ini に明記されていない場合、デフォルトでは 以下に依存します。

- PHP をビルドする際に --enable-debug を指定しているか否か
- PHP をビルドする際に (実験段階の) ZTS (Zend Thread Safety) サポートを有効にしているか否か
- 現在の ZEND_MODULE_API_NO(Zend 内部モジュール API 番号。基本的にはメジャーモジュール API の変更が発生した日時。 例:20010901)

上記を考慮して、ディレクトリのデフォルトは <install-dir>/lib/php/extensions/ <debug-or-not>-<zts-or-not>-ZEND_MODULE_API_NO となる。 例: /usr/local/php/lib/php/extensions/debug-non-zts-20010901 または /usr/local/php/lib/php/extensions/no-debug-zts-20010901.

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 拡張モジュールのロード機能が無効(注意書き参照)だったり、あるいは 無効化されている(enable_dl でオフにされているか または php.ini で セーフモード が有効になっている)場合は、 **E_ERROR** を発行して実行は停止されます。 指定されたライブラリをロードできず **dl()** が 失敗した場合、**FALSE** に加えて **E_WARNING** メッセージが 発行されます。

例

Example#1 dl() の例

```
<?php
// OS によってロードするファイルを切り替える
if (!extension_loaded('sqlite')) {
    if (strtoupper(substr(PHP_OS, 0, 3)) === 'WIN') {
        dl('php_sqlite.dll');
    } else {
        dl('sqlite.so');
    }
}

// PHP 4.3.0 では PHP_SHLIB_SUFFIX 定数を利用することも可能
if (!extension_loaded('sqlite')) {
    $prefix = (PHP_SHLIB_SUFFIX === 'dll') ? 'php_' : '';
    dl($prefix . 'sqlite.' . PHP_SHLIB_SUFFIX);
}
?>
```

注意

注意: **dl()** はマルチスレッド Web サーバ上では サポートされません。 そのような環境の場合には php.ini 上で extensions 命令を使用するようにしてください。しかし、CGI や CLI には 影響しません!

注意: PHP 5 では、CLI 以外の あらゆる SAPI において **dl()** 関数が非推奨となっています。 代わりに、拡張モジュール 読み込み用のディレクティブを利用してください。

注意: PHP 6 以降では、CLI, CGI および組み込み版 (embed) を除くすべての SAPI でこの関数は無効となっています。

注意: **dl()** は Unix プラットフォーム上では 大文字小文字を区別します。

注意: この関数は、safe-mode では無効となります。

参考

- 拡張モジュール読み込み用のディレクティブ
- extension_loaded()

---

extension_loaded

(PHP 4, PHP 5)

extension_loaded — ある拡張機能がロードされているかどうかを調べる

説明

bool **extension_loaded** (string $name)

拡張モジュールがロードされているかどうかを調べます。

パラメータ

name

拡張モジュールの名前。

phpinfo()を使って種々の拡張モジュールの 名前を見ることができます。PHP の CGIまたはCLIバージョン を使っている場合には -m スイッチで、使用できる全ての拡張モジュールのリストを得ることができます:

```
$ php -m
[PHP Modules]
xml
tokenizer
standard
sockets
session
posix
pcre
overload
mysql
```

```
mbstring
ctype

[Zend  Modules]
```

返り値

name で指定する拡張機能がロードされている場合に **TRUE** を返します。さもなければ **FALSE** を返します。

例

Example#1 extension_loaded() の例

```
<?php
if (!extension_loaded('gd')) {
    if (!dl('gd.so')) {
        exit;
    }
}
?>
```

注意

> **注意**: 特定の拡張が使用可能かどうかをテストするために、 **extension_loaded()**はその内部拡張の名前を 使用します。ほとんどの内部拡張の名前は小文字ですが、 大文字のものもあるかもしれません。この関数は 大文字小文字を区別することに 注意してください!

参考

- get_loaded_extensions()
- get_extension_funcs()
- phpinfo()
- dl()

---

get_cfg_var

(PHP 4, PHP 5)

get_cfg_var — PHP 設定オプションの値を取得する

説明

string **get_cfg_var** (string $option)

PHP の設定オプション option の値を取得します。

この関数は、 PHP がコンパイルされた際にセットされた設定情報や Apache の設定ファイルから (php3_configuration_option 命令により) 読んだ設定情報は返しません。

システムが 設定ファイル を使用しているかどうかを確認するには、cfg_file_path の設定値を取得してみてください。 この値が利用可能なら、設定ファイルが使用されています。

パラメータ

option

設定オプションの名前。

返り値

varname で指定された PHP 設定オプション の現在の値を返し、エラーの場合は **FALSE** を返します。

参考

- ini_get()

---

get_current_user

(PHP 4, PHP 5)

get_current_user — 現在の PHP スクリプトの所有者の名前を取得する

説明

string **get_current_user** (void)

現在の PHP スクリプトの所有者の名前を返します。

返り値

ユーザ名を表す文字列。

参考

- getmyuid()
- getmygid()
- getmypid()
- getmyinode()
- getlastmod()

---

get_defined_constants

(PHP 4 >= 4.0.7, PHP 5)

get_defined_constants — すべての定数の名前とその値を連想配列として返す

説明

array **get_defined_constants** ([mixed $categorize])

現在定義されている全ての定数の名前と値を返します。返される値には、 拡張モジュールにより作成された定数や define() 関数で作成された定数も含まれます。

パラメータ

categorize

これを渡すと、この関数は多次元の配列を返すようになります。 最初の次元のキーがカテゴリとなり、 そのカテゴリ内の定数とその値が下位レベルに格納されます。

```
<?php
define("MY_CONSTANT", 1);
print_r(get_defined_constants(true));
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [internal] => Array
         (
               [E_ERROR] => 1
               [E_WARNING] => 2
               [E_PARSE] => 4
               [E_NOTICE] => 8
               [E_CORE_ERROR] => 16
               [E_CORE_WARNING] => 32
               [E_COMPILE_ERROR] => 64
               [E_COMPILE_WARNING] => 128
               [E_USER_ERROR] => 256
               [E_USER_WARNING] => 512
               [E_USER_NOTICE] => 1024
              [E_ALL] => 2047
              [TRUE] => 1
         )

     [pcre] => Array
         (
               [PREG_PATTERN_ORDER] => 1
               [PREG_SET_ORDER] => 2
               [PREG_OFFSET_CAPTURE] => 256
               [PREG_SPLIT_NO_EMPTY] => 1
               [PREG_SPLIT_DELIM_CAPTURE] => 2
               [PREG_SPLIT_OFFSET_CAPTURE] => 4
              [PREG_GREP_INVERT] => 1
         )

     [user] => Array
         (
               [MY_CONSTANT] => 1
         )

)
```

注意: categorize パラメータの内容は何でもかまいません。 パラメータを渡したかどうかだけが判断の対象となります。

返り値

変更履歴

バージョン	説明
5.0.0	パラメータ categorize が追加されました。

例

Example#1 get_defined_constants() の例

```
<?php
print_r(get_defined_constants());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [E_ERROR] => 1
     [E_WARNING] => 2
     [E_PARSE] => 4
     [E_NOTICE] => 8
     [E_CORE_ERROR] => 16
     [E_CORE_WARNING] => 32
     [E_COMPILE_ERROR] => 64
     [E_COMPILE_WARNING] => 128
     [E_USER_ERROR] => 256
     [E_USER_WARNING] => 512
     [E_USER_NOTICE] => 1024
     [E_ALL] => 2047
     [TRUE] => 1
)
```

参考

- defined()
- get_loaded_extensions()
- get_defined_functions()
- get_defined_vars()

---

get_extension_funcs

(PHP 4, PHP 5)

get_extension_funcs — あるモジュールの関数名を配列として返す

説明

array **get_extension_funcs** (string $module_name)

この関数は、module_name で示したモジュールで定義された全ての関数の名前を返します。

パラメータ

module_name

モジュール名。

注意: このパラメータは 小文字 でなければなりません。

返り値

すべての関数を含む配列を返します。 module_name が拡張モジュールでない場合は **FALSE** を返します。

例

Example#1 XML 関数の出力

```
<?php
print_r(get_extension_funcs("xml"));
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => xml_parser_create
     [1] => xml_parser_create_ns
     [2] => xml_set_object
     [3] => xml_set_element_handler
     [4] => xml_set_character_data_handler
     [5] => xml_set_processing_instruction_handler
     [6] => xml_set_default_handler
     [7] => xml_set_unparsed_entity_decl_handler
     [8] => xml_set_notation_decl_handler
     [9] => xml_set_external_entity_ref_handler
     [10] => xml_set_start_namespace_decl_handler
     [11] => xml_set_end_namespace_decl_handler
     [12] => xml_parse
     [13] => xml_parse_into_struct
     [14] => xml_get_error_code
     [15] => xml_error_string
     [16] => xml_get_current_line_number
     [17] => xml_get_current_column_number
     [18] => xml_get_current_byte_index
     [19] => xml_parser_free
     [20] => xml_parser_set_option
     [21] => xml_parser_get_option
```

```
    [22] => utf8_encode
    [23] => utf8_decode
)
```

参考

- get_loaded_extensions()

---

get_include_path

(PHP 4 >= 4.3.0, PHP 5)

get_include_path — 現在の include_path 設定オプションを取得する

説明

string **get_include_path** (void)

現在の include_path 設定オプションを取得します。

返り値

パスを表す文字列を返します。

例

Example#1 get_include_path() の例

```
<?php
// PHP 4.3.0 以降で動作します
echo get_include_path();

// すべてのバージョンの PHP で動作します
echo ini_get('include_path');
?>
```

参考

- ini_get()
- restore_include_path()
- set_include_path()
- include()

---

get_included_files

(PHP 4, PHP 5)

get_included_files — include または require で読み込まれたファイルの名前を配列として返す

説明

array **get_included_files** (void)

この関数は、include(), include_once(), require(), require_once() によりスクリプトにロードされた すべてのファイルの名前を取得します。

返り値

すべてのファイル名を含む配列を返します。

最初にコールされたスクリプトは "include されたファイル" という扱いに なります。そのため、include() やその仲間たちにより 読み込まれたファイルの一覧に含めて表示されます。

複数回読み込まれているファイルも、 返される配列には一度しかあらわれません。

変更履歴

バージョン	説明
4.0.1	PHP 4.0.1および以前のバージョンにおいて、 **get_included_files()** は読み込まれるファイルが拡張子 *.php* となることを仮定しており、他の拡張子のファイルは返されませんでした。 **get_included_files()** により返される配列は 連想配列であり、include() および include_once() で読み込まれたファイルのみが 一覧に含まれていました。

例

Example#1 get_included_files() の例

```php
<?php
// このファイルは abc.php です

include 'test1.php';
include_once 'test2.php';
require 'test3.php';
require_once 'test4.php';

$included_files = get_included_files();

foreach ($included_files as $filename) {
    echo "$filename¥n";
}

?>
```

上の例の出力は以下となります。

```
abc.php
test1.php
test2.php
test3.php
test4.php
```

注意

注意: 設定ディレクティブ auto_prepend_file で読み込まれたファイルは、返される配列には含まれません。

参考

- include()
- include_once()
- require()
- require_once()
- get_required_files()

---

get_loaded_extensions

(PHP 4, PHP 5)

get_loaded_extensions — コンパイル/ロードされている全てのモジュールの名前を配列として返す

説明

array **get_loaded_extensions** ([bool $zend_extensions=FALSE])

この関数は、PHPインタプリタにコンパイル、 ロードされている全てのモジュールの名前を返します。

パラメータ

zend_extensions

Zend エクステンションを返すかどうか。 デフォルトは **FALSE** (Zend エクステンションは一覧に含めない) です。

返り値

モジュール名の配列を返します。

変更履歴

バージョン	説明
5.2.4	オプションの zend_extensions パラメータが追加されました。

例

Example#1 get_loaded_extensions() の例

```php
<?php
print_r(get_loaded_extensions());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [0] => xml
    [1] => wddx
    [2] => standard
    [3] => session
    [4] => posix
    [5] => pgsql
    [6] => pcre
    [7] => gd
    [8] => ftp
```

```
    [9] => db
    [10] => calendar
    [11] => bcmath
)
```

参考

- get_extension_funcs()
- extension_loaded()
- dl()
- phpinfo()

---

get_magic_quotes_gpc

(PHP 4, PHP 5)

get_magic_quotes_gpc — magic quotes gpc の現在の設定を得る

説明

int **get_magic_quotes_gpc** (void)

magic_quotes_gpc の現在の設定を返します。

magic_quotes_gpcは 実行時にセットしても反映されないことに 留意してください。

magic_quotes についての詳細な情報は セキュリティの欄 を参照してください。

返り値

magic quotes gpc がオフの場合に 0、そうでない場合に 1 を返します。

例

Example#1 get_magic_quotes_gpc() の例

```
<?php
echo get_magic_quotes_gpc();          // 1
echo $_POST['lastname'];              // O¥'reilly
echo addslashes($_POST['lastname']); // O¥¥'reilly

if (!get_magic_quotes_gpc()) {
    $lastname = addslashes($_POST['lastname']);
} else {
    $lastname = $_POST['lastname'];
}

echo $lastname; // O¥'reilly
$sql = "INSERT INTO lastnames (lastname) VALUES ('$lastname')";
?>
```

注意

> **注意**: magic_quotes_sybase ディレクティブがONの場合、magic_quotes_gpcは 完全に上書きされます。そのため **get_magic_quotes_gpc()** が**TRUE**を返したとしてもダブルクォーテーションやバックスラッシュ、 NULLはエスケープされません。シングルクォーテーションだけがエスケープ されます。そのケースでは''のように見えます。

参考

- addslashes()
- stripslashes()
- get_magic_quotes_runtime()
- ini_get()

---

get_magic_quotes_runtime

(PHP 4, PHP 5)

get_magic_quotes_runtime — magic_quotes_runtime の現在アクティブな設定値を取得する

説明

int **get_magic_quotes_runtime** (void)

magic_quotes_runtime の現在アクティブな値を返します。

返り値

magic quotes runtime がオフの場合に 0、そうでない場合に 1 を返します。

参考

- get_magic_quotes_gpc()
- set_magic_quotes_runtime()

---

get_required_files

(PHP 4, PHP 5)

get_required_files — get_included_files() のエイリアス

説明

この関数は次の関数のエイリアスです。 get_included_files().

---

getenv

(PHP 4, PHP 5)

getenv — 環境変数の値を取得する

説明

string **getenv** (string $varname)

環境変数の値を取得します。

phpinfo() を使用して全ての環境変数の一覧を見る ことができます。 » CGI specification 、特に » 環境変数のページを参照すること により、それらの参照する環境変数の役割の多くを知ることができます。

パラメータ

varname

変数の名前。

返り値

varname が示す環境変数の値を返し、 エラーの場合は**FALSE**を返します。

例

Example#1 getenv() の例

```
<?php
// getenv() の使用例
$ip = getenv('REMOTE_ADDR');

// または単純にスーパーグローバル ($_SERVER または $_ENV) を使用します
$ip = $_SERVER['REMOTE_ADDR'];
?>
```

参考

- putenv()
- apache_getenv()
- スーパーグローバル

---

getlastmod

(PHP 4, PHP 5)

getlastmod — 最終更新時刻を取得する

説明

int **getlastmod** (void)

現在のページの最終更新時刻を取得します。

別のファイルの最終更新時刻が知りたい場合は、 filemtime() を使用してください。

返り値

現在のページの最終更新時刻を返します。 この値は Unix のタイムスタンプで、そのまま **date ()**に渡す事ができます。エラーの場合は **FALSE** を返します。

例

Example#1 getlastmod() の例

```
<?php
// たとえば、'最終更新時刻：March 04 1998 20:43:59.' を出力します
echo "最終更新時刻: " . date ("F d Y H:i:s.", getlastmod());
?>
```

参考

- date()
- getmyuid()
- getmygid()
- get_current_user()
- getmyinode()
- getmypid()
- filemtime()

---

getmygid

(PHP 4 >= 4.0.7, PHP 5)

getmygid — PHP スクリプトの所有者の GID を得る

説明

int **getmygid** (void)

現在のスクリプトのグループ ID を取得します。

返り値

現在のスクリプトのグループ ID を返します。またはエラー時に **FALSE** を返します。

参考

- getmyuid()
- getmypid()
- get_current_user()
- getmyinode()
- getlastmod()

---

getmyinode

(PHP 4, PHP 5)

getmyinode — 現在のスクリプトの inode を取得する

説明

int **getmyinode** (void)

現在のスクリプトの inode を取得します。

返り値

現在のスクリプトの inode を表す文字列、あるいはエラーの場合は **FALSE** を返します。

注意

> **注意**: この関数は Windows 環境にはまだ実装されていません。

参考

- getmygid()
- getmyuid()
- getmypid()
- get_current_user()
- getlastmod()

---

getmypid

(PHP 4, PHP 5)

getmypid — PHP のプロセス ID を取得する

説明

int ***getmypid*** *(void)*

現在の PHP プロセスの ID を取得します。

返り値

現在の PHP のプロセス ID を返し、エラーの場合は ***FALSE*** *を返します。*

注意

警告

プロセス ID は一意ではなく、エントロピ源として優れたものではありません。 セキュリティが問題となる状況では、PID に頼らないようにしましょう。

参考

- *getmygid()*
- *getmyuid()*
- *get_current_user()*
- *getmyinode()*
- *getlastmod()*

---

getmyuid

(PHP 4, PHP 5)

getmyuid — PHP スクリプト所有者のユーザ ID を取得する

説明

int ***getmyuid*** *(void)*

現在のスクリプトのユーザ ID を取得します。

返り値

現在のスクリプトのユーザ ID を返し、エラーの場合は ***FALSE*** *を返します。*

参考

- *getmygid()*
- *getmypid()*
- *get_current_user()*
- *getmyinode()*
- *getlastmod()*

---

getopt

(PHP 4 >= 4.3.0, PHP 5)

getopt — コマンドライン引数のリストからオプションを取得する

説明

array ***getopt*** *(string $options [, array $longopts])*

スクリプトに渡されたオプションをパースします。

パラメータ

options
この文字列の各文字をオプション文字として使用し、 スクリプトにハイフンひとつ (-) で始まるオプションとして渡された内容とマッチさせます。 たとえば、オプション文字列 "x" は -x というオプションを認識します。
longopts
オプションの配列。 この配列の各要素をオプション文字列として使用し、 スクリプトにハイフンふたつ (--) で始まるオプションとして渡された内容とマッチさせます。 たとえば、longopts の要素 "opt" は --opt というオプションを認識します。

> ***注意***: *PHP5.3.0 より前のバージョンでは、 このパラメータは一部のシステムでしか使用できません。*

options パラメータに含まれる要素には次のようなものがあります。

- *単一の文字 (値を受け付けない)*
- *文字の後にコロンをひとつ続けたもの (値が必須であるパラメータ)*
- *文字の後にコロンをふたつ続けたもの (値がオプションであるパラメータ)*

オプションの値は、文字列の後の最初の引数となります。 値の前に空白があるかどうかは関係ありません。

注意: オプションの値で、" "（空白）を区切り文字として使用することはできません。

注意: options と longopts の書式はほぼ同じです。唯一の違いは、 longopts はオプションの配列（その各要素がオプションとなる）を受け取るけれども options は文字列（その各文字がオプションとなる）を受け取るということです。

返り値

この関数はオプション/引数のペアを連想配列で返します。 失敗した場合は **FALSE** を返します。

変更履歴

バージョン	説明
5.3.0	この関数はシステムに依存しなくなり、 Windows でも動作するようになりました。

例

Example#1 getopt() の例

```
<?php
$options = getopt("f:hp:");
var_dump($options);
?>
```

上のスクリプトを php script.php -fvalue -h として実行すると、次のようになります。

```
array(2) {
  ["f"]=>
  string(5) "value"
  ["h"]=>
  bool(false)
}
```

Example#2 getopt() の例 その2

```
<?php
$shortopts  = "";
$shortopts .= "f:";  // 値が必須
$shortopts .= "v::"; // 値がオプション
$shortopts .= "abc"; // これらのオプションは値を受け取りません

$longopts  = array(
    "required:",     // 値が必須
    "optional::",    // 値がオプション
    "option",        // 値なし
    "opt",           // 値なし
);
$options = getopt($shortopts, $longopts);
var_dump($options);
?>
```

上のスクリプトを php script.php -f "value for f" -v -a --required value --optional="optional value" --option として実行すると、次のようになります。

```
array(6) {
  ["f"]=>
  string(11) "value for f"
  ["v"]=>
  bool(false)
  ["a"]=>
  bool(false)
  ["required"]=>
  string(5) "value"
  ["optional"]=>
  string(14) "optional value"
  ["option"]=>
  bool(false)
}
```

Example#3 getopt() の例 その3

複数のオプションを一度に渡す例

```
<?php
$options = getopt("abc");
var_dump($options);
?>
```

上のスクリプトを php script.php -aaac として実行すると、次のようになります。

```
array(2) {
  ["a"]=>
  array(3) {
    [0]=>
    bool(false)
    [1]=>
    bool(false)
    [2]=>
    bool(false)
  }
  ["c"]=>
```

```
    bool(false)
}
```

注意

注意: register_argc_argv オプションを有効にしないと、この館数は動作しません。

---

getrusage

(PHP 4, PHP 5)

getrusage — カレントリソースの使用に関する情報を得る

説明

array **getrusage** ([int $who])

この関数は、getrusage(2) へのインターフェースです。 システムコールから返されたデータを含む連想配列を返します。

パラメータ

who

who が 1 の場合、getrusage は RUSAGE_CHILDREN を付けてコールされます。

返り値

システムコールから返されたデータを含む連想配列を返します。 すべてのエントリは、記述されたフィールド名を用いてアクセス可能です。

例

Example#1 getrusage() の例

```
<?php
$dat = getrusage();
echo $dat["ru_nswap"];          // スワップの数
echo $dat["ru_majflt"];         // ページフォルトの数
echo $dat["ru_utime.tv_sec"];  // 使用するユーザー時間 (秒)
echo $dat["ru_utime.tv_usec"]; // 使用するユーザー時間 (マイクロ秒)
?>
```

注意

注意: この関数は Windows 環境にはまだ実装されていません。

参考

- getrusage(2) についてのシステムの man ページ

---

ini_alter

(PHP 4, PHP 5)

ini_alter — ini_set() のエイリアス

説明

この関数は次の関数のエイリアスです。 ini_set().

---

ini_get_all

(PHP 4 >= 4.2.0, PHP 5)

ini_get_all — 全ての設定オプションを得る

説明

array **ini_get_all** ([string $extension])

すべての登録済み設定オプションを返します。

パラメータ

extension

オプションで指定する拡張モジュール名。指定した場合は、 その拡張モジュールに関するオプションのみを返します。

返り値

返される配列ではキーとしてディレクティブの名前が使用されています。 配列の要素には、 global_value(php.iniで設定されている)、local_value(おそらくini_set()または .htaccessでセットされている)、access(アクセスレベル) の配列が入ります。 アクセスレベルの意味についてはマニュアルの 設定を変更するには をご覧ください。

> **注意**: ディレクティブに複数のアクセスレベルを設定することができます。 それが、access が妥当なビットマスク値を持っている理由です。

例

Example#1 ini_get_all() の例

```
<?php
$inis = ini_get_all();

print_r($inis);

?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [allow_call_time_pass_reference] => Array
    (
        [global_value] => 1
        [local_value] => 1
        [access] => 6
    )
    [allow_url_fopen] => Array
    (
        [global_value] => 1
        [local_value] => 1
        [access] => 7
    )

    ...

)
```

参考

- ini_get()
- ini_restore()
- ini_set()
- get_loaded_extensions()
- phpinfo()

---

ini_get

(PHP 4, PHP 5)

ini_get — 設定オプションの値を得る

説明

string **ini_get** (string $varname)

成功時に、設定オプションの値を返します。

パラメータ

varname

設定オプションの名前。

返り値

成功した場合に設定オプションの値、失敗した場合あるいは null 値を指定した場合に空の文字列を返します。

例

Example#1 ini_get() の例

```
<?php
/*
php.ini で以下のように設定されているものとします

display_errors = On
register_globals = Off
post_max_size = 8M
*/

echo 'display_errors = ' . ini_get('display_errors') . "¥n";
echo 'register_globals = ' . ini_get('register_globals') . "¥n";
echo 'post_max_size = ' . ini_get('post_max_size') . "¥n";
echo 'post_max_size+1 = ' . (ini_get('post_max_size')+1) . "¥n";
```

```
echo 'post_max_size in bytes = ' . return_bytes(ini_get('post_max_size'));

function return_bytes($val) {
    $val = trim($val);
    $last = strtolower($val{strlen($val)-1});
     switch($last) {
        // 'G' は PHP 5.1.0 以降で使用可能です
        case 'g':
            $val *= 1024;
        case 'm':
            $val *= 1024;
        case 'k':
            $val *= 1024;
    }

    return $val;
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
display_errors = 1
register_globals = 0
post_max_size = 8M
post_max_size+1 = 9
post_max_size in bytes = 8388608
```

注意

注意: **boolean 値を探す場合** off という boolean の ini 値は空文字列または "0" として返されます。 一方で on の ini 値は "1" として返されます。 また、この関数はリテラル文字列で設定された INI 値を返すこともできます。

注意: **メモリサイズの値を探す場合** upload_max_filesize のようなメモリサイズの値の場合、 php.ini上で省略形で格納されています。 **ini_get()**はphp.iniに格納されている値を そのままの形式で返します。整数表現に変換したりはしません。 これらの値に通常の算術的な関数を施すと予期しない結果を 得てしまいます。上の例では、省略形の表記から本来のバイト数に変換する PHP ソースのひとつの例を示しています。

参考

- get_cfg_var()
- ini_get_all()
- ini_restore()
- ini_set()

---

ini_restore

(PHP 4, PHP 5)

ini_restore — 設定オプションの値を元に戻す

説明

void **ini_restore** (string $varname)

指定した設定オプションを元の値に戻します。

パラメータ

varname

設定オプションの名前。

返り値

値を返しません。

参考

- ini_get()
- ini_get_all()
- ini_set()

---

ini_set

(PHP 4, PHP 5)

ini_set — 設定オプションの値を設定する

説明

string **ini_set** (*string $varname , string $newvalue*)

指定した設定オプションの値を設定します。 設定オプションは、スクリプトの実行中は新しい値を保持し、 スクリプト終了時に元の値へ戻されます。

パラメータ

varname

全てのオプションが **ini_set()** を使用して変更することが 可能なわけではありません。有効なオプションの完全な一覧は 付録 を参照ください。

newvalue

オプションの新しい値。

返り値

成功した場合に元の値、失敗した場合に **FALSE** を返します。

参考

- get_cfg_var()
- ini_get()
- ini_get_all()
- ini_restore()
- 設定を変更するには

---

main

(No version information available, might be only in CVS)

main — **main()** のダミー

説明

PHP のソースを除き、**main()** という名前の関数は 存在しません。PHP 4.3.0 では、PHP ソースにおいて新しいタイプの エラー処理 (php_error_docref)が追加されました。そのひとつの機能として、 html_errors (デフォルトでオン)と docref_root (PHP 4.3.2 まではデフォルトでオン)のディレクティブがセットされている 場合に、PHP のエラーメッセージ上に PHP のマニュアルへのリンクが 表示されるようになりました。

ときおり、エラーメッセージは **main()** 関数の マニュアルを指すことがあります。それがこのページが存在する理由です。 どの PHP 関数に関連するエラーで **main()** を 指すのか、ユーザーコメント欄に追加してくださいますようお願いします。 そうすれば、適切なドキュメントを指すように修正されるでしょう。

現在判明している、main() を指すエラー

関数名	このバージョン以降、ここを指すことはありません
include()	5.1.0
include_once()	5.1.0
require()	5.1.0
require_once()	5.1.0

参考

- html_errors
- display_errors

---

memory_get_peak_usage

(PHP 5 >= 5.2.0)

memory_get_peak_usage — PHP によって割り当てられたメモリの最大値を返す

説明

int **memory_get_peak_usage** *([bool $real_usage])*

PHP スクリプトに割り当てられたメモリの最大値を、バイト単位で返します。

パラメータ

real_usage

これを **TRUE** に設定すると、システムが割り当てた実際のメモリの大きさを取得します。 省略したり **FALSE** を設定したりすると、 emalloc() が使用するメモリのみを報告します。

返り値

目盛りの最大値をバイト数で返します。

変更履歴

バージョン	説明
5.2.1	この関数を使用するために、 --enable-memory-limit でコンパイルすることは必須ではなくなりました。
5.2.0	real_usage が追加されました。

参考

- memory_get_usage()
- memory_limit

---

memory_get_usage

(PHP 4 >= 4.3.2, PHP 5)

memory_get_usage — PHP に割り当てられたメモリの量を返す

説明

int **memory_get_usage** ([bool $real_usage])

現在の PHP スクリプトに割り当てられたメモリの量をバイト単位で返します。

パラメータ

real_usage

これを **TRUE** に設定すると、システムが割り当てた実際のメモリの大きさを取得します。 省略したり **FALSE** を設定したりすると、 emalloc() が使用するメモリのみを報告します。

返り値

メモリの量をバイト単位で返します。

変更履歴

バージョン	説明
5.2.1	この関数を使用するために、 --enable-memory-limit でコンパイルすることは必須ではなくなりました。
5.2.0	real_usage が追加されました。

例

Example#1 memory_get_usage() の例

```
<?php
// これは単なる例にすぎません。
// 以下の数値はシステムによって変化します。

echo memory_get_usage() . "¥n"; // 36640

$a = str_repeat("Hello", 4242);

echo memory_get_usage() . "¥n"; // 57960

unset($a);

echo memory_get_usage() . "¥n"; // 36744

?>
```

参考

- memory_get_peak_usage()
- memory_limit

---

php_ini_scanned_files

(PHP 4 >= 4.3.0, PHP 5)

php_ini_scanned_files — 追加の ini ディレクトリにある .ini ファイルのリストを取得する

説明

string **php_ini_scanned_files** (void)

php_ini_scanned_files()は、php.iniをパースした 後で、設定ファイルのリストをカンマ区切りで返します。 これらのファイルは、PHPのコンパイル時に --with-config-file-scan-dir オプションを使って指定されたディレクトリから取得されます。

戻り値のファイル名には --with-config-file-scan-dir オプションで 指定されたパスが含まれます。

返り値

成功すると、.iniファイルをカンマ区切りにした文字列が返されます。 --with-config-files-scan-dir がセットされていなければ、 **FALSE**を返します。指定されたディレクトリが空であれば、 空文字列が返されます。ファイルが認識できないものであれば、 そのファイルは文字列には含まれますが同時にPHPがエラーを起こします。 このエラーはコンパイルの時と、**php_ini_scanned_files()** 関数を使用したときの両方で発生します。

例

Example#1 返される ini ファイルを一覧する例

```
<?php
if ($filelist = php_ini_scanned_files()) {
    if (strlen($filelist) > 0) {
        $files = explode(',', $filelist);

        foreach ($files as $file) {
            echo "<li>" . trim($file) . "</li>¥n";
        }
    }
}
?>
```

参考

- ini_set()
- phpinfo()

---

php_logo_guid

(PHP 4, PHP 5)

php_logo_guid — ロゴの guid を取得する

説明

string **php_logo_guid** (void)

ビルトインされている画像を使って PHP ロゴを表示する際に使用できる ID を返します。 ロゴが表示されるのは、expose_php が On の場合のみです。

返り値

PHPE9568F34-D428-11d2-A769-00AA001ACF42 を返します。

例

Example#1 php_logo_guid() の例

```
<?php

echo '<img src="' . $_SERVER['PHP_SELF'] .
     '?=' . php_logo_guid() . '" alt="PHP Logo !" />';

?>
```

参考

- phpinfo()
- phpversion()
- phpcredits()
- zend_logo_guid()

---

php_sapi_name

(PHP 4 >= 4.0.1, PHP 5)

php_sapi_name — ウェブサーバと PHP の間のインターフェイスの型を返す

説明

string **php_sapi_name** (void)

Web サーバと PHP (サーバ API、SAPI) の間のインターフェイスの型を小文字の文字列で返します。 CGI 版の PHP ではこの文字列は「cgi」となり、Apache の mod_php 版ではこの文字列は「apache」となるといったようになります。

返り値

インターフェイスの型を小文字の文字列で返します。

例

Example#1 php_sapi_name() の例

```
<?php
$sapi_type = php_sapi_name();
if (substr($sapi_type, 0, 3) == 'cgi') {
    echo "CGI 版の PHP を使用しています\n";
} else {
    echo "CGI 版の PHP を使用していません\n";
}
?>
```

参考

- PHP_SAPI

---

php_uname

(PHP 4 >= 4.0.2, PHP 5)

php_uname — PHP が稼動しているオペレーティングシステムに関する情報を返す

説明

string **php_uname** ([string $mode])

php_uname() は、PHP が稼動しているオペレーティング システムに関する説明を返します。単に OS の名前を取得したい場合には **PHP_OS** 定数の利用を考えてください。ただし、 この定数が返すのは PHP が構築された OS の 情報であることに注意しましょう。

Unix では、もし現在稼動中の OS が判定できない場合には PHP が構築された OS を表示します。

パラメータ

mode

mode は、どのような情報を返すのかを一文字で指定します:

- 'a': デフォルトです。すべてのモードを "s n r v m" の順で返します。
- 's': オペレーティングシステム名。 例: FreeBSD
- 'n': ホスト名。 例: localhost.example.com
- 'r': リリース名。 例: 5.1.2-RELEASE
- 'v': バージョン情報。 オペレーティングシステムによって大きく変わります。
- 'm': マシン型。例: i386

返り値

説明を文字列で返します。

例

Example#1 php_uname() の例

```
<?php
echo php_uname();
echo PHP_OS;

/* 出力の例
Linux localhost 2.4.21-0.13mdk #1 Fri Mar 14 15:08:06 EST 2003 i686
Linux

FreeBSD localhost 3.2-RELEASE #15: Mon Dec 17 08:46:02 GMT 2001
FreeBSD

Windows NT XN1 5.1 build 2600
WINNT
*/

if (strtoupper(substr(PHP_OS, 0, 3)) === 'WIN') {
    echo 'このサーバは Windows です!';
} else {
    echo 'このサーバは Windows ではありません!';
}

?>
```

関連する 定義済みの定数 を使うほうが簡単なこともあります。例えばこのようになります。

Example#2 OS 関連の定数の例

```
<?php
// *nix
echo DIRECTORY_SEPARATOR; // /
echo PHP_SHLIB_SUFFIX;    // so
echo PATH_SEPARATOR;      // :

// Win*
echo DIRECTORY_SEPARATOR; // ¥
echo PHP_SHLIB_SUFFIX;    // dll
echo PATH_SEPARATOR;      // ;
?>
```

参考

- phpversion()
- php_sapi_name()
- phpinfo()

---

phpcredits

(PHP 4, PHP 5)

phpcredits — PHP に関するクレジットを出力する

説明

bool **phpcredits** ([int $flag])

この関数は、PHP 開発者、モジュール等のリストを有するクレジットを出力します。 ページに情報を挿入するために、適切な HTML コードが生成されます。

パラメータ

flag

独自のクレジットページを出力したい場合に flag を利用するとよいでしょう。flag はオプションで、デフォルトは **CREDITS_ALL** です。

定義済みの phpcredits() フラグ

名前	説明
CREDITS_ALL	すべてのクレジットを含めます。 CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE と同等です。これは、適切なタグを含んだ それ単体で成立する HTML ページを生成します。
CREDITS_DOCS	ドキュメントチームのクレジット
CREDITS_FULLPAGE	通常、他のフラグと組み合わせて使用します。 他のフラグで指定した情報を含む、それ単体で完全に独立した HTML ページを出力することを指定します。
CREDITS_GENERAL	一般的なクレジット: 言語の設計およびコンセプト、PHP 4.0 作者、SAPIモジュール
CREDITS_GROUP	コア開発者のリスト
CREDITS_MODULES	PHPの拡張モジュール、およびその作者のリスト
CREDITS_SAPI	PHP のサーバ API モジュールとその作者のリスト

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 全般的なクレジットの出力

```
<?php
phpcredits(CREDITS_GENERAL);
?>
```

Example#2 コア開発者およびドキュメントグループの表示

```
<?php
phpcredits(CREDITS_GROUP  + CREDITS_DOCS  + CREDITS_FULLPAGE);
?>
```

Example#3 すべてのクレジットの表示

```
<html>
  <head>
    <title>独自のクレジットページ</title>
  </head>
  <body>
<?php
// あなたが書いた独自のコード
phpcredits(CREDITS_ALL  - CREDITS_FULLPAGE);
// さらに別のコード
?>
  </body>
</html>
```

参考

- phpversion()
- php_logo_guid()
- phpinfo()

---

phpinfo

(PHP 4, PHP 5)

phpinfo — いろいろな PHP 情報を出力する

説明

bool **phpinfo** ([int $what])

現在の PHP の状態に関する、多くの情報を出力します。出力される情報には、 PHP コンパイルオプションと拡張機能、PHP のバージョン、 サーバ情報と環境（モジュールとしてコンパイルされた場合）、 PHP の環境、OS バージョン情報、パス、構成オプションのマスター およびローカルの値、HTTP ヘッダ、PHP License などがあります。

システムの設定はそれぞれ違うため、実行時設定 や 利用できる 定義済みの変数 を調べるために **phpinfo()** がよく使われます。

また、**phpinfo()** には EGPCS (Environment, GET, POST, Cookie, Server) の情報が含まれているため、デバッグツールとしても便利です。

パラメータ

what

以下にあるconstantsビット値をひとつまたは 複数個を加算して、オプションのwhat 引数に 渡すことによって出力をカスタマイズできます。 それぞれの定数やビット値をor演算子 で結んで渡すこともできます。

phpinfo() のオプション

名前(定数)	値	説明
INFO_GENERAL	1	configure オプション、php.ini の場所、ビルド日時、 Web サーバ、オペレーティングシステム等。
INFO_CREDITS	2	PHP クレジット。phpcredits() も参照ください。
INFO_CONFIGURATION	4	ローカルおよびマスタの、現在の PHP ディレクティブの値。 ini_get() も参照ください。
INFO_MODULES	8	ロードされているモジュールと、それぞれの設定。 get_loaded_extensions() も参照ください。
INFO_ENVIRONMENT	16	$_ENV で取得できる環境変数の情報。
INFO_VARIABLES	32	EGPCS（環境変数・GET・POST・クッキー・サーバ変数）から すべての 定義済みの変数を表示します。
INFO_LICENSE	64	PHP ライセンス情報。» ライセンス FAQ も参照ください。
INFO_ALL	-1	上記のすべてを表示します。これがデフォルトです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 phpinfo() の例

```
<?php

// すべての情報を表示します。デフォルトは INFO_ALL です。
phpinfo();

// モジュール情報だけを表示します。
// phpinfo(8) としても同じです。
phpinfo(INFO_MODULES);

?>
```

注意

> **注意**: expose_php が off の場合、一部の情報は表示されません。 これにはPHPとZendのロゴ、そしてクレジットが含まれます。
>
> **注意**: CLI モードを利用している場合、**phpinfo()** は HTML ではなくプレーンテキストで結果を出力します。

参考

- phpversion()
- phpcredits()
- php_logo_guid()
- ini_get()
- ini_set()
- get_loaded_extensions()
- 定義済みの変数

---

phpversion

(PHP 4, PHP 5)

phpversion — 現在の PHP バージョンを取得する

説明

string **phpversion** ([string $extension])

現在動作中の PHP パーサあるいは拡張モジュールのバージョンを表す文字列を返します。

パラメータ

extension

オプションで指定する拡張モジュール名。

返り値

オプションの extension パラメータが指定されている場合、**phpversion()** はその拡張モジュールのバージョンを返します。 関連するバージョン情報が存在しない場合、 あるいは拡張モジュールが有効でない場合は **FALSE** を返します。

例

Example#1 phpversion() の例

```
<?php
// たとえば 'Current PHP version: 4.1.1' などと表示します
echo 'Current PHP version: ' . phpversion();

// たとえば '2.0' などと表示します。拡張モジュールが有効でない場合は何も表示しません
echo phpversion('tidy');
?>
```

注意

> **注意**: この情報は、定義済みの定数 **PHP_VERSION** でも取得可能です。

参考

- version_compare()
- phpinfo()
- phpcredits()
- php_logo_guid()
- zend_version()

---

putenv

(PHP 4, PHP 5)

putenv — 環境変数の値を設定する

説明

bool **putenv** (string $setting)

サーバの環境変数に setting を追加します。 この環境変数は、カレントのリクエストを実行している間のみ存在します。 リクエスト終了時、環境変数は元の状態に戻されます。

ある種の環境変数が変更されることは潜在的なセキュリティリスクとなる 可能性があります。safe_mode_allowed_env_vars ディレクティブには接頭辞のカンマ区切りのリストが含まれます。セーフ モードでは、ユーザはこのディレクティブで指定された接頭辞で始まる名前 を有する環境変数のみを変更可能となります。 デフォルトでは、ユーザはPHP_ で始まる環境変数 (例えばPHP_FOO=BAR)のみを変更可能です。注意:この ディレクティブが空の場合、PHPはユーザに全ての環境変数を修正できる許可 を与えてしまいます!

safe_mode_protected_env_vars ディレクティブには、 カンマ区切りの環境変数のリストが含まれます。ユーザは、この環境変数 を**putenv()**により変更することができません。これら の変数は、safe_mode_allowed_env_varsが変更するこ とを許可している場合でも保護されます。

パラメータ

setting

"FOO=BAR" 形式の設定。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 環境変数の設定

```
<?php
putenv("UNIQID=$uniqid");
?>
```

注意

警告

これらのディレクティブは、 セーフモード が有効な場合にのみ効果があります!

参考

- getenv()

---

restore_include_path

(PHP 4 >= 4.3.0, PHP 5)

restore_include_path — include_path 設定オプションの値を元に戻す

説明

void **restore_include_path** (void)

include_path 設定値を php.ini でセットされたオリジナルの設定に戻します。

返り値

値を返しません。

例

Example#1 restore_include_path() の例

```
<?php

echo get_include_path();  //  .:/usr/local/lib/php

set_include_path('/inc');

echo get_include_path();  // /inc

// PHP 4.3.0 以降で動作します
restore_include_path();

// すべてのバージョンの PHP で動作します
ini_restore('include_path');

echo get_include_path();  //  .:/usr/local/lib/php

?>
```

参考

- ini_restore()
- get_include_path()
- set_include_path()
- include()

---

set_include_path

(PHP 4 >= 4.3.0, PHP 5)

set_include_path — include_path 設定オプションをセットする

説明

string **set_include_path** (string $new_include_path)

include_path 設定オプションの値を、このスクリプト内でだけ変更します。

パラメータ

new_include_path

include_path の新しい値。

返り値

成功した場合に元の include_path の値、失敗した場合に **FALSE** を返します。

例

Example#1 set_include_path() の例

```
<?php
// PHP 4.3.0 以降で動作します
set_include_path('/inc');

// すべてのバージョンの PHP で動作します
ini_set('include_path', '/inc');
?>
```

Example#2 include path の追加

PATH_SEPARATOR 定数を利用することで、 オペレーティングシステムに依存せずに include path を追加することが可能です。

この例では、既存の include_path の最後に /usr/lib/pear を追加します。

```
<?php
$path = '/usr/lib/pear';
set_include_path(get_include_path() . PATH_SEPARATOR . $path);
?>
```

参考

- ini_set()
- get_include_path()
- restore_include_path()
- include()

---

set_magic_quotes_runtime

(PHP 4, PHP 5)

set_magic_quotes_runtime — magic_quotes_runtime の現在アクティブな設定をセットする

説明

bool **set_magic_quotes_runtime** (int $new_setting)

magic_quotes_runtime の現在アクティブな設定をセットします。

パラメータ

new_setting

0 はオフ、1 はオンを表します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- get_magic_quotes_gpc()
- get_magic_quotes_runtime()

---

set_time_limit

(PHP 4, PHP 5)

set_time_limit — 実行時間の最大値を制限する

説明

void **set_time_limit** (int $seconds)

スクリプトが実行可能な秒数を設定します。 この制限にかかるとスクリプトは致命的エラーを返します。 デフォルトの制限値は 30 秒です。 なお、php.iniでmax_execution_timeの 値が定義されている場合にはそれを用います。

この関数がコールされた場合、 タイムアウトカウンタをゼロから再スタートします。 言いかえると、タイムアウトがデフォルトの 30 秒で スクリプト実行までに 25 秒かかる場合に、 set_time_limit(20) を実行すると、スクリプトは、 タイムアウトまでに全体で 45秒 の間実行されます。

パラメータ

seconds

最大実行時間を表す秒数。ゼロを設定すると、時間制限を行いません。

返り値

値を返しません。

注意

警告

この関数は、PHP が セーフモード で実行されている場合には効果がないことに注意してください。 セーフモードをオフにするか、php.ini の時間制限を変更する以外に対策はありません。

> **注意**: 関数 **set_time_limit()** と設定ディレクティブ max_execution_time は、 このスクリプト自体の実行時間にのみ影響を与えます。 system() を用いたシステムコール、ストリーム操作、 データベースクエリ等のスクリプト実行以外で発生する処理にかかった時間は スクリプトが実行される最大時間を定義する際には含まれません。

参考

- max_execution_time
- max_input_time

---

sys_get_temp_dir

(PHP 5 >= 5.2.1)

sys_get_temp_dir — 一時ファイル用に使用されるディレクトリのパスを返す

説明

string **sys_get_temp_dir** (void)

PHP が一時ファイルを保存するデフォルトのディレクトリのパスを返します。

返り値

一時ディレクトリのパスを返します。

参考

- tmpfile()
- tempnam()

---

version_compare

(PHP 4 >= 4.0.7, PHP 5)

version_compare — ふたつの "PHP 標準" バージョン番号文字列を比較する

説明

mixed **version_compare** (string $version1 , string $version2 [, string $operator])

version_compare()は、ふたつの "PHP 標準" バージョン 番号文字列を比較します。この関数は、いくつかのバージョンの PHP でのみ 動作するプログラムを書きたい場合に有用です。

この関数はまず、バージョン文字列の _, -, + をドット . で置き換えます。 さらに、数値でない部分の前後にドット . を追加します。 例えば '4.3.2RC1' は '4.3.2.RC.1' となります。 次に、explode('.', $ver) とするのと同じように結果を分割し、左から右へ 各部分を比較していきます。特殊な文字列が含まれている場合は以下の順で 並べ替えます: dev < alpha = a < beta = b < RC < pl. この方法により、'4.1' と '4.1.2' のようなバージョンの違いだけではなく PHP 固有の開発ステータスの違いも判断することが可能となります。

パラメータ

version1

最初のバージョン番号。

version2

ふたつめのバージョン番号。

operator

三番目のオプション引数 operator を指定した場合、 特定の関係を調べることが可能です。指定可能な演算子を以下に示します。 <, lt, <=, le, >, gt, >=, ge, ==, =, eq, !=, <>, ne この引数を用いると、この関数はこの演算子により指定された関係が 成り立つ場合に **TRUE** 、そうでない場合に **FALSE** を返します。

返り値

version_compare()は、最初のバージョンが二番目よりも小さい場合に -1、等しい場合に 0、二番目のほうが小さい場合に +1 を返します。

例

Example#1 version_compare() の例

```
<?php
// -1 を表示します
echo version_compare("4.0.4", "4.0.6");

// これらはすべて 1 を表示します
echo version_compare("4.0.4", "4.0.6", "<");
echo version_compare("4.0.6", "4.0.6", "eq");
?>
```

注意

> **注意**: **PHP_VERSION** 定数には現在の PHP のバージョンが格納されます。

---

zend_logo_guid

(PHP 4, PHP 5)

zend_logo_guid — Zend guid を取得する

説明

string **zend_logo_guid** (void)

この関数は、ビルトインされている画像を使って Zend ロゴを表示する際に使用できる ID を返します。

返り値

PHPE9568F35-D428-11d2-A769-00AA001ACF42 を返します。

例

Example#1 zend_logo_guid() の例

```
<?php

echo '<img src="' . $_SERVER['PHP_SELF'] .
     '?=' . zend_logo_guid() . '" alt="Zend Logo !" />';

?>
```

参考

- php_logo_guid()

---

zend_thread_id

(PHP 5)

zend_thread_id — 現在のスレッドの一意な ID を返す

説明

int **zend_thread_id** (void)

この関数は、現在のスレッドの一意な ID を返します。

返り値

スレッドの ID を表す整数値を返します。

注意

注意: この関数が使用できるのは、PHP を ZTS (Zend Thread Safety) サポートつきでビルドした場合のみです。

---

zend_version

(PHP 4, PHP 5)

zend_version — 現在の Zend Engine のバージョンを取得する

説明

string **zend_version** (void)

現在実行中の Zend Engine のバージョンを含む文字列を返します。

返り値

Zend Engine のバージョン番号を文字列で返します。

例

Example#1 zend_version() の例

```
<?php
echo "Zend engine version: " . zend_version();
?>
```

上の例の出力は、たとえば以下のようになります。

```
Zend engine version: 2.2.0
```

参考

- phpinfo()
- phpcredits()
- php_logo_guid()
- phpversion()

目次

- assert — assertion が FALSE であるかどうかを調べる
- dl — 実行時に PHP 拡張モジュールをロードする
- extension_loaded — ある拡張機能がロードされているかどうかを調べる
- get_cfg_var — PHP 設定オプションの値を取得する
- get_current_user — 現在の PHP スクリプトの所有者の名前を取得する
- get_defined_constants — すべての定数の名前とその値を連想配列として返す
- get_extension_funcs — あるモジュールの関数名を配列として返す
- get_include_path — 現在の include_path 設定オプションを取得する
- get_included_files — include または require で読み込まれたファイルの名前を配列として返す
- get_loaded_extensions — コンパイル/ロードされている全てのモジュールの名前を配列として返す
- get_magic_quotes_gpc — magic quotes gpc の現在の設定を得る
- get_magic_quotes_runtime — magic_quotes_runtime の現在アクティブな設定値を取得する
- get_required_files — get_included_files のエイリアス
- getenv — 環境変数の値を取得する
- getlastmod — 最終更新時刻を取得する
- getmygid — PHP スクリプトの所有者の GID を得る
- getmyinode — 現在のスクリプトの inode を取得する
- getmypid — PHP のプロセス ID を取得する
- getmyuid — PHP スクリプト所有者のユーザ ID を取得する
- getopt — コマンドライン引数のリストからオプションを取得する
- getrusage — カレントリソースの使用に関する情報を得る
- ini_alter — ini_set のエイリアス
- ini_get_all — 全ての設定オプションを得る
- ini_get — 設定オプションの値を得る
- ini_restore — 設定オプションの値を元に戻す
- ini_set — 設定オプションの値を設定する
- main — main のダミー
- memory_get_peak_usage — PHP によって割り当てられたメモリの最大値を返す
- memory_get_usage — PHP に割り当てられたメモリの量を返す
- php_ini_scanned_files — 追加の ini ディレクトリにある .ini ファイルのリストを取得する
- php_logo_guid — ロゴの guid を取得する
- php_sapi_name — ウェブサーバと PHP の間のインターフェイスの型を返す
- php_uname — PHP が稼動しているオペレーティングシステムに関する情報を返す
- phpcredits — PHP に関するクレジットを出力する
- phpinfo — いろいろな PHP 情報を出力する
- phpversion — 現在の PHP バージョンを取得する
- putenv — 環境変数の値を設定する
- restore_include_path — include_path 設定オプションの値を元に戻す
- set_include_path — include_path 設定オプションをセットする
- set_magic_quotes_runtime — magic_quotes_runtime の現在アクティブな設定をセットする
- set_time_limit — 実行時間の最大値を制限する
- sys_get_temp_dir — 一時ファイル用に使用されるディレクトリのパスを返す
- version_compare — ふたつの "PHP 標準" バージョン番号文字列を比較する
- zend_logo_guid — Zend guid を取得する
- zend_thread_id — 現在のスレッドの一意な ID を返す
- zend_version — 現在の Zend Engine のバージョンを取得する

POSIX 関数

導入

このモジュールは、IEEE 10003.1 (POSIX.1) 標準ドキュメントで 定義された関数へのインターフェースを有しています。 これらの関数は、他の手段からは利用できません。 POSIX.1 としては例えばかなり以前から PHP 3 の一部として open(), read(), write() および close() 関数が定義されていました。 いくつかのよりシステム依存の関数は、以前は利用できませんでしたが、 このモジュールではこれらの関数に対する簡単なアクセス手段を提供する ことにより、これらの問題を解決しようとしています。

警告

posix_getpwnam() のような POSIX 関数で重要な データを取得することができます。 safe mode が有効な場合に、 POSIX関数のどれもアクセスチェックを行うことはできません。 このため、このような環境で処理を行うには、 (configure において --disable-posix を指定して) POSIX 拡張モジュールを無効にしておくことを 強く推奨します。

注意: この拡張モジュールは Windows 環境では利用できません。

インストール手順

POSIX 関数は、デフォルトで有効となっています。POSIX 互換関数を --disable-posix により無効にする ことができます。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

POSIX_F_OK (integer)
ファイルが存在するかどうかを調べます。
POSIX_R_OK (integer)
ファイルが存在し、読み込みが許可されているかどうかを調べます。
POSIX_W_OK (integer)
ファイルが存在し、書き込みが許可されているかどうかを調べます。
POSIX_X_OK (integer)
ファイルが存在し、実行が許可されているかどうかを調べます。
POSIX_S_IFBLK (integer)
ブロックスペシャルファイル。
POSIX_S_IFCHR (integer)
キャラクタスペシャルファイル。
POSIX_S_IFIFO (integer)
FIFO（名前つきパイプ）スペシャルファイル。
POSIX_S_IFREG (integer)
通常のファイル。
POSIX_S_IFSOCK (integer)
ソケット。

> **注意**: これらの定数は PHP 5.1.0 以降で有効です。システムによっては これらのうちのいくつかが使用できない場合があることにも注意してください。

参考

プロセス制御関数に関する節も 役に立つでしょう。

---

posix_access

(PHP 5 >= 5.1.0)

posix_access — ファイルのアクセス権限を判断する

説明

bool **posix_access** (string $file [, int $mode])

posix_access() は、ファイルに対するユーザの アクセス権限を調べます。

パラメータ

file

調べるファイルの名前。

mode

POSIX_F_OK、**POSIX_R_OK**、 **POSIX_W_OK** および **POSIX_X_OK** のうちのひとつあるいは複数からなるマスク。 デフォルトは **POSIX_F_OK** です。

POSIX_R_OK、**POSIX_W_OK** および **POSIX_X_OK** は、ファイルが存在して読み込み/書き込み/ 実行の権限があるかどうかを調べます。 **POSIX_F_OK** は単にファイルが存在するかどうか だけを調べます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 posix_access() の例

この例は $file が読み書き可能であるかどうかを調べ、 そうでない場合にエラーメッセージを表示します。

```
<?php

$file = 'some_file';

if (posix_access($file, POSIX_R_OK | POSIX_W_OK)) {
    echo 'このファイルの読み込みと書き込みが可能です!';

} else {
    $error = posix_get_last_error();

    echo "エラー $error: " . posix_strerror($error);
}

?>
```

注意

> **注意**: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

参考

- posix_get_last_error()
- posix_strerror()

posix_ctermid

(PHP 4, PHP 5)

posix_ctermid — 制御する端末のパス名を得る

説明

string **posix_ctermid** (void)

そのプロセスで現在制御している端末のパス名を表す文字列を作成します。 エラーが発生した場合は errno を設定します。この値を調べるには posix_get_last_error() を使用します。

返り値

処理に成功した場合は、現在制御している端末のパス名を表す文字列を返します。 それ以外の場合は **FALSE** を返し、errno を設定します。 この値を調べるには posix_get_last_error() を使用します。

例

Example#1 posix_ctermid() の例

この例は、現在の TTY へのパスを表示します。

```
<?php
echo "I am running from ".posix_ctermid();
?>
```

参考

- posix_ttyname()
- posix_get_last_error()

---

posix_get_last_error

(PHP 4 >= 4.2.0, PHP 5)

posix_get_last_error — 直近で失敗した posix 関数が設定したエラー番号を取得する

説明

int **posix_get_last_error** (void)

直近で失敗した posix 関数が設定したエラー番号を取得します。 エラー番号に対応するエラーメッセージを取得するには posix_strerror() を使用します。

返り値

直近で失敗した posix 関数が設定した errno（エラー番号）を返します。 エラーが発生していない場合は 0 が返されます。

例

Example#1 posix_get_last_error() の例

この例では、存在しないプロセス ID のプロセスを殺そうとしています。 その結果として発生したエラーの番号を表示します。

```
<?php
posix_kill(999459,SIGKILL);
echo 'エラーが返されました。エラー番号:  '.posix_get_last_error(); //Your error was ___
?>
```

参考

- posix_strerror()

---

posix_getcwd

(PHP 4, PHP 5)

posix_getcwd — 現在のディレクトリのパス名

説明

string **posix_getcwd** (void)

スクリプトの現在の実行ディレクトリのパスを、絶対パスで取得します。 エラー時には errno を設定します。この値は posix_get_last_error() で取得することができます。

返り値

絶対パスを表す文字列を返します。 エラー時には **FALSE** を返し、errno を設定します。この値は posix_get_last_error() で取得することができます。

例

Example#1 posix_getcwd() の例

この例は、このスクリプトの現在の作業ディレクトリの絶対パスを返します。

```
<?php
echo '現在の作業ディレクトリは  '.posix_getcwd();
?>
```

注意

注意: 以下のような場合は、この関数は失敗します。

- 読み込みあるいは検索の権限が取得できない
- パスが存在しない

---

posix_getegid

(PHP 4, PHP 5)

posix_getegid — 現在のプロセスの有効なグループ ID を返す

説明

int **posix_getegid** (void)

現在のプロセスの有効なグループ ID を返します。

返り値

実効グループ ID を表す整数値を返します。

例

Example#1 posix_getegid() example

この例は、実効グループ ID を表示し、それを posix_setegid() で変更します。

```
<?php
echo '実グループ ID は '.posix_getgid(); //20
posix_setegid(40);
echo '実グループ ID は '.posix_getgid(); //20
echo '実効グループ ID は '.posix_getegid(); //40
?>
```

注意

posix_getegid() は posix_getgid() と異なります。 実効グループ ID は、コール元のプロセスから posix_setegid() で変更できるからです。

参考

- posix_getgrgid() は、これをグループ名に変換します
- posix_getgid() は、実グループ ID を取得します
- posix_setgid() は、実効グループ ID を変更します

---

posix_geteuid

(PHP 4, PHP 5)

posix_geteuid — 現在のプロセスの有効なユーザ ID を返す

説明

int **posix_geteuid** (void)

現在のプロセスの有効なユーザ ID を返します。 使用可能なユーザ名に変換する方法に関する情報については、posix_getpwuid() も参照ください。

返り値

ユーザ ID を表す整数値を返します。

例

Example#1 posix_geteuid() の例

この例は、まず現在のユーザ ID を表示し、 それから posix_seteuid() で実効ユーザ ID を設定します。その後、改めて実ユーザ ID と実効ユーザ ID を表示します。

```
<?php
echo posix_getuid()."¥n"; //10001
echo posix_geteuid()."¥n"; //10001
posix_seteuid(10000);
echo posix_getuid()."¥n"; //10001
echo posix_geteuid()."¥n"; //10000
?>
```

参考

- posix_getpwuid() は、ユーザについての詳細な情報を取得します
- posix_getuid() は、実ユーザ ID を取得します
- posix_setuid() は、実効ユーザ ID を変更します
- POSIX man ページ GETEUID(2)

---

posix_getgid

(PHP 4, PHP 5)

posix_getgid — 現在のプロセスの実際のグループ ID を返す

説明

int **posix_getgid** (void)

現在のプロセスの実際のグループ ID を返します。

返り値

実際のグループ ID を表す整数値を返します。

例

Example#1 posix_getgid() の例

この例では、実グループ ID を表示し、 実効グループ ID を変更した後で再度実グループ ID を表示します。 have changed your effective group id.

```
<?php
echo '実グループ ID は '.posix_getgid(); //20
posix_setegid(40);
echo '実グループ ID は '.posix_getgid(); //20
echo '実効グループ ID は '.posix_getegid(); //40
?>
```

参考

- posix_getgrgid() は、グループ ID からグループ名を取得します
- posix_getegid() は、実効グループ ID を取得します
- posix_setgid() は、実効グループ ID を変更します
- POSIX man ページ GETGID(2)

---

posix_getgrgid

(PHP 4, PHP 5)

posix_getgrgid — 指定したグループ ID を有するグループに関する情報を返す

説明

array **posix_getgrgid** (int $gid)

指定した ID のグループに関する情報を取得します。

パラメータ

gid

グループ ID。

返り値

以下の要素を持つ配列を返します。

グループ情報の配列

要素	説明
name	グループ名。これは、16 文字以下の短い文字列からなる 「ハンドル」であり、実際の完全な名前とは異なります。

要素	説明
passwd	グループのパスワードを暗号化したもの。 システムが「シャドー」パスワードを使用している場合は、 ここではアスタリスクが返されます。
gid	グループ ID。この関数をコールする際に指定した *gid* と同じものになるので、冗長なデータです。
members	このグループに属する全メンバーを表す文字列の配列。

変更履歴

バージョン	説明
4.2.0	このバージョンより前は、members の内容は単なる整数 (グループに所属するユーザの数)であり、 メンバの名前は数値添字で返されました。

例

Example#1 posix_getgrgid() の使用例

```
<?php

$groupid   = posix_getegid();
$groupinfo = posix_getgrgid($groupid);

print_r($groupinfo);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [name]    => toons
    [passwd]  => x
    [members] => Array
        (
            [0] => tom
            [1] => jerry
        )
    [gid]     => 42
)
```

参考

- posix_getegid()
- posix_getgrnam()
- filegroup()
- stat()
- safe_mode_gid
- POSIX man ページ GETGRNAM(3)

---

posix_getgrnam

(PHP 4, PHP 5)

posix_getgrnam — 指定した名前のグループに関する情報を返す

説明

```
array posix_getgrnam ( string $name )
```

指定した名前のグループに関する情報を取得します。

パラメータ

name

グループの名前。

返り値

以下の要素を持つ配列を返します。

グループ情報の配列

要素	説明
name	グループ名。これは、16 文字以下の短い文字列からなる 「ハンドル」であり、実際の完全な名前とは異なります。 グループ ID。この関数をコールする際に指定した *name* と同じものになるので、冗長なデータです。
passwd	グループのパスワードを暗号化したもの。 システムが「シャドー」パスワードを使用している場合は、 ここではアスタリスクが返されます。
gid	グループ ID を表す数値。
members	このグループに属する全メンバーを表す文字列の配列。

変更履歴

バージョン	説明
4.2.0	このバージョンより前は、members の内容は単なる整数 (グループに所属するユーザの数)であり、 メンバの名前は数値添字で返されました。

例

Example#1 posix_getgrnam() の使用例

```
<?php

$groupinfo = posix_getgrnam("toons");

print_r($groupinfo);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [name]    => toons
    [passwd]  => x
    [members] => Array
        (
            [0] => tom
            [1] => jerry
        )
    [gid]     => 42
)
```

参考

- posix_getegid()
- posix_getgrgid()
- filegroup()
- stat()
- safe_mode_gid
- POSIX man ページ GETGRNAM(3)

---

posix_getgroups

(PHP 4, PHP 5)

posix_getgroups — 現在のプロセスのグループセットを返す

説明

```
array posix_getgroups ( void )
```

現在のプロセスのグループセットを取得します。

返り値

現在のプロセスのグループセットについて、 グループ ID を表す整数値を含む配列を返します。

例

Example#1 posix_getgroups() の使用例

```
<?php

$groups = posix_getgroups();

print_r($groups);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [0] => 4
    [1] => 20
    [2] => 24
    [3] => 25
    [4] => 29
    [5] => 30
    [6] => 33
    [7] => 44
    [8] => 46
    [9] => 104
    [10] => 109
    [11] => 110
    [12] => 1000
)
```

参考

- posix_getgrgid()

---

posix_getlogin

(PHP 4, PHP 5)

posix_getlogin — ログイン名を返す

説明

string **posix_getlogin** (void)

現在のプロセスを所有するユーザのログイン名を返します。

返り値

ユーザのログイン名を文字列で返します。

例

Example#1 posix_getlogin() の使用例

```
<?php
echo posix_getlogin(); //apache
?>
```

参考

- posix_getpwnam()
- POSIX man ページ GETLOGIN(3)

---

posix_getpgid

(PHP 4, PHP 5)

posix_getpgid — ジョブ制御のプロセスグループ ID を得る

説明

int **posix_getpgid** (int $pid)

プロセス pid のプロセスグループ ID を返します。

パラメータ

pid

プロセス ID。

返り値

ID を表す整数値を返します。

例

Example#1 posix_getpgid() の使用例

```
<?php
$pid = posix_getppid();
echo posix_getpgid($pid); //35
?>
```

注意

注意: この関数は POSIX 関数ではなく、BSD および System V のシステムで共通な関数です。 使用するシステムがこの関数をサポートしていない場合は、 PHP のコンパイル時にこの関数は組み込まれません。 この関数が使用可能かどうかを調べるには、 function_exists() を使用します。

参考

- posix_getppid()
- man ページ SETPGID(2)

---

posix_getpgrp

(PHP 4, PHP 5)

posix_getpgrp — 現在のプロセスのグループ ID を返す

説明

int **posix_getpgrp** (void)

現在のプロセスのグループ ID を返します。

返り値

ID を表す整数値を返します。

参考

- プロセスグループの詳細な情報に関しては、POSIX.1 および POSIX システムのマニュアルページ getpgrp(2) を参照ください。

---

posix_getpid

(PHP 4, PHP 5)

posix_getpid — 現在のプロセス ID を返す

説明

int **posix_getpid** (void)

現在のプロセスのプロセス ID を返します。

返り値

ID を表す整数値を返します。

例

Example#1 posix_getpid() の使用例

```
<?php
echo posix_getpid(); //8805
?>
```

参考

- posix_kill() は、プロセスを殺します
- POSIX man ページ GETPID(2)

---

posix_getppid

(PHP 4, PHP 5)

posix_getppid — 親プロセスの ID を返す

説明

int **posix_getppid** (void)

現在のプロセスの親プロセスの ID を返します。

返り値

ID を表す整数値を返します。

例

Example#1 posix_getppid() の使用例

```
<?php
echo posix_getppid(); //8259
?>
```

---

posix_getpwnam

(PHP 4, PHP 5)

posix_getpwnam — 指定した名前のユーザに関する情報を返す

説明

array **posix_getpwnam** (string $username)

指定したユーザに関する情報を配列で返します。

パラメータ

username

英数字で表したユーザ名。

返り値

返される配列の要素は次のようになります。

ユーザ情報配列

要素	説明
name	要素 name はユーザ名を有しています。これは、通常、 実際の完全な名前ではなく16文字未満のユーザの"ハンドル名"となります。 この値はこの関数をコールした際に使用したパラメータ *username* と同じとする必要があり、 このため冗長な定義となります。
passwd	要素passwd には暗号化されたユーザのパスワードが含まれます。 シャドウパスワードを使用しているシステムでは、アスタリスクが代わりに 返されます。
uid	数値形式で表したユーザ ID。
gid	ユーザのグループ ID。 実際のグループ名を調べたりそのグループのメンバーの一覧を 得るには関数 posix_getgrgid() を使用してください。
gecos	GECOS は旧式の項であり、Honeywell バッチ処理プログラムの finger 情報フィールドを参照します。 しかし、このフィールドはまだ生きており、その内容はPOSIXで 規定されています。 このフィールドには、カンマで区切られた ユーザのフルネーム、オフィスの電話番号、家の電話番号に関する リストが含まれています。多くのシステムでは、ユーザのフルネーム のみが利用可能です。
dir	この要素には、ユーザのホームディレクトリへの絶対パスが含まれています。
shell	shell 要素には、ユーザのデフォルトシェルの実行ファイルへの絶対パスが 含まれています。

例

Example#1 posix_getpwnam() の使用例

```
<?php

$userinfo = posix_getpwnam("tom");

print_r($userinfo);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [name]   => tom
    [passwd] => x
    [uid]    => 10000
    [gid]    => 42
    [geocs]  => "tom,,,"
    [dir]    => "/home/tom"
    [shell]  => "/bin/bash"
)
```

参考

- posix_getpwuid()
- POSIX man ページ GETPWNAM(3)

---

posix_getpwuid

(PHP 4, PHP 5)

posix_getpwuid — 指定 ID のユーザに関する情報を返す

説明

array **posix_getpwuid** (int $uid)

指定したユーザ ID のユーザについての情報を配列で返します。

パラメータ

uid

ユーザ ID。

返り値

返される連想配列の要素は次のようになります。

ユーザ情報配列

要素	説明
name	要素 name はユーザ名を有しています。これは、通常、 実際の完全な名前ではなく16文字未満のユーザの"ハンドル名"となります。
passwd	要素passwd には暗号化されたユーザのパスワードが含まれます。 シャドウパスワードを使用しているシステムでは、アスタリスクが代わりに 返されます。
uid	ユーザID。これは、この関数をコールする際に使用するパラメータ uid と同じとなり、このため冗長になります。
gid	ユーザのグループID。 実際のグループ名を調べたりそのグループのメンバーの一覧を 得るには関数posix_getgrgid()を使用してください。
gecos	GECOS は旧式の項であり、Honeywell バッチ処理プログラムの finger 情報フィールドを参照します。 しかし、このフィールドはまだ生きており、その内容はPOSIXで 規定されています。 このフィールドには、カンマで区切られた ユーザのフルネーム、オフィスの電話番号、家の電話番号に関する リストが含まれています。多くのシステムでは、ユーザのフルネーム のみが利用可能です。
dir	この要素には、ユーザのホームディレクトリへの絶対パスが含まれています。
shell	shell 要素には、ユーザのデフォルトシェルの実行ファイルへの絶対パスが 含まれています。

例

Example#1 posix_getpwuid() の使用例

```
<?php

$userinfo = posix_getpwuid(10000);

print_r($userinfo);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [name]   => tom
    [passwd] => x
    [uid]    => 10000
    [gid]    => 42
    [geocs]  => "tom,,,"
    [dir]    => "/home/tom"
    [shell]  => "/bin/bash"
)
```

参考

- posix_getpwnam()
- POSIX man ページ GETPWNAM(3)

---

posix_getrlimit

(PHP 4, PHP 5)

posix_getrlimit — システムリソース制限に関する情報を返す

説明

array **posix_getrlimit** (void)

posix_getrlimit() は、 現在のリソースにおけるソフトリミットおよびハードリミットを配列で返します。

各リソースには、それぞれソフトリミットとハードリミットがあります。 ソフトリミットとは、そのリソースに対してカーネルが課す制限値のことです。 ハードリミットとは、ソフトリミットの上限値のことです。 特権を持たないプロセスは、ソフトリミットの値を 0 からハードリミット値の間で設定することが可能です。 また、ハードリミットの値を下げることができますが、 いったん下げた値を再び上げることはできません。

返り値

各リソースに関する制限値を含む連想配列を返します。 個々のリミット値には、ソフトリミットとハードリミットがあります。

返される制限の一覧

制限の名前	制限についての説明
core	コアファイルの最大サイズ。0 の場合はコアファイルを作成しません。 コアファイルのサイズがこの値を超えると、 このサイズまで切り詰められます。
totalmem	プロセスのメモリの最大サイズを表すバイト数。
virtualmem	プロセスの仮想メモリの最大サイズを表すバイト数。
data	プロセスのデータセグメントの最大サイズを表すバイト数。
stack	プロセスのスタックの最大サイズを表すバイト数。
rss	RAM 上の仮想ページの最大数。
maxproc	呼び出し元のプロセスの実ユーザ ID で作成できるプロセスの最大数。
memlock	RAM 内にロックできるメモリの最大バイト数。
cpu	そのプロセスが使用できる CPU 時間。
filesize	そのプロセスが使用できるデータセグメントの最大サイズを表すバイト数。
openfiles	オープンできるファイル記述子の最大値よりひとつ大きい値。

例

Example#1 posix_getrlimit() の使用例

```
<?php

$limits = posix_getrlimit();

print_r($limits);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [soft core] => 0
     [hard core] => unlimited
     [soft data] => unlimited
     [hard data] => unlimited
     [soft stack] => 8388608
     [hard stack] => unlimited
     [soft totalmem] => unlimited
     [hard totalmem] => unlimited
     [soft rss] => unlimited
     [hard rss] => unlimited
     [soft maxproc] => unlimited
     [hard maxproc] => unlimited
     [soft memlock] => unlimited
     [hard memlock] => unlimited
     [soft cpu] => unlimited
     [hard cpu] => unlimited
     [soft filesize] => unlimited
     [hard filesize] => unlimited
     [soft openfiles] => 1024
     [hard openfiles] => 1024
)
```

注意

> **注意**: この関数は POSIX 関数ではなく、BSD および System V のシステムで共通な関数です。 使用するシステムがこの関数をサポートしていない場合は、 PHP のコンパイル時にこの関数は組み込まれません。 この関数が使用可能かどうかを調べるには、 function_exists() を使用します。

参考

- man ページ GETRLIMIT(2)

---

posix_getsid

(PHP 4, PHP 5)

posix_getsid — プロセスの現在の sid を得る

説明

int **posix_getsid** (int $pid)

プロセス pid のセッション ID を返します。 プロセスのセッション ID とは、セッションリーダーのプロセスグループ ID のことです。

パラメータ

pid

プロセス ID。0 を指定すると、現在のプロセスとみなされます。

返り値

ID を表す整数値を返します。

例

Example#1 posix_getsid() の使用例

```
<?php
$pid = posix_getpid();
echo posix_getsid($pid); //8805
?>
```

参考

- posix_getpgid()
- posix_setsid()
- POSIX man ページ GETSID(2)

---

posix_getuid

(PHP 4, PHP 5)

posix_getuid — 現在のプロセスの実際のユーザ ID を返す

説明

int **posix_getuid** (void)

現在のプロセスの実際のユーザ ID を返します。

返り値

ID を表す整数値を返します。

例

Example#1 posix_getuid() の使用例

```
<?php
echo posix_getuid(); //10000
?>
```

参考

- 使用可能なユーザ名に変換する方法に関する情報については、 posix_getpwuid() も参照ください。
- POSIX man ページ GETUID(2)

---

posix_initgroups

(PHP 5 >= 5.2.0)

posix_initgroups — グループアクセスリストを求める

説明

bool **posix_initgroups** (string $name , int $base_group_id)

指定した名前のユーザについてのグループアクセスリストを求めます。

パラメータ

name

リストを取得したいユーザ。

base_group_id

パスワードファイルから取得したグループ番号。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- Unix の initgroups(3) のマニュアル

---

posix_isatty

(PHP 4, PHP 5)

posix_isatty — ファイル記述子が対話型端末であるかどうかを定義する

説明

bool **posix_isatty** (int $fd)

ファイル記述子 fd が、 有効な端末デバイスを指しているかどうかを調べます。

パラメータ

fd

ファイル記述子。

返り値

fd がオープンされており、 かつ端末に接続されている場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

参考

- posix_ttyname()

---

posix_kill

(PHP 4, PHP 5)

posix_kill — プロセスにシグナルを送信する

説明

bool **posix_kill** (int $pid , int $sig)

シグナル sig をプロセス ID pid のプロセスに送信します。

パラメータ

pid

プロセス ID。

sig

PCNTL シグナル定数 のいずれか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- POSIX システムの kill(2) マニュアルページも参照ください。 そこには、負のプロセスID、特別プロセスID 0、特別プロセスID -1、 シグナル番号 0 に関する追加情報があります。

---

posix_mkfifo

(PHP 4, PHP 5)

posix_mkfifo — fifo スペシャルファイル(名前付きパイプ)を作成する

説明

bool **posix_mkfifo** (string $pathname , int $mode)

posix_mkfifo() は、 FIFO スペシャルファイルを作成します。 これはファイルシステム内に存在し、プロセス間の双方向通信の末端として 動作します。

パラメータ

pathname

FIFO ファイルへのパス。

mode

2 番目のパラメータ mode は、8 進表記 (例: 0644)で指定する必要があります。新しく作成される FIFO のパーミッションは、現在の umask() の設定にも依存します。 作成されるファイルのパーミッションは (mode & ~umask) となります。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

注意

注意: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

---

posix_mknod

(PHP 5 >= 5.1.0)

posix_mknod — スペシャルファイルあるいは通常のファイルを作成する (POSIX.1)

説明

bool **posix_mknod** (string $pathname , int $mode [, int $major [, int $minor]])

スペシャルファイルあるいは通常のファイルを作成します。

パラメータ

pathname

作成するファイル。

mode

このパラメータは、ファイル型(以下の定数 **POSIX_S_IFREG**、 **POSIX_S_IFCHR**、**POSIX_S_IFBLK**、 **POSIX_S_IFIFO** あるいは **POSIX_S_IFSOCK** のうちのひとつ) およびパーミッションの論理和で構成されます。

major

メジャーデバイスカーネル ID (**S_IFCHR** あるいは **S_IFBLK** を使用している場合に渡す必要があります)。

minor

マイナーデバイスカーネル ID (デフォルトは 0 です)。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 posix_mknod() の例

```
<?php

$file = '/tmp/tmpfile';  //  ファイル名
$type = POSIX_S_IFBLK;   //  ファイル型
$permissions  = 0777;     // 8 進数
$major  = 1;
$minor  = 8;              //  /dev/random

if (!posix_mknod($file, $type | $permissions, $major, $minor)) {
    die('Error ' . posix_get_last_error() . ': ' . posix_strerror(posix_get_last_error()));
}

?>
```

参考

- posix_mkfifo()

---

posix_setegid

(PHP 4 >= 4.0.2, PHP 5)

posix_setegid — 現在のプロセスの実効 GID を設定する

説明

bool **posix_setegid** (int $gid)

現在のプロセスの実効グループ ID を設定します。 この関数は特権関数であり、実行するにはシステム上において適当な権限 (通常は root) が必要です。

パラメータ

gid

グループ ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 posix_setegid() の例

この例では、まず実効グループ ID を変更してから、それを表示します。

```
<?php
echo '実グループ ID は '.posix_getgid(); //20
posix_setegid(40);
echo '実グループ ID は '.posix_getgid(); //20
echo '実効グループ ID は '.posix_getegid(); //40
?>
```

参考

- posix_getgrgid() は、グループ ID からグループ名を取得します
- posix_getgid() は、実グループ ID を取得します
- posix_setgid() は、実効グループ ID を変更します

---

posix_seteuid

(PHP 4 >= 4.0.2, PHP 5)

posix_seteuid — 現在のプロセスの実効 UID を設定する

説明

bool **posix_seteuid** (int $uid)

現在のプロセスの実効ユーザ ID を設定します。 この関数は特権関数であり、実行するにはシステム上において適当な権限（通常は root）が必要です。

パラメータ

uid

ユーザ ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- posix_setgid()

---

posix_setgid

(PHP 4, PHP 5)

posix_setgid — 現在のプロセスの GID を設定する

説明

bool **posix_setgid** (int $gid)

現在のプロセスのグループ ID を設定します。 この関数は特権関数であり、実行するにはシステム上において適当な権限（通常は root）が必要です。 **posix_setgid()** を最初に、 posix_setuid() を最後にコールするのが、 関数コールの正しい順序となります。

注意: コール元がスーパーユーザの場合は、実効グループ ID も設定します。

パラメータ

gid

グループ ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 posix_setgid() の例

この例は、いったん変更した後で実効グループ ID を表示します。

```php
<?php
echo '実グループ ID は '.posix_getgid(); //20
posix_setgid(40);
echo '実グループ ID は '.posix_getgid(); //40
echo '実効グループ ID は '.posix_getegid(); //40
?>
```

参考

- posix_getgrgid() は、この情報からグループ名を取得します
- posix_getgid() は、実グループ ID を取得します

---

posix_setpgid

(PHP 4, PHP 5)

posix_setpgid — ジョブ制御のプロセスグループ ID を設定する

説明

bool **posix_setpgid** (int $pid , int $pgid)

プロセス pid をプロセスグループ pgid に加えます。

パラメータ

pid

プロセス ID。

pgid

プロセスグループ ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- プロセスグループおよびジョブ制御に関する詳細は、 POSIX.1 および POSIX システムのマニュアルページ setsid(2) を参照ください。

---

posix_setsid

(PHP 4, PHP 5)

posix_setsid — 現在のプロセスをセッションリーダーにする

説明

int **posix_setsid** (void)

現在のプロセスをセッションリーダーにします。

返り値

セッション ID、あるいはエラー時に -1 を返します。

参考

- プロセスグループおよびジョブ制御に関する詳細は、 POSIX.1 および POSIX システムのマニュアルページ setsid(2) を参照ください。

---

posix_setuid

(PHP 4, PHP 5)

posix_setuid — 現在のプロセスの UID を設定する

説明

bool **posix_setuid** (int $uid)

現在のプロセスの実際のユーザ ID を設定します。 この関数は特権関数であり、実行するにはシステム上において適当な権限（通常は root）が必要です。

パラメータ

uid

ユーザ ID。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 posix_setuid() の例

この例は、現在のユーザ ID を表示し、それを別のものに変更します。

```
<?php
echo posix_getuid()."¥n"; //10001
echo posix_geteuid()."¥n"; //10001
posix_setuid(10000);
echo posix_getuid()."¥n"; //10000
echo posix_geteuid()."¥n"; //10000
?>
```

参考

- posix_setgid()
- posix_seteuid()
- posix_getuid()
- posix_geteuid()

---

posix_strerror

(PHP 4 >= 4.2.0, PHP 5)

posix_strerror — 指定したエラー番号に対応するシステムのエラーメッセージを取得する

説明

string **posix_strerror** (int $errno)

指定したエラー番号 errno に対応する POSIX システムエラーメッセージを返します。 errno を取得するには、posix_get_last_error() をコールします。

パラメータ

errno

posix_get_last_error() が返す POSIX エラー番号。0 に設定すると、文字列 "Success" が返されます。

返り値

エラーメッセージを表す文字列を返します。

例

Example#1 posix_strerror() の例

この例では、存在しないプロセスを殺そうとしています。 そして、その結果として発生したエラーに対応するメッセージを表示します。

```
<?php
posix_kill(50,SIGKILL);
echo posix_strerror(posix_get_last_error())."¥n";
?>
```

上の例の出力は、たとえば以下のようになります。

```
No such process
```

参考

- posix_get_last_error()

---

posix_times

(PHP 4, PHP 5)

posix_times — プロセス時間を得る

説明

array **posix_times** (void)

現在の CPU 使用状況についての情報を取得します。

返り値

現在のプロセスの CPU 使用状況に関する情報を表す文字列を連想配列 として返します。連想配列のキーは次のようになります。

- ticks - リブートからの経過クロック数。
- utime - 現在のプロセスにより使用されているユーザ時間。
- stime - 現在のプロセスにより使用されているシステム時間。
- cutime - 現在のプロセスおよび子プロセスにより使用されているユーザ時間。
- cstime - 現在のプロセスおよび子プロセスにより使用されているシステム時間。

注意

警告

この関数の返す値は信頼できません。 時間が大きくなると負の値を返すこともあります。

例

Example#1 posix_times() の使用例

```
<?php

$times = posix_times();

print_r($times);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [ticks] => 25814410
     [utime] => 1
     [stime] => 1
     [cutime] => 0
     [cstime] => 0
)
```

---

posix_ttyname

(PHP 4, PHP 5)

posix_ttyname — 端末のデバイス名を調べる

説明

string **posix_ttyname** (int $fd)

ファイル記述子 fd 上でオープンしている現在の端末デバイスへの絶対パスを文字列で返します。

パラメータ

fd

ファイル記述子。

返り値

成功した場合に fd の絶対パスを表す文字列、 失敗した場合に **FALSE** を返します。

---

posix_uname

(PHP 4, PHP 5)

posix_uname — システム名を得る

説明

array **posix_uname** (void)

システムについての情報を取得します。

POSIX では、値のフォーマットに関して何の仮定も設けないことを規定しています。 例えば、バージョン番号が 3 桁の数字であることや、 この関数により返されるその他のものに依存してはいけません。

返り値

システムに関する情報を文字列の連想配列として返します。 連想配列のキーは、次のようになります。

- sysname - オペレーティングシステムの名前 (例 Linux)

- nodename - システムの名前 (例 valiant)
- release - オペレーティングシステムのリリース (例 2.2.10)
- version - オペレーティングシステムのバージョン (例 #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - システムアーキテクチャ (例 i586)
- domainname - DNS ドメイン名 (例 example.com)

domainname は、GNU の拡張機能で POSIX.1 には含まれていません。 このため、このフィールドは GNU システム上または GNU libc を使用している場合にのみ使用可能です。

例

Example#1 posix_uname() の使用例

```
<?php
$uname=posix_uname();
print_r($uname);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [sysname] => Linux
     [nodename] => funbox
     [release] => 2.6.20-15-server
     [version] => #2 SMP Sun Apr 15 07:41:34 UTC 2007
     [machine] => i686
)
```

目次

正規表現(regex)関数 (POSIX拡張サポート)

導入

ヒント

PHPは、PCRE関数によりPerl互換の構 文を使用する正規表現式もサポートします。これらの関数は、「ものぐさ」 マッチ、言明、条件付きサブパターン、そしてPOSIX拡張正規表現構文 でサポートされていない他の複数の機能をサポートします。

警告

これらの正規表現関数はバイナリセーフではありません。PCRE関数はバイナリセーフです。

正規表現は、複雑な文字列操作の際に使用されます。 PHPはPOSIX 1003.2で定義されたPOSIX拡張正規表現を使用します。 POSIX正規表現に関する詳細については、PHP配布ファイルのregexディレ クトリにある» regexのmanページを参照ください。 このページはmanpageフォーマットであり、読むには man /usr/local/src/regex/regex.7のようにします。

要件

外部ライブラリを必要としません。

インストール手順

警告

動作に関する知識がある場合以外は、TYPE を変更しないでください。

PHP で正規表現のサポートを有効にするには、 --with-regex=TYPE を指定して PHP の configure を行ってください。TYPE は system、apache、php のいずれかで、デフォルトでは php を使用します。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

例

Example#1 正規表現の例

```
// "abc" が $string のどこかにある場合に true を返す
ereg ("abc", $string);

// "abc" が $string の最初にある場合に true を返す
ereg ("^abc", $string);

// "abc" が $string の最後にある場合に true を返す
ereg ("abc$", $string);

// クライアントブラウザがNetscape 2, 3またはMSIE 3である場合にtrue を返す
eregi("(ozilla.[23]|MSIE.3)", $_SERVER["HTTP_USER_AGENT"]);

// 空白で区切られた3つ単語を
// $regs[1], $regs[2],$regs[3]に代入する
ereg ("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)", $string,$regs);

// <br /> タグを $string の先頭に挿入する
$string = ereg_replace ("^", "<br />", $string);

// <br /> タグを $string の最後に挿入する
$string = ereg_replace ("$", "<br />", $string);

// $string の改行文字を全て取り除く
$string = ereg_replace ("¥n", "", $string);
```

参考

Perl互換の構文を有する正規表現については、 PCRE関数を参照してください。 簡単なシェル形式のワイルドカードパターンマッチングがfnmatch()で提供されています。

---

ereg_replace

(PHP 4, PHP 5)

ereg_replace — 正規表現による置換を行う

説明

```
string ereg_replace ( string $pattern , string $replacement , string $string )
```

この関数は、string をスキャンして pattern にマッチするものを探し、 マッチしたテキストを replacement で置換します。

パラメータ

pattern

POSIX 拡張正規表現。

replacement

pattern の中に括弧でくくられた部分 文字列が含まれている場合、replacement の中に ¥¥数字 のような部分文字列を埋め込むこともできます。この部分は、 「数字」番目の括弧でくくられた部分文字列にマッチする文字列に 置き換えられます。また、¥¥0 は文字列全体を 指します。9 個までの部分文字列を使うことができます。括弧は 入れ子になっていても構いません。この場合は開き括弧 '(' が 最大 9 個まで使用可能です。

string

入力文字列。

返り値

置換後の文字列を返します。 マッチしなかった場合は、元の文字列をそのまま返します。

例

たとえば、以下のサンプルコードは "This was a test" と 3 回表示します。

Example#1 ereg_replace() の例

```
<?php

$string = "This is a test";
echo str_replace(" is", " was", $string);
echo ereg_replace("( )is", "¥¥1was", $string);
echo ereg_replace("(( )is)", "¥¥2was", $string);

?>
```

注意しなければならないのは、パラメータ replacement として整数値を使用する場合、 期待する結果が得られない可能性があるということです。これは、 **ereg_replace()** がその数値を文字コードとして 解釈し使用するためです。例えば、次のようになります。

Example#2 ereg_replace() の例

```
<?php
/* これは期待した通りに動作しません。 */
$num = 4;
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* 出力: 'This string has    words.' */

/* これは動作します。 */
$num = '4';
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* 出力: 'This string has 4 words.' */
?>
```

Example#3 URL をリンクで置換する

```
<?php
$text = ereg_replace("[[:alpha:]]+://[^<>[:space:]]+[[:alnum:]/]",
                     "<a  href=¥"¥¥0¥">¥¥0</a>", $text);

?>
```

注意

ヒント

Perl 互換の正規表現構文を使用する preg_replace() のほうが、 **ereg_replace()** より高速に動作することがよくあります。

参考

- ereg()
- eregi()
- eregi_replace()
- str_replace()
- preg_match()

---

ereg

(PHP 4, PHP 5)

ereg — 正規表現によるマッチングを行う

説明

```
int ereg ( string $pattern , string $string [, array &$regs ] )
```

pattern で指定した正規表現 により、大文字小文字を区別して string を検索します。

パラメータ

pattern

大文字小文字を区別する正規表現。

string

入力文字列。

regs

pattern の括弧でくくられた部分文字列に マッチし、かつこの関数が 3 番目の引数 regs を 指定してコールされた場合、マッチした部分が配列 regs に格納されます。

$regs[1] は最初の左括弧が始まる部分文字列を保持、 $regs[2] は二番目の左括弧が始まる部分文字列を保持、 といったようになっています。 $regs[0] は マッチした文字列全体のコピーを保持しています。

返り値

string の中で pattern がマッチした場合にはマッチした文字列の長さを返し、マッチしなかった場合 またはエラーとなった場合は **FALSE** を返します。

オプションのパラメータ regs が渡されなかったり マッチした文字列の長さが 0 だったりした場合は、この関数は 1 を返します。

例

Example#1 ereg() の例

以下のサンプルコードは、ISO フォーマット(YYYY-MM-DD) で格納されている 日付を DD.MM.YYYY フォーマットで表示するものです。

```
<?php
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Invalid date format: $date";
}
?>
```

注意

> **注意**: Perl 互換の正規表現構文を使用する preg_match() のほうが、多くの場合 **ereg()** よりも速く動作します。
>
> **注意**: PHP 4.1.0 以前(4.1.0 を含む)のバージョンでは、$regs にはちょうど 10 個の要素が代入されました。これは、実際にマッチした 括弧付きのサブ文字列が 10 より多くても少なくても同じでした。ただし、 **ereg()** にはより多くの部分文字列にマッチする 能力があります。マッチするものが見付からなかった場合、 $regs は、**ereg()** により 書き換えられません。

参考

- eregi()
- ereg_replace()
- eregi_replace()
- preg_match()
- strpos()
- strstr()
- quotemeta()

---

eregi_replace

(PHP 4, PHP 5)

eregi_replace — 大文字小文字を区別せずに正規表現による置換を行う

説明

string **eregi_replace** (string $pattern , string $replacement , string $string)

この関数は、アルファベット文字をマッチングさせる際に大文字 小文字の区別をしないこと以外は ereg_replace() と同じです。

パラメータ

pattern

POSIX 拡張正規表現。

replacement

pattern の中に括弧でくくられた部分 文字列が含まれている場合、replacement の中に ¥¥数字 のような部分文字列を埋め込むこともできます。この部分は、 「数字」番目の括弧でくくられた部分文字列にマッチする文字列に 置き換えられます。また、¥¥0 は文字列全体を 指します。9 個までの部分文字列を使うことができます。括弧は 入れ子になっていても構いません。この場合は開き括弧 '(' が 最大 9 個まで使用可能です。

string

入力文字列。

返り値

置換後の文字列を返します。 マッチしなかった場合は、元の文字列をそのまま返します。

例

Example#1 検索結果の強調表示

```
<?php
$pattern = '(>[^<]*)('. quotemeta($_GET['search']) .')';
$replacement = '¥¥1<span class="search">¥¥2</span>';
$body = eregi_replace($pattern, $replacement, $body);
?>
```

参考

- ereg()
- eregi()
- ereg_replace()

---

eregi

(PHP 4, PHP 5)

eregi — 大文字小文字を区別せずに正規表現によるマッチングを行う

説明

int **eregi** (string $pattern , string $string [, array &$regs])

この関数は、アルファベット文字をマッチングさせる際に 大文字小文字の区別をしないこと以外は ereg() と同じです。

パラメータ

pattern

大文字小文字を区別しない正規表現。

string

入力文字列。

regs

pattern の括弧でくくられた部分文字列に マッチし、かつこの関数が 3 番目の引数 regs を 指定してコールされた場合、マッチした部分が配列 regs に格納されます。

$regs[1] は最初の左括弧が始まる部分文字列を保持、 $regs[2] は二番目の左括弧が始まる部分文字列を保持、 といったようになっています。 $regs[0] は マッチした文字列全体のコピーを保持しています。

返り値

string の中で pattern がマッチした場合にはマッチした文字列の長さを返し、マッチしなかった場合 またはエラーとなった場合は **FALSE** を返します。

オプションのパラメータ regs が渡されなかったり マッチした文字列の長さが 0 だったりした場合は、この関数は 1 を返します。

例

Example#1 eregi() の例

```
<?php
$string = 'XYZ';
if (eregi('z', $string)) {
    echo "'$string' は、'z' または 'Z' を含んでいます!";
}
?>
```

参考

- ereg()
- ereg_replace()
- eregi_replace()
- stripos()
- stristr()

---

split

(PHP 4, PHP 5)

split — 正規表現により文字列を分割し、配列に格納する

説明

array **split** (string $pattern , string $string [, int $limit])

string を、正規表現によって配列に分割します。

パラメータ

pattern

大文字小文字を区別する正規表現。

正規表現で特殊文字とみなされる文字を分割したい場合、それらを まずエスケープする必要があります。**split()**（または他の正規表現関数に関して）の動作が何か変であると感じたならば、 PHP 配布ファイルの regex/ サブディレクトリにあるファイル regex.7 を読んでください。このファイルは man ページ形式となっており、読むには **man /usr/local/src/regex/regex.7** のようにします。

string

入力文字列。

limit

limit が指定された場合、返される配列には 最大 limit の要素が含まれます。この場合、 最後の要素には string の残りの部分が全て 入っています。

返り値

文字列の配列を返します。配列の各要素は、大文字小文字を区別する正規表現 pattern により区切られた string 中の部分文字列です。

pattern が n 回 現れる場合、返される配列には、 n+1 個のアイテムが 含まれます。例えば、pattern が現れない場合、 1 個の要素のみを有する配列が返されます。もちろん、これは エラーを生じた場合、**split()** は **FALSE** を返します。

例

Example#1 split() の例

/etc/passwd の行から最初の 4 つのフィールドを 取り出すには、以下のようにします。

```
<?php
list($user, $pass, $uid, $gid, $extra) =
    split(":", $passwd_line, 5);
?>
```

Example#2 split() の例

スラッシュ、ドット、ハイフンをデリミタとすると日付を処理する例を 以下に示します。

```
<?php
// デリミタはスラッシュ、ドット、ハイフンのどれかです。
$date = "04/30/1973";
list($month, $day, $year) = split('[/.-]', $date);
echo "Month: $month; Day: $day; Year: $year<br />¥n";
?>
```

注意

ヒント

Perl 互換の正規表現構文を使用する preg_split() は、往々にして **split()** よりも速い代替案となります。 正規表現の威力が必要ないのであれば、explode() を使用するほうがより高速です。これは正規表現エンジンの オーバーヘッドを受けません。

ヒント

Perl の **@chars = split('', $str)** と同等の処理をする方法を知りたい場合は、 preg_split() あるいは str_split() の例を参照ください。

参考

- preg_split()
- spliti()
- str_split()
- explode()
- implode()
- chunk_split()
- wordwrap()

---

spliti

(PHP 4 >= 4.0.1, PHP 5)

spliti — 大文字小文字を区別しない正規表現により文字列を分割し、配列に格納する

説明

array **spliti** (string $pattern , string $string [, int $limit])

string を、正規表現によって配列に分割します。

この関数は、アルファベット文字にマッチさせる際に大文字小文字を 区別しないこと以外は、split() と同じです。

パラメータ

pattern

大文字小文字を区別しない正規表現。

正規表現で特殊文字とみなされる文字を分割したい場合、それらを まずエスケープする必要があります。**spliti()** (または他の正規表現関数に関して)の動作が何か変であると感じたならば、 PHP 配布ファイルの regex/ サブディレクトリにあるファイル regex.7 を読んでください。このファイルは man ページ形式となっており、読むには **man /usr/local/src/regex/regex.7** のようにします。

string

入力文字列。

limit

limit が指定された場合、返される配列には 最大 limit の要素が含まれます。この場合、 最後の要素には string の残りの部分が全て 入っています。

返り値

文字列の配列を返します。配列の各要素は、大文字小文字を区別する正規表現 pattern により区切られた string 中の部分文字列です。

pattern が n 回 現れる場合、返される配列には、 n+1 個のアイテムが 含まれます。例えば、pattern が現れない場合、 1 個の要素のみを有する配列が返されます。もちろん、これは string が空の場合でも有効です。 エラーを生じた場合、**spliti()** は **FALSE** を返します。

例

この例では 'a' を区切り文字として文字列を分割します。

Example#1 spliti() の例

```
<?php
$string = "aBBBaCCCADDDaEEEaGGGA";
$chunks = spliti ("a", $string, 5);
print_r($chunks);
?>
```

上の例の出力は以下となります。

```
Array
(
   [0] =>
   [1] => BBB
   [2] => CCC
   [3] => DDD
   [4] => EEEaGGGA
)
```

参考

- preg_split()
- split()
- explode()
- implode()

---

sql_regcase

(PHP 4, PHP 5)

sql_regcase — 大文字小文字を区別しないマッチングのための正規表現を作成する

説明

string **sql_regcase** (string $string)

大文字小文字を区別しないマッチングのための正規表現を作成します。

パラメータ

string

入力文字列。

返り値

大文字小文字を区別せず string にマッチする、 有効な正規表現式を返します。この表現式は、string の中の個々のアルファベットを '[]' 表現に変換したものです。 この '[]' 表現は、大文字と小文字がそれぞれ含まれます。 アルファベット以外の文字の場合は、元々の文字がそのまま残ります。

例

Example#1 sql_regcase() の例

```
<?php
```

```
echo sql_regcase("Foo - bar.");
?>
```

上の例の出力は以下となります。

```
[Ff][Oo][Oo] - [Bb][Aa][Rr].
```

この関数は、大文字小文字を区別する正規表現しかサポートしない製品において、 大文字小文字を区別しないパターンマッチングを行いたい場合に役立ちます。

目次

PostgreSQL 関数

導入

PostgreSQL データベースはオープンソースの製品であり、無料で 使用可能です。Postgres は元々 UCB（カリフォルニア大学バークレイ校） コンピュータ・サイエンス学部で開発されたものです。この Postgres は、 現在いくつかの商用データベースにおいてサポートされつつある オブジェクトリレーショナルデータベース的概念の多くの先駆けでした。 Postgres は、SQL92/SQL99 言語サポート・トランザクション・参照整合性・ ストアドプロシージャ・拡張可能な型を提供しています。PostgreSQL は、 バークレイ校での Postgres のオリジナルコードの、オープンソースの 子孫にあたります。

要件

PostgreSQL サポートを使用するには、PostgreSQL 6.5 以降が必要です。 PostgreSQL 8.0 以降では PostgreSQL モジュールの全ての機能を使用可能 です。PostgreSQL は、マルチバイト文字エンコーディングを含む多くの 文字エンコーディングをサポートしています。現在のバージョン及び PostgreSQLに関するより詳細な情報は、 » http://www.postgresql.org/ と » PostgreSQL Documentation で入手可能です。

インストール手順

PostgreSQLサポートを利用可能とするには、PHPコンパイル時に --with-pgsql[=DIR] を指定することが必要です。 共有オブジェクトモジュールが利用可能な場合、php.ini の extension ディレクティブ または dl() 関数によりPostgreSQLモジュール をロードすることが可能です。

実行時設定

php.ini の設定により動作が変化します。

PostgreSQL設定オプション

名前	デフォルト	変更の可否	変更履歴
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM	
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM	
pgsql.max_links	"-1"	PHP_INI_SYSTEM	
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM	PHP 4.2.0 以降で有効です。
pgsql.ignore_notice	"0"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。
pgsql.log_notice	"0"	PHP_INI_ALL	PHP 4.3.0 以降で有効です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

pgsql.allow_persistent boolean

持続的な Postgres 接続を可能にするかどうか。

pgsql.max_persistent integer

プロセス毎の持続的 Postgres 接続の最大数。

pgsql.max_links integer

持続的接続を含むプロセス毎の Postgres 接続の最大数。

pgsql.auto_reset_persistent integer

pg_pconnect() で作成した持続的接続の障害を 検出する。少々のオーバーヘッドを要します。

pgsql.ignore_notice integer

PostgreSQL バックエンドの通知メッセージを無視するかどうか。

pgsql.log_notice integer

PostgreSQL バックエンドの通知メッセージをログに記録するかしないか。 ログに記録するには、PHP ディレクティブ pgsql.ignore_notice を off にする必要があります。

リソース型

PostgreSQL モジュールで使用されるリソース型は 2 種類あります。ひとつは データベース接続のリンク ID で、もうひとつはクエリの結果を保持する リソースです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

PGSQL_ASSOC (integer)
pg_fetch_array() に渡します。 フィールド名と値の連想配列を返します。
PGSQL_NUM (integer)
pg_fetch_array() に渡します。 フィールド番号と値の数値添字配列を返します。
PGSQL_BOTH (integer)
pg_fetch_array() に渡します。 フィールド値の配列を、数値添字（フィールド番号で）と 連想配列（フィールド名で）の両方で返します。
PGSQL_CONNECT_FORCE_NEW (integer)
pg_connect() に渡し、既存の同一接続を無視して 新しい接続を確立させるようにします。
PGSQL_CONNECTION_BAD (integer)
pg_connection_status() から返され、データベースとの 接続が不正な状態になっていることを示します。
PGSQL_CONNECTION_OK (integer)
pg_connection_status() から返され、データベースとの 接続が正常であることを示します。
PGSQL_SEEK_SET (integer)
pg_lo_seek() に渡します。シーク操作は オブジェクトの先頭から始められます。
PGSQL_SEEK_CUR (integer)
pg_lo_seek() に渡します。シーク操作は カレントの位置から始められます。
PGSQL_SEEK_END (integer)
pg_lo_seek() に渡します。シーク操作は オブジェクトの最後から始められます。
PGSQL_EMPTY_QUERY (integer)
pg_result_status() から返されます。 サーバに送信された文字列が空でした。
PGSQL_COMMAND_OK (integer)
pg_result_status() から返されます。 コマンドは正常に終了し、何もデータを返しませんでした。
PGSQL_TUPLES_OK (integer)
pg_result_status() から返されます。 コマンドは正常に終了し、データを返しました （SELECT や SHOW など）。
PGSQL_COPY_OUT (integer)
pg_result_status() から返されます。 （サーバからの）データのコピーが始まりました。
PGSQL_COPY_IN (integer)
pg_result_status() から返されます。 （サーバへの）データのコピーが始まりました。
PGSQL_BAD_RESPONSE (integer)
pg_result_status() から返されます。 サーバからの応答を判別できませんでした。
PGSQL_NONFATAL_ERROR (integer)
pg_result_status() から返されます。 致命的ではないエラー（通知や警告など）が発生しました。
PGSQL_FATAL_ERROR (integer)
pg_result_status() から返されます。 致命的なエラーが発生しました。
PGSQL_TRANSACTION_IDLE (integer)
pg_transaction_status() から返されます。 接続は現在アイドル状態で、トランザクション内ではありません。
PGSQL_TRANSACTION_ACTIVE (integer)
pg_transaction_status() から返されます。 接続内でコマンドを実行中です。クエリが接続を通じて送信され、 まだ完了していません。
PGSQL_TRANSACTION_INTRANS (integer)
pg_transaction_status() から返されます。 接続は現在アイドル状態で、トランザクションブロック内にあります。
PGSQL_TRANSACTION_INERROR (integer)
pg_transaction_status() から返されます。 接続は現在アイドル状態で、トランザクション処理に失敗しています。
PGSQL_TRANSACTION_UNKNOWN (integer)
pg_transaction_status() から返されます。 接続が正常ではありません。
PGSQL_DIAG_SEVERITY (integer)
pg_result_error_field() に渡します。 深刻度です。その内容は ERROR、 FATAL、あるいは PANIC （エラーメッセージ内）、あるいは WARNING、 NOTICE、DEBUG、 INFO、あるいは LOG （通知メッセージ内）、あるいはこれらの各国語版のうちのどれかです。 常に存在します。
PGSQL_DIAG_SQLSTATE (integer)
pg_result_error_field() に渡します。 エラーの SQLSTATE コードです。SQLSTATE コードは発生したエラーの種別を 特定します。これは、データベースエラーに対してフロントエンドの アプリケーションが適切な操作（エラー処理など）をできるようにするために 使用されます。このフィールドはローカライズされていません。また、 常に存在します。
PGSQL_DIAG_MESSAGE_PRIMARY (integer)
pg_result_error_field() に渡します。 人間が判読できる最初のエラーメッセージ（たいてい 1 行）です。常に存在します。
PGSQL_DIAG_MESSAGE_DETAIL (integer)
pg_result_error_field() に渡します。 オプションの 2 番目のエラーメッセージで、問題に対する詳細な情報を 含みます。複数行にまたがることもあります。
PGSQL_DIAG_MESSAGE_HINT (integer)
pg_result_error_field() に渡します。 オプションのメッセージで、問題に対してどのように対応すべきかを指摘 します。エラーの詳細情報と違う点は、発生した事実ではなくアドバイス （時に不適切な場合もある）をするところです。複数行にまたがることも あります。
PGSQL_DIAG_STATEMENT_POSITION (integer)
pg_result_error_field() に渡します。 エラーカーソルの位置を、もとのステートメント中の位置として表す 整数値を保持する文字列です。 最初の文字の位置は 1 で、それ以降（バイト数ではなく）文字数で 位置を数えます。
PGSQL_DIAG_INTERNAL_POSITION (integer)
pg_result_error_field() に渡します。 **PG_DIAG_STATEMENT_POSITION** と同じ定義ですが、 こちらはカーソル位置が内部で生成されたコマンドを参照している場合に 使用されます。このフィールドが存在する場合は、常に **PG_DIAG_INTERNAL_QUERY** フィールドも存在します。
PGSQL_DIAG_INTERNAL_QUERY (integer)
pg_result_error_field() に渡します。 失敗した内部生成コマンドのテキストです。例としては、PL/pgSQL 関数で 発行された SQL クエリなどがあてはまります。
PGSQL_DIAG_CONTEXT (integer)
pg_result_error_field() に渡します。 エラーが発生した状況を指定します。現在ここに含まれているのは、 アクティブなプロシージャの関数や内部生成クエリのコール履歴です。 これはエントリごとに 1 行のデータとなっており、直近のデータが 先頭にきます。
PGSQL_DIAG_SOURCE_FILE (integer)
pg_result_error_field() に渡します。 エラーが報告された箇所の、PostgreSQL ソースコードでのファイル名です。
PGSQL_DIAG_SOURCE_LINE (integer)
pg_result_error_field() に渡します。 エラーが報告された箇所の、PostgreSQL ソースコードでの行番号です。
PGSQL_DIAG_SOURCE_FUNCTION (integer)
pg_result_error_field() に渡します。 エラーが報告された箇所の、PostgreSQL ソースコードでの関数名です。
PGSQL_ERRORS_TERSE (integer)
pg_set_error_verbosity() に渡します。 返されるメッセージを指定します。ここには深刻度・短い説明および 発生位置のみを含めます。通常は 1 行に収まるでしょう。
PGSQL_ERRORS_DEFAULT (integer)
pg_set_error_verbosity() に渡します。 デフォルトで返されるメッセージで、上の内容に加えて詳細・ ヒントあるいは詳細情報を含めます（複

数行にまたがるかもしれません）。
PGSQL_ERRORS_VERBOSE (integer)
pg_set_error_verbosity() に渡します。 詳細モードでのメッセージで、有効なフィールドをすべて含めます。
PGSQL_STATUS_LONG (integer)
pg_result_status() に渡します。 結果として数値が要求されていることを示します。
PGSQL_STATUS_STRING (integer)
pg_result_status() に渡します。 結果としてテキストのコマンドタグが要求されていることを示します。
PGSQL_CONV_IGNORE_DEFAULT (integer)
pg_convert() に渡します。 変換時に、テーブルのデフォルト値を無視します。
PGSQL_CONV_FORCE_NULL (integer)
pg_convert() に渡します。 空文字列に対して、SQL の NULL を使用します。
PGSQL_CONV_IGNORE_DEFAULT (integer)
pg_convert() に渡します。 **NULL** を SQL の NOT NULL に変換しないように します。

注意

> **注意**: すべての関数がすべての構築環境でサポートされるわけではありません。 サポートされる関数は、使用する libpq (PostgreSQL の C クライアント ライブラリ) のバージョンと libpq のコンパイル方法に依存します。 もし PHP の PostgreSQL 拡張モジュールに足りない関数がある場合、 その原因は libpq はその関数をサポートしていないことです。
>
> **注意**: ほとんどの PostgreSQL 関数は、オプションの第 1 引数として connection を受け付けます。もしこれを 指定しなかった場合、直近にオープンされた接続を使用します。 そのような接続が存在しなかった場合、関数は **FALSE** を返します。
>
> **注意**: PostgreSQL は、オブジェクトの生成時やクエリの実行時に 識別子（例: テーブル名・カラム名）を自動的に小文字に変換します。 この自動変換を防ぐには、識別子をダブルクォート（""）でエスケープする 必要があります。
>
> **注意**: PostgreSQL には、データベースのスキーマ情報（例: データベース内の すべてのテーブルなど）を取得するための特別なコマンドがありません。 その代わりに、PostgreSQL 7.4 以降では information_schema という標準スキーマが存在し、必要な情報が検索しやすい形式で格納されています。 詳しい情報は » PostgreSQL ドキュメンテーション を参照ください。

例

この例では、PostgreSQL への接続・クエリの実行・結果の表示 そして切断の方法を説明します。

Example#1 PostgreSQL 拡張モジュールの概要

```
<?php
// 接続し、データベースを選択する
$dbconn = pg_connect("host=localhost dbname=publishing user=www password=foo")
      or die('Could not connect: ' . pg_last_error());

// SQL クエリを実行する
$query = 'SELECT * FROM authors';
$result = pg_query($query) or die('Query failed: ' . pg_last_error());

// 結果を HTML で表示する
echo "<table>¥n";
while ($line = pg_fetch_array($result, null, PGSQL_ASSOC)) {
     echo "¥t<tr>¥n";
     foreach ($line as $col_value) {
         echo "¥t¥t<td>$col_value</td>¥n";
     }
     echo "¥t</tr>¥n";
}
echo "</table>¥n";

// 結果セットを開放する
pg_free_result($result);

// 接続をクローズする
pg_close($dbconn);
?>
```

---

pg_affected_rows

(PHP 4 >= 4.2.0, PHP 5)

pg_affected_rows — 変更されたレコード(タプル)の数を返す

説明

int **pg_affected_rows** (resource $result)

pg_affected_rows() は、INSERT, UPDATE, DELETE クエリにより変更されたタプル(インスタンス/レコード/行)の数を 返します。

> **注意**: この関数は、以前は **pg_cmdtuples()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

返り値

クエリによって変更された行の数を返します。もし変更されたタプルがない場合は 0 を返します。

例

Example#1 pg_affected_rows() の例

```
<?php
```

```
$result = pg_query($conn, "INSERT INTO authors VALUES ('オーウェル', 2002, '動物農場')");

$cmdtuples = pg_affected_rows($result);

echo $cmdtuples . " タプルが変更されました。\n";
?>
```

上の例の出力は以下となります。

```
1 タプルが変更されました。
```

参考

- pg_query()
- pg_query_params()
- pg_execute()
- pg_num_rows()

---

pg_cancel_query

(PHP 4 >= 4.2.0, PHP 5)

pg_cancel_query — 非同期クエリを取り消す

説明

bool **pg_cancel_query** (resource $connection)

pg_cancel_query() は、 pg_send_query()・pg_send_query_params() あるいは pg_send_execute() により送信された非同期クエリを キャンセルします。pg_query() により実行されたクエリを キャンセルすることはできません。

パラメータ

connection

PostgreSQL データベース接続リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_cancel_query() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

   if (!pg_connection_busy($dbconn)) {
       pg_send_query($dbconn, "select * from authors; select count(*) from authors;");
   }

   $res1 = pg_get_result($dbconn);
   echo "First call to pg_get_result(): $res1¥n";
   $rows1 = pg_num_rows($res1);
   echo "$res1 has $rows1 records¥n¥n";

  // 実行中のクエリをキャンセルする。もしまだ実行中なら、2 番目のクエリが
  // 対象となるだろう。
  pg_cancel_query($dbconn);
?>
```

上の例の出力は以下となります。

```
First call to pg_get_result(): Resource id #3
Resource id #3 has 3 records
```

参考

- pg_send_query()
- pg_connection_busy()

---

pg_client_encoding

(PHP 4 >= 4.0.3, PHP 5)

pg_client_encoding — クライアントのエンコーディングを取得する

説明

```
string pg_client_encoding ([ resource $connection ] )
```

PostgreSQL は、ある種の文字セットについてはサーバとクライアントの間の 自動文字コード変換をサポートしています。 **pg_client_encoding()** は、クライアントの エンコーディングを文字列として返します。返される値は、標準の PostgreSQL エンコーディング識別子のなかのひとつとなります。

> **注意**: この関数を使用するには、PHP 4.0.3 以降と PostgreSQL 7.0 以降が必要です。 libpq がマルチバイトエンコーディングのサポートを有効にせずに コンパイルされている場合、**pg_client_encoding()** は常に SQL_ASCII を返します。サポートされる エンコーディングは PostgreSQL のバージョンに依存します。サポートされる エンコーディングの詳細については PostgreSQL のドキュメントを参照ください。
> この関数は、以前は **pg_clientencoding()** と呼ばれていました。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

クライアントのエンコーディングを返します。エラー時には **FALSE** を返します。

例

Example#1 pg_client_encoding() の例

```
<?php
// $conn の接続先のエンコーディングは ISO-8859-1 であるとする
$encoding = pg_client_encoding($conn);

echo "Client encoding is: ", $encoding, "¥n";
?>
```

上の例の出力は以下となります。

```
Client encoding is: ISO-8859-1
```

参考

- pg_set_client_encoding()

---

pg_close

(PHP 4, PHP 5)

pg_close — PostgreSQL 接続をクローズする

説明

```
bool pg_close ([ resource $connection ] )
```

pg_close() は、 connection リソースで指定した PostgreSQL データベースへの持続的でない接続を閉じます。

> **注意**: 持続的でない接続はスクリプトの実行終了時に自動的にクローズされるため、 **pg_close()** は通常は必要ありません。

接続の中でラージオブジェクトをオープンしている場合は、すべての ラージオブジェクトリソースをクローズするまで接続を閉じないでください。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_close() の例

```
<?php
$dbconn = pg_connect("host=localhost port=5432 dbname=mary")
    or die("Could not connect");
echo "Connected successfully";
pg_close($dbconn);
?>
```

上の例の出力は以下となります。

```
Connected successfully
```

参考

- pg_connect()

---

pg_connect

(PHP 4, PHP 5)

pg_connect — PostgreSQL 接続をオープンする

説明

resource **pg_connect** (string $connection_string [, int $connect_type])

pg_connect() は、connection_string で指定された情報を用いてデータベースとの接続をオープンします。

同じ connection_string 引数で 2 回 **pg_connect()** 関数がコールされた場合は、 connect_type に **PGSQL_CONNECT_FORCE_NEW** を指定していない限り 既存の接続が返されます。

複数のパラメータをサポートする古い構文 **$conn = pg_connect("host", "port", "options", "tty", "dbname")** は推奨されません。

パラメータ

connection_string

すべてデフォルトのパラメータを使用する場合には connection_string を空にすることが可能です。 または 1 つ以上のパラメータを空白で区切って指定することも可能です。 個々のパラメータは keyword = value の形式で 設定します。等号の前後の空白はあってもなくてもかまいません。空の値や空白を含む値を指定する場合は、その値をシングルクォートで 囲みます（例: keyword = 'a value'）。値の中に シングルクォートやバックスラッシュが含まれる場合は、それらを バックスラッシュでエスケープする必要があります（例: \' および \\）。

現在利用できるパラメータは以下のとおりです。 host , hostaddr , port , dbname , user , password , connect_timeout , options , tty （無視されます）, sslmode , requiressl （非推奨。代わりに sslmode を推奨します） および service 。これらのうち実際にどの パラメータが使えるかは、PostgreSQL のバージョンに依存します。

connect_type

PGSQL_CONNECT_FORCE_NEW が渡された場合は、 たとえ connection_string が既存の接続と まったく同一であっても新しい接続をオープンします。

返り値

成功した場合に PostgreSQL の接続リソース、失敗した場合に **FALSE** を返します。

例

Example#1 pg_connect() の使用法

```
<?php
$dbconn = pg_connect("dbname=mary");
// "mary"という名前のデータベースに接続

$dbconn2 = pg_connect("host=localhost port=5432 dbname=mary");
// "localhost"のポート"5432"にて"mary"という名前のデータベースに接続

$dbconn3 = pg_connect("host=sheep port=5432 dbname=mary user=lamb password=foo");
// ユーザ名とパスワードを指定してホスト"sheep"上の"mary"という名前のデータベースに接続

$conn_string = "host=sheep port=5432 dbname=test user=lamb password=bar";
$dbconn4 = pg_connect($conn_string);
// ユーザ名とパスワードを指定してホスト"sheep"上の"test"という名前のデータベースへ接続
?>
```

参考

- pg_pconnect()
- pg_close()
- pg_host()
- pg_port()
- pg_tty()
- pg_options()
- pg_dbname()

---

pg_connection_busy

(PHP 4 >= 4.2.0, PHP 5)

pg_connection_busy — 接続がビジーかどうか調べる

説明

bool **pg_connection_busy** (resource $connection)

pg_connection_busy() は、接続がビジーかどうかを調べます。 接続がビジーである場合、以前のクエリがまだ実行中です。 もしこの接続に pg_get_result() がコールされた場合、 それはブロックされます。

パラメータ

connection

PostgreSQL データベースの接続リソース。

返り値

接続がビジーの場合に **TRUE** 、そうでない場合に **FALSE** を返します。

例

Example#1 pg_connection_busy() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");
  $bs = pg_connection_busy($dbconn);
  if ($bs) {
       echo 'connection is busy';
  } else {
      echo 'connection is not busy';
  }
?>
```

参考

- pg_connection_status()
- pg_get_result()

---

pg_connection_reset

(PHP 4 >= 4.2.0, PHP 5)

pg_connection_reset — 接続をリセット(再接続)する

説明

bool **pg_connection_reset** (resource $connection)

pg_connection_reset() は接続をリセットします。 エラーからの復旧の際に有用です。

パラメータ

connection

PostgreSQL データベースの接続リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_connection_reset() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");
  $dbconn2 = pg_connection_reset($dbconn);
  if ($dbconn2) {
      echo "reset successful¥n";
  } else {
      echo "reset failed¥n";
  }
?>
```

参考

- pg_connect()
- pg_pconnect()
- pg_connection_status()

---

pg_connection_status

(PHP 4 >= 4.2.0, PHP 5)

pg_connection_status — 接続ステータスを取得する

説明

int **pg_connection_status** (resource $connection)

pg_connection_status() は、 connection で指定した接続のステータスを返します。

パラメータ

connection

PostgreSQL データベースの接続リソース。

返り値

PGSQL_CONNECTION_OK あるいは **PGSQL_CONNECTION_BAD** 。

例

Example#1 pg_connection_status() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");
  $stat = pg_connection_status($dbconn);
  if ($stat === PGSQL_CONNECTION_OK) {
      echo 'Connection status ok';
  } else {
      echo 'Connection status bad';
  }
?>
```

参考

- pg_connection_busy()

---

pg_convert

(PHP 4 >= 4.3.0, PHP 5)

pg_convert — 連想配列の値を、SQL 文として実行可能な形式に変換する

説明

array **pg_convert** (resource $connection , string $table_name , array $assoc_array [, int $options])

pg_convert() は、assoc_array の中の値をチェックし、SQL 文の中で使用可能な形式に変換します。 少なくとも assoc_array の要素数以上のカラムを持つ テーブル table_name が存在することが前提条件と なります。assoc_array の要素名が table_name のフィールド名と一致し、また 要素に対応する値はフィールドのデータ型と互換性がなければなりません。 成功した場合に変換後の値の配列、それ以外の場合に **FALSE** を返します。

> **注意**: 論理型のフィールドが table_name にあった場合、 対応する assoc_array では定数 **TRUE** を使用 しないでください。これは文字列 'TRUE' に変換されてしまい、PostgreSQL では論理型として無効な形式になります。代わりに t、true、1、y、yes のうちのひとつを用いてください。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

変換対象となるテーブルの名前。

assoc_array

変換されるデータ。

options

PGSQL_CONV_IGNORE_DEFAULT, **PGSQL_CONV_FORCE_NULL** あるいは **PGSQL_CONV_IGNORE_NOT_NULL** の組み合わせ。

返り値

変換された値の配列を返します。エラー時には **FALSE** を返します。

例

Example#1 pg_convert() の例

```
<?php
   $dbconn = pg_connect('dbname=foo');

  $tmp = array(
      'author' => 'Joe Thackery',
      'year' => 2005,
      'title' => 'My Life, by Joe Thackery'
  );
```

```
  $vals = pg_convert($dbconn, 'authors', $tmp);
?>
```

参考

- pg_meta_data()

---

pg_copy_from

(PHP 4 >= 4.2.0, PHP 5)

pg_copy_from — 配列からテーブルに挿入する

説明

bool **pg_copy_from** (resource $connection , string $table_name , array $rows [, string $delimiter [, string $null_as]])

pg_copy_from() は、rows の内容をテーブルに挿入します。レコードを挿入するために、内部では COPY FROM SQL コマンドを発行します。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

rows をコピーするテーブルの名前。

rows

table_name にコピーするデータの配列。 rows の個々の値が table_name のひとつの行となります。rows の個々の値は、 それぞれのフィールドに対応する値が区切り文字で区切られており、最後は 改行で終了していなければなりません。

delimiter

rows の要素内で、各フィールドに対応する値を 区切る文字。デフォルトは TAB です。

null_as

rows の中で、SQL の NULL をどのように表現するか。デフォルトは \N ("\\N") です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_copy_from() の例

```
<?php
   $db = pg_connect("dbname=publisher") or die("Could not connect");

   $rows = pg_copy_to($db, $table_name);

   pg_query($db, "DELETE FROM $table_name");

   pg_copy_from($db, $table_name, $rows);
?>
```

参考

- pg_copy_to()

---

pg_copy_to

(PHP 4 >= 4.2.0, PHP 5)

pg_copy_to — 配列にテーブルをコピーする

説明

array **pg_copy_to** (resource $connection , string $table_name [, string $delimiter [, string $null_as]])

pg_copy_to() は、テーブルを配列にコピーします。 レコードを取得するために、内部では COPY TO SQL コマンドを発行します。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

データを rows にコピーするテーブルの名前。

delimiter

rows の要素内で、各フィールドに対応する値を 区切る文字。デフォルトは TAB です。

null_as

rows の中で、SQL の NULL をどのように表現するか。デフォルトは \N ("\\N") です。

返り値

COPY されたデータが 1 行ごとにひとつの要素と なっている配列を返します。失敗した場合には FALSE を返します。

例

Example#1 pg_copy_to() の例

```
<?php
   $db = pg_connect("dbname=publisher") or die("Could not connect");

   $rows = pg_copy_to($db, $table_name);

   pg_query($db, "DELETE FROM $table_name");

   pg_copy_from($db, $table_name, $rows);
?>
```

参考

- pg_copy_from()

---

pg_dbname

(PHP 4, PHP 5)

pg_dbname — データベース名を取得する

説明

string **pg_dbname** ([resource $connection])

pg_dbname() は、 PostgreSQL connection リソースで指定した データベースの名前を返します。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

connection が指すデータベースの名前を表す 文字列を返します。エラー時には FALSE を返します。

例

Example#1 pg_dbname() の例

```
<?php
   error_reporting(E_ALL);

  pg_connect("host=localhost port=5432 dbname=mary");
  echo pg_dbname(); // mary
?>
```

---

pg_delete

(PHP 4 >= 4.3.0, PHP 5)

pg_delete — レコードを削除する

説明

mixed **pg_delete** (resource $connection , string $table_name , array $assoc_array [, int $options])

pg_delete() は、assoc_array で指定したキーと値を用いてテーブルからレコードを削除します。 options が指定された場合、指定したオプションを つけて pg_convert() が assoc_array に適用されます。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

行を削除するテーブルの名前。

assoc_array

そのキーが table_name テーブルのフィールド名、 値が削除したいフィールドの値となっている配列。

options

PGSQL_CONV_FORCE_NULL、 ***PGSQL_DML_NO_CONV***、 ***PGSQL_DML_EXEC*** *あるいは* ***PGSQL_DML_STRING*** *を組み合わせた数。 options に* ***PGSQL_DML_STRING*** *が含まれている場合、クエリ文字列が返されます。*

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。 options で* ***PGSQL_DML_STRING*** *が渡された場合は文字列を返します。*

例

Example#1 pg_delete() の例

```
<?php
  $db = pg_connect('dbname=foo');
  // これは安全です。なぜなら $_POST は自動的に変換されるからです
  $res = pg_delete($db, 'post_log', $_POST);
  if ($res) {
      echo "POST data is deleted: $res¥n";
  } else {
      echo "User must have sent wrong inputs¥n";
  }
?>
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- *pg_convert()*

---

pg_end_copy

(PHP 4 >= 4.0.3, PHP 5)

pg_end_copy — PostgreSQL バックエンドと同期する

説明

bool ***pg_end_copy*** *([resource $connection])*

pg_end_copy() *は、pg_put_line() によるコピー操作の後で PostgreSQL フロントエンド（通常は Web サーバ プロセス）と PostgreSQL サーバを同期させます。* ***pg_end_copy()*** *を実行しなければ、PostgreSQL サーバが フロントエンドとの同期を失ってしまい、エラーが発生します。*

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

例

Example#1 pg_end_copy() の例

```
<?php
  $conn = pg_pconnect("dbname=foo");
  pg_query($conn, "create table bar (a int4, b char(16), d float8)");
  pg_query($conn, "copy bar from stdin");
  pg_put_line($conn, "3¥thello world¥t4.5¥n");
  pg_put_line($conn, "4¥tgoodbye world¥t7.11¥n");
  pg_put_line($conn, "¥¥.¥n");
  pg_end_copy($conn);
?>
```

参考

- *pg_put_line()*

---

pg_escape_bytea

(PHP 4 >= 4.2.0, PHP 5)

pg_escape_bytea — bytea フィールドに挿入するために文字列をエスケープする

説明

string **pg_escape_bytea** ([resource $connection], string $data)

pg_escape_bytea() は、bytea 型のために文字列を エスケープし、エスケープした文字列を返します。

> **注意**: bytea 型を SELECT した場合、PostgreSQL は '\' で 始まる 8 進数のバイト値（例: \032）を返します。これをユーザが手動で コンバートすることを期待されています。
> この関数は PostgreSQL 7.2 以降のバージョンを必要とします。PostgreSQL 7.2.0 および 7.2.1 では、マルチバイトのサポートを有効にした場合は bytea の値をキャストする必要があります。例: INSERT INTO test_table (image) VALUES ('$image_escaped'::bytea); PostgreSQL 7.2.2 以降ではキャストする必要はありません。 クライアントとバックエンドの文字エンコーディングが一致しない場合は 例外で、この場合はマルチバイトストリームエラーが発生します。この エラーを避けるためには bytea 型へのキャストが必要になります。

パラメータ

connection

PostgreSQL データベース接続リソース。 connection が存在しない場合は、 デフォルトの接続を使用します。デフォルトの接続は、pg_connect() あるいは pg_pconnect() で直近に作成されたものとなります。

data

bytea 型のカラムに挿入するテキストまたはバイナリデータを含む 文字列。

返り値

エスケープされたデータを文字列で返します。

変更履歴

バージョン	説明
5.2.0	connection が追加されました。

例

Example#1 pg_escape_bytea() の例

```php
<?php
  // データベースに接続する
  $dbconn = pg_connect('dbname=foo');

  // バイナリファイルを読み込む
  $data = file_get_contents('image1.jpg');

  // バイナリデータをエスケープする
  $escaped = pg_escape_bytea($data);

  // それをデータベースに挿入する
  pg_query("INSERT INTO gallery (name, data) VALUES ('Pine trees', '{$escaped}')");
?>
```

参考

- pg_unescape_bytea()
- pg_escape_string()

---

pg_escape_string

(PHP 4 >= 4.2.0, PHP 5)

pg_escape_string — テキスト型フィールドに挿入するために、文字列をエスケープする

説明

string **pg_escape_string** ([resource $connection], string $data)

pg_escape_string() は、データベースに挿入するための 文字列をエスケープします。PostgreSQL フォーマットにエスケープされた 文字列を返します。addslashes() の代わりにこの関数を 使用することを推奨します。カラム型が bytea の場合は、代わりに pg_escape_bytea() を使用しなければなりません。

> **注意**: この関数は、PostgreSQL 7.2 以降が必要です。

パラメータ

connection

PostgreSQL データベース接続リソース。 connection が存在しない場合は、 デフォルトの接続を使用します。デフォルトの接続は、pg_connect() あるいは pg_pconnect() で直近に作成されたものとなります。

data

エスケープするテキスト文字列。

返り値

エスケープされたデータを文字列で返します。

変更履歴

バージョン	説明
5.2.0	connection が追加されました。

例

Example#1 pg_escape_string() の例

```
<?php
  // データベースに接続する
  $dbconn = pg_connect('dbname=foo');

  // テキストファイルを読み込む（アポストロフィやスラッシュが含まれている）
  $data = file_get_contents('letter.txt');

  // テキストデータをエスケープする
  $escaped = pg_escape_string($data);

  // それをデータベースに挿入する
  pg_query("INSERT INTO correspondence (name, data) VALUES ('My letter', '{$escaped}')");
?>
```

参考

- pg_escape_bytea()

---

pg_execute

(PHP 5 >= 5.1.0)

pg_execute — 指定したパラメータを用いてプリペアドステートメントを実行するリクエストを 送信し、その結果を待つ

説明

```
resource pg_execute ( resource $connection , string $stmtname , array $params )
resource pg_execute ( string $stmtname , array $params )
```

指定したパラメータを用いてプリペアドステートメントを実行するリクエストを 送信し、その結果を待ちます。

pg_execute() は pg_query_params() と似ています。しかし、実行するコマンドはクエリを指定することで 決まるのではなく、事前に準備されたステートメントに値を指定する ことによって決まります。この機能のおかげで、繰り返し使用される コマンドに構文解析や実行計画作成が最初の一度だけで済みます。 実行するステートメントは、カレントのセッションで事前に準備しておく 必要があります。**pg_execute()** は、PostgreSQL 7.4 以降のバージョンの接続にのみ対応しています。それ以前のバージョンでは 失敗します。

パラメータは pg_query_params() と同じですが、 クエリ文字列のかわりにプリペアドステートメントの名前を指定する という点だけが違います。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

stmtname

実行するプリペアドステートメントの名前。"" が指定された場合は 無名ステートメントが実行されます。名前は、事前に pg_prepare() ・ pg_send_prepare() あるいは PREPARE SQL コマンドで準備されたものである 必要があります。

params

プリペアドステートメント中の $1、$2 などのプレースホルダを 置き換えるパラメータの配列。配列の要素数はプレースホルダの 数と一致する必要があります。

警告

この関数をコールする際に、要素の内容は文字列に変換されます。

返り値

成功した場合にクエリ結果リソース、失敗した場合に **FALSE** を返します。

例

Example#1 pg_execute() の使用法

```
<?php
// "mary"という名前のデータベースに接続
$dbconn = pg_connect("dbname=mary");

// 実行するクエリを準備
$result = pg_prepare($dbconn, "my_query", 'SELECT * FROM shops WHERE name = $1');

// プリペアドクエリの実行。文字列 "Joe's Widgets" のエスケープ処理は
// 一切必要ないことに注意
$result = pg_execute($dbconn, "my_query", array("Joe's Widgets"));
```

```
// 同じプリペアドクエリを違うパラメータで実行
$result = pg_execute($dbconn, "my_query", array("Clothes  Clothes  Clothes"));

?>
```

参考

- pg_prepare()
- pg_send_prepare()
- pg_query_params()

---

pg_fetch_all_columns

(PHP 5 >= 5.1.0)

pg_fetch_all_columns — 指定したカラムの全ての行を配列として取得する

説明

array **pg_fetch_all_columns** (resource $result [, int $column])

pg_fetch_all_columns() は、結果リソースの 指定したカラムについて全ての行（レコード）を含む配列を返します。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

column

結果リソースから取得するカラムの番号。0 から始まります。 指定しない場合のデフォルトは最初のカラムです。

返り値

結果カラムの全ての値を配列で返します。

column が結果のカラム数より大きい場合や その他のエラーが発生した場合に **FALSE** を返します。

例

Example#1 pg_fetch_all_columns() の例

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

$result = pg_query($conn, "SELECT title, name, address FROM authors");
if (!$result) {
   echo "An error occured.¥n";
   exit;
}

// 全ての著者名を配列で得る
$arr = pg_fetch_all_columns($result, 1);

var_dump($arr);

?>
```

参考

- pg_fetch_all()

---

pg_fetch_all

(PHP 4 >= 4.3.0, PHP 5)

pg_fetch_all — 取得されたすべての行を配列として取得する

説明

array **pg_fetch_all** (resource $result)

pg_fetch_all()は、結果リソースのすべての行 （レコード）を保持する配列を返します。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

返り値

結果のすべての行を含む配列を返します。それぞれの行は、フィールド名を 添字とするフィールド値の配列です。

結果の行数が 0 だった場合、あるいはその他のエラーが発生した場合に FALSE を返します。

例

Example#1 PostgreSQL fetch all

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

$result = pg_query($conn, "SELECT * FROM authors");
if (!$result) {
   echo "An error occured.¥n";
   exit;
}

$arr = pg_fetch_all($result);

var_dump($arr);

?>
```

参考

- pg_fetch_row()
- pg_fetch_array()
- pg_fetch_object()
- pg_fetch_result()

---

pg_fetch_array

(PHP 4, PHP 5)

pg_fetch_array — 行を配列として取得する

説明

*array **pg_fetch_array** (resource $result [, int $row [, int $result_type]])*

***pg_fetch_array()** は、取得した行（レコード）を 配列で返します。*

***pg_fetch_array()** は拡張版の pg_fetch_row() です。結果配列のフィールド番号に 対応する要素にデータを格納し、それに加えてフィールド名をキーとした 連想配列にも格納します。デフォルトで、両方ともが有効になっています。*

> ***注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。*

***pg_fetch_array()** は、 pg_fetch_row() に比べてきわめて遅いというわけでは 「ありません」。そして、きわめて簡単に使用できます。*

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

row

取得する行番号。最初の行は 0 です。指定されなかった場合、 次の行が取得されます。

result_type

*result_type は、返り値の形式を制御する オプションのパラメータです。result_type は定数であり、次の値のどれかとすることが可能です。**PGSQL_ASSOC**、**PGSQL_NUM** および **PGSQL_BOTH**。 **PGSQL_NUM** を使用すると、**pg_fetch_array()** は数値添字の配列を返します。また、**PGSQL_ASSOC** を使用すると連想配列形式で返します。**PGSQL_BOTH** がデフォルト設定で、これは数値添字の配列と連想配列の両方を返します。*

返り値

*0 から始まる数値添字の配列か連想配列（フィールド名をキーとする）、 あるいはその両方を返します。配列の各要素の値は文字列です。 データベースの NULL 値は、**NULL** として返します。*

*row が結果の行数より大きい場合や行が存在しない場合、 そしてそれ以外のエラーが発生した場合は **FALSE** を返します。*

変更履歴

バージョン	説明
4.1.0	row パラメータがオプションとなりました。

バージョン	説明
4.0.0	*result_type* パラメータが追加されました。

例

Example#1 pg_fetch_array() の例

```
<?php

$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

$result = pg_query($conn, "SELECT author, email FROM authors");
if (!$result) {
   echo "An error occured.¥n";
   exit;
}

$arr = pg_fetch_array($result, 0, PGSQL_NUM);
echo $arr[0] . " <- Row 1 Author¥n";
echo $arr[1] . " <- Row 1 E-mail¥n";

// PHP 4.1.0 以降、row パラメータはオプションです。result_type を指定
// したい場合は NULL を渡しておきます。pg_fetch_array を続けてコール
// すると、次の行を取得します。
$arr = pg_fetch_array($result, NULL, PGSQL_ASSOC);
echo $arr["author"] . " <- Row 2 Author¥n";
echo $arr["email"] . " <- Row 2 E-mail¥n";

$arr = pg_fetch_array($result);
echo $arr["author"] . " <- Row 3 Author¥n";
echo $arr[1] . " <- Row 3 E-mail¥n";

?>
```

参考

- pg_fetch_row()
- pg_fetch_object()
- pg_fetch_result()

---

pg_fetch_assoc

(PHP 4 >= 4.3.0, PHP 5)

pg_fetch_assoc — 行を連想配列として取得する

説明

array **pg_fetch_assoc** (resource $result [, int $row])

pg_fetch_assoc() は、取得した行（レコード）を 保持する連想配列を返します。

pg_fetch_assoc() は、オプションの第 3 パラメータに **PGSQL_ASSOC** を指定して pg_fetch_array() をコールするのと同じです。連想配列のみを返します。もし数値添字の配列が 必要な場合は pg_fetch_row() を使用してください。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

pg_fetch_assoc() は、 pg_fetch_row() に比べてきわめて遅いというわけでは 「ありません」。そして、きわめて簡単に使用できます。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

row

取得する行番号。最初の行は 0 です。指定されなかった場合、 次の行が取得されます。

返り値

連想配列（フィールド名をキーとする）を返します。 配列の各要素の値は文字列です。 データベースの NULL 値は、**NULL** として返します。

row が結果の行数より大きい場合、行が存在しない場合、 そしてそれ以外のエラーが発生した場合は **FALSE** を返します。

変更履歴

バージョン	説明
4.1.0	*row* パラメータがオプションとなりました。

例

Example#1 pg_fetch_assoc() の例

```
<?php
$conn = pg_connect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

$result = pg_query($conn, "SELECT id, author, email FROM authors");
if (!$result) {
   echo "An error occured.¥n";
   exit;
}

while ($row = pg_fetch_assoc($result)) {
   echo $row['id'];
   echo $row['author'];
   echo $row['email'];
}
?>
```

参考

- pg_fetch_row()
- pg_fetch_array()
- pg_fetch_object()
- pg_fetch_result()

---

pg_fetch_object

(PHP 4, PHP 5)

pg_fetch_object — 行をオブジェクトとして得る

説明

```
object pg_fetch_object ( resource $result [, int $row [, int $result_type ]] )
object pg_fetch_object ( resource $result [, int $row [, string $class_name [, array $params ]]] )
```

pg_fetch_object() は、取得した行のフィールド名に対応するプロパティを有するオブジェクトを返します。オプションとして、 指定したクラスのコンストラクタにパラメータを渡してインスタンス化する ことも可能です。

> **注意**: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

速度面では、この関数は pg_fetch_array() と同じであり、 pg_fetch_row() とほとんど同じ程度です （違いはわずかです）。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

row

取得する行番号。最初の行は 0 です。指定されなかった場合、 次の行が取得されます。

result_type

非推奨で、無視されます。デフォルトは **PGSQL_ASSOC** です。

class_name

インスタンス化し、プロパティを設定して返り値とするクラスの名前。 指定しない場合は stdClass オブジェクトが返されます。

params

class_name オブジェクトのコンストラクタに 渡すオプションの配列。

返り値

結果の各フィールドに対応する属性を持つ object を返します。 データベースの NULL 値は **NULL** として返します。

row が結果の行数より大きい場合・行が存在しない場合 、そしてそれ以外のエラーが発生した場合は **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	*class_name* と *params* が追加されました。*result_type* を用いた古い形式は、過去のバージョンとの互換性を保つために残されています。
4.3.0	*result_type* のデフォルト値が、 **PGSQL_BOTH** から **PGSQL_ASSOC** に変わりました。数値インデックスが使用できないためです。
4.1.0	*row* パラメータがオプションとなりました。

例

Example#1 pg_fetch_object() の例

```php
<?php

$database = "store";

$db_conn = pg_connect("host=localhost port=5432 dbname=$database");
if (!$db_conn) {
  echo "Failed connecting to postgres database $database¥n";
  exit;
}

$qu = pg_query($db_conn, "SELECT * FROM books ORDER BY author");


while ($data = pg_fetch_object($qu)) {
  echo $data->author . " (";
  echo $data->year . "): ";
  echo $data->title . "<br />";
}

pg_free_result($qu);
pg_close($db_conn);

?>
```

参考

- pg_query()
- pg_fetch_array()
- pg_fetch_assoc()
- pg_fetch_row()
- pg_fetch_result()

---

pg_fetch_result

(PHP 4 >= 4.2.0, PHP 5)

pg_fetch_result — 結果リソースから値を返す

説明

```
string pg_fetch_result ( resource $result , int $row , mixed $field )
string pg_fetch_result ( resource $result , mixed $field )
```

pg_fetch_result() は、PostgreSQL 結果リソースから 特定の行とフィールド（カラム）の値を返します。

> **注意**: この関数は、以前は **pg_result()** という名前でした。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

row

結果から取得する行の番号。行番号は 0 から始まります。指定しなかった 場合は、次の行が読み込まれます。

field

取得するフィールド（カラム）の名前を表す文字列、あるいは取得する フィールドの番号。フィールド番号は 0 から始まります。

返り値

論理型の値は "t" あるいは "f" の形式で返します。 配列を含むそれ以外の型は、PostgreSQL のやりかたにしたがって文字列として フォーマットされた形式で返します。これは **psql** プログラムの出力と同じ形式です。データベースの NULL 値は、**NULL** として返します。

row が結果の行数より大きい場合、 あるいはそれ以外のエラーが発生した場合は **FALSE** を返します。

例

Example#1 pg_fetch_result() の例

```php
<?php
$db = pg_connect("dbname=users user=me") || die();

$res = pg_query($db, "SELECT 1 UNION ALL SELECT 2");

$val = pg_fetch_result($res, 1, 0);

echo "First field in the second row is: ", $val, "¥n";
?>
```

上の例の出力は以下となります。

```
First field in the second row is: 2
```

参考

- pg_query()
- pg_fetch_array()

---

pg_fetch_row

(PHP 4, PHP 5)

pg_fetch_row — 数値添字の配列として行を得る

説明

array **pg_fetch_row** (resource $result [, int $row])

pg_fetch_row() は、指定した result リソースが指す結果から 1 行分のデータを取得します。

注意: この関数は、 NULL フィールドに PHPの **NULL** 値を設定します。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

row

結果から取得する行の番号。行番号は 0 から始まります。指定しなかった 場合は、次の行が読み込まれます。

返り値

個々の値を文字列として格納した配列を返します。配列の添字は 0 から 始まります。データベースの NULL 値は **NULL** として返します。

row が結果の行数より大きい場合・行が存在しない場合 、そしてそれ以外のエラーが発生した場合は **FALSE** を返します。

変更履歴

バージョン	説明
4.1.0	row パラメータがオプションとなりました。

例

Example#1 pg_fetch_row() の例

```
<?php

$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

$result = pg_query($conn, "SELECT author, email FROM authors");
if (!$result) {
   echo "An error occured.¥n";
   exit;
}

while ($row = pg_fetch_row($result)) {
   echo "Author: $row[0]  E-mail: $row[1]";
   echo "<br />¥n";
}

?>
```

参考

- pg_query()
- pg_fetch_array()
- pg_fetch_object()
- pg_fetch_result()

---

pg_field_is_null

(PHP 4 >= 4.2.0, PHP 5)

pg_field_is_null — フィールドが SQL の NULL かどうか調べる

説明

```
int pg_field_is_null ( resource $result , int $row , mixed $field )
int pg_field_is_null ( resource $result , mixed $field )
```

pg_field_is_null() は、PostgreSQL 結果リソースの フィールドが SQL の NULL であるかどうかを調べます。

注意: この関数は、以前は **pg_fieldisnull()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

row

取得する結果の行番号。行番号は 0 から始まります。指定しなかった場合は カレントの行を取得します。

field

(0 から始まる) フィールド番号を表す数値、 あるいはフィールド名を表す文字列。

返り値

指定した行のフィールドが SQL の NULL だった場合に 1、そうでない場合に 0 を返します。 もし範囲外の行を指定したりその他のエラーが発生したりした場合には **FALSE** を返します。

例

Example#1 pg_field_is_null() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die ("Could not connect");
  $res = pg_query($dbconn, "select * from authors where author = 'Orwell'");
  if ($res) {
      if (pg_field_is_null($res, 0, "year") == 1) {
          echo "The value of the field year is null.¥n";
      }
      if (pg_field_is_null($res, 0, "year") == 0) {
          echo "The value of the field year is not null.¥n";
    }
 }
?>
```

---

pg_field_name

(PHP 4 >= 4.2.0, PHP 5)

pg_field_name — フィールドの名前を返す

説明

string **pg_field_name** (resource $result , int $field_number)

pg_field_name() は、指定した PostgreSQL result リソースの指定した field_number にあるフィールドの名前を 返します。フィールド番号は 0 から始まります。

注意: この関数は、以前は **pg_fieldname()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

field_number

フィールド番号。0 から始まります。

返り値

フィールド名を返します。エラー時には **FALSE** を返します。

例

Example#1 フィールド情報を取得する

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  $res = pg_query($dbconn, "select * from authors where author = 'Orwell'");
  $i = pg_num_fields($res);
  for ($j = 0; $j < $i; $j++) {
      echo "column $j¥n";
      $fieldname = pg_field_name($res, $j);
      echo "fieldname: $fieldname¥n";
      echo "printed length: " . pg_field_prtlen($res, $fieldname) . " characters¥n";
      echo "storage length: " . pg_field_size($res, $j) . " bytes¥n";
      echo "field type: " . pg_field_type($res, $j) . " ¥n¥n";
  }
?>
```

上の例の出力は以下となります。

```
column 0
fieldname: author
printed length: 6 characters
```

```
storage length: -1 bytes
field type: varchar

column 1
fieldname: year
printed length: 4 characters
storage length: 2 bytes
field type: int2

column 2
fieldname: title
printed length: 24 characters
storage length: -1 bytes
field type: varchar
```

参考

- pg_field_num()

---

pg_field_num

(PHP 4 >= 4.2.0, PHP 5)

pg_field_num — 指定されたフィールドのフィールド番号を返す

説明

```
int pg_field_num ( resource $result , string $field_name )
```

pg_field_num() は、指定した PostgreSQL の 結果リソース(result)において field_name に相当するカラム(フィールド) のフィールド番号を返します。

> **注意**: この関数は、以前は **pg_fieldnum()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

field_name

フィールドの名前。

返り値

フィールド番号（0 から始まります）を返します。エラー時には -1 を返します。

例

Example#1 フィールドの情報を取得する

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  $res = pg_query($dbconn, "select author, year, title from authors where author = 'Orwell'");

   echo "Column 'title' is field number: ", pg_field_num($res, 'title');
?>
```

上の例の出力は以下となります。

```
Column 'title' is field number: 2
```

参考

- pg_field_name()

---

pg_field_prtlen

(PHP 4 >= 4.2.0, PHP 5)

pg_field_prtlen — 表示される長さを返す

説明

```
int pg_field_prtlen ( resource $result , int $row_number , mixed $field_name_or_number )
int pg_field_prtlen ( resource $result , mixed $field_name_or_number )
```

pg_field_prtlen() は、PostgreSQL の result の指定した値に関して、実際に 表示した場合の長さ(文字の数)を返します。行番号

(row_number)は 0 から始まります。この関数は、エラーの場合に-1を返します。

field_name_or_number は、integer または string のどちらかで渡すことが可能です。 integer で渡された場合、PHP はそれをフィールド番号と判断 します。それ以外の場合はフィールド名と判断します。

pg_field_name() ページの例を参照ください。

注意: この関数は、以前は **pg_fieldprtlen()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

row

結果の行番号。行番号は 0 から始まります。指定しなかった場合は カレントの行を取得します。

返り値

フィールドの表示される長さを返します。エラー時には **FALSE** を返します。

例

Example#1 フィールドの情報を取得する

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  $res = pg_query($dbconn, "select * from authors where author = 'Orwell'");
  $i = pg_num_fields($res);
  for ($j = 0; $j < $i; $j++) {
       echo "column $j¥n";
       $fieldname = pg_field_name($res, $j);
       echo "fieldname: $fieldname¥n";
       echo "printed length: " . pg_field_prtlen($res, $fieldname) . " characters¥n";
       echo "storage length: " . pg_field_size($res, $j) . " bytes¥n";
       echo "field type: " . pg_field_type($res, $j) . " ¥n¥n";
  }
?>
```

上の例の出力は以下となります。

```
column 0
fieldname: author
printed length: 6 characters
storage length: -1 bytes
field type: varchar

column 1
fieldname: year
printed length: 4 characters
storage length: 2 bytes
field type: int2

column 2
fieldname: title
printed length: 24 characters
storage length: -1 bytes
field type: varchar
```

参考

- pg_field_size()

---

pg_field_size

(PHP 4 >= 4.2.0, PHP 5)

pg_field_size — 指定したフィールドの内部記憶領域におけるサイズを返す

説明

int **pg_field_size** (resource $result , int $field_number)

pg_field_size() は、指定した PostgreSQL の 結果において、指定したフィールド番号の内部記憶領域のサイズを (バイト数で)返します。

注意: この関数は、以前は **pg_fieldsize()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

field_number

フィールド番号。0 から始まります。

返り値

内部記憶領域のサイズ（バイト数）を返します。-1 は可変長フィールドを 示します。エラー時には **FALSE** を返します。

例

Example#1 フィールドの情報を取得する

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  $res = pg_query($dbconn, "select * from authors where author = 'Orwell'");
  $i = pg_num_fields($res);
  for ($j = 0; $j < $i; $j++) {
       echo "column $j¥n";
       $fieldname = pg_field_name($res, $j);
       echo "fieldname: $fieldname¥n";
       echo "printed length: " . pg_field_prtlen($res, $fieldname) . " characters¥n";
       echo "storage length: " . pg_field_size($res, $j) . " bytes¥n";
       echo "field type: " . pg_field_type($res, $j) . " ¥n¥n";
  }
?>
```

上の例の出力は以下となります。

```
column 0
fieldname: author
printed length: 6 characters
storage length: -1 bytes
field type: varchar

column 1
fieldname: year
printed length: 4 characters
storage length: 2 bytes
field type: int2

column 2
fieldname: title
printed length: 24 characters
storage length: -1 bytes
field type: varchar
```

参考

- pg_field_prtlen()
- pg_field_type()

---

pg_field_table

(PHP 5 >= 5.2.0)

pg_field_table — tables フィールドの名前あるいは oid を返す

説明

mixed **pg_field_table** (resource $result , int $field_number [, bool $oid_only])

pg_field_table() は、フィールドが属するテーブルの名前か あるいは oid_only が **TRUE** の場合にテーブルの oid を返します。

パラメータ

result

pg_query()、pg_query_params() あるいは pg_execute()（その他も含む）から返された、PostgreSQL クエリ結果リソース。

field_number

0 から始まるフィールド番号。

oid_only

デフォルトでは、フィールドが属するテーブルの名前が返されます。 しかし oid_only を**TRUE** に設定すると、 その代わりに oid が返されます。

返り値

成功した場合にフィールドのテーブル名あるいは oid、 あるいは失敗した場合に **FALSE** を返します。

例

Example#1 フィールドについてのテーブル情報の取得

```
<?php
$dbconn = pg_connect("dbname=publisher") or die("接続できません");
```

```
$res = pg_query($dbconn, "SELECT bar FROM foo");

echo pg_field_table($res, 0);
echo pg_field_table($res, 0, true);

$res = pg_query($dbconn, "SELECT version()");
var_dump(pg_field_table($res, 0));
?>
```

上の例の出力は、たとえば以下のようになります。

```
foo
14379580

bool(false)
```

注意

> **注意**: テーブル名を返すよりも oid を返すほうがずっと高速です。 なぜなら、テーブル名を取得するには、 データベースのシステムテーブルにクエリを発行しなければならないからです。

参考

- pg_field_name()
- pg_field_type()

---

pg_field_type_oid

(PHP 5 >= 5.1.0)

pg_field_type_oid — フィールド番号に対応する型 ID (OID) を返す

説明

int **pg_field_type_oid** (resource $result , int $field_number)

pg_field_type_oid() は、指定した PostgreSQL result リソースにおける指定した field_number の型の OID を返します。

フィールド型についての詳細な情報を得るには、PostgreSQL のシステムテーブル pg_type に対して、この関数で取得した OID を用いて 問い合わせます。PostgreSQL の **format_type()** 関数は、 型の OID を SQL の型名に変換します。

> **注意**: フィールドが（基本型ではなく）PostgreSQL ドメインを使用している場合は、 ドメインそのものの OID ではなくドメインの元となっている型の OID を返します。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

field_number

フィールド番号。0 から始まります。

返り値

フィールドの型に対応する OID を返します。エラー時には **FALSE** を返します。

例

Example#1 フィールドの情報を得る

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  // 'title' は varchar 型であると仮定する
  $res = pg_query($dbconn, "select title from authors where author = 'Orwell'");

   echo "Title field type OID: ", pg_field_type_oid($res, 0);
?>
```

上の例の出力は以下となります。

```
Title field type OID: 1043
```

参考

- pg_field_type()
- pg_field_prtlen()
- pg_field_name()

---

pg_field_type

(PHP 4 >= 4.2.0, PHP 5)

pg_field_type — フィールド番号に対応する型名を返す

説明

string **pg_field_type** (resource $result , int $field_number)

pg_field_type() は、指定した PostgreSQL の result リソースにおいて、指定した field_number の型名を保持する文字列を 返します。

> **注意**: フィールドが（基本型ではなく）PostgreSQL ドメインを使用している場合は、 ドメインそのものの名前ではなくドメインの元となっている型の名前を返します。
>
> **注意**: この関数は、以前は **pg_fieldtype()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

field_number

フィールド番号。0 から始まります。

返り値

フィールド型の名前を文字列で返します。エラー時には **FALSE** を返します。

例

Example#1 フィールドの情報を取得する

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  // 'title' は varchar 型と仮定する
  $res = pg_query($dbconn, "select title from authors where author = 'Orwell'");

   echo "Title field type: ", pg_field_type($res, 0);
?>
```

上の例の出力は以下となります。

```
Title field type: varchar
```

参考

- pg_field_prtlen()
- pg_field_name()
- pg_field_type_oid()

---

pg_free_result

(PHP 4 >= 4.2.0, PHP 5)

pg_free_result — メモリを開放する

説明

bool **pg_free_result** (resource $result)

pg_free_result() は、指定された PostgreSQL クエリ 結果 resource に関するメモリとデータを開放します。

この関数は、スクリプト実行中のメモリ使用量を抑制したい場合にのみ コールする必要があります。それ以外の場合は、すべての結果保持用 メモリは、スクリプトが終了する際に自動的に開放されます。

> **注意**: この関数は、以前は **pg_freeresult()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_free_result() の例

```
<?php
$db = pg_connect("dbname=users user=me") || die();

$res = pg_query($db, "SELECT 1 UNION ALL SELECT 2");

$val = pg_fetch_result($res, 1, 0);

echo "First field in the second row is: ", $val, "¥n";

pg_free_result($res);
?>
```

上の例の出力は以下となります。

```
First field in the second row is: 2
```

参考

- pg_query()
- pg_query_params()
- pg_execute()

---

pg_get_notify

(PHP 4 >= 4.3.0, PHP 5)

pg_get_notify — SQL NOTIFY メッセージを取得する

説明

array **pg_get_notify** (resource $connection [, int $result_type])

pg_get_notify()は、NOTIFY SQL コマンドにより送信された通知メッセージを取得します。 通知メッセージを取得するには、LISTEN SQL コマンドを発行する必要があります。

パラメータ

connection

PostgreSQL データベースの接続リソース。

result_type

result_type は、返り値の形式を制御する オプションのパラメータです。result_type は定数であり、次の値のどれかとすることが可能です。**PGSQL_ASSOC**、**PGSQL_NUM** および **PGSQL_BOTH**。 **PGSQL_NUM** を使用すると、**pg_get_notify()** は数値添字の配列を返します。また、**PGSQL_ASSOC** を使用すると連想配列形式で返します。**PGSQL_BOTH** がデフォルト設定で、これは数値添字の配列と連想配列の両方を返します。

返り値

NOTIFY メッセージ名とバックエンドの PID を含む 配列を返します。もし待ち受ける NOTIFY が存在しない 場合は then **FALSE** を返します。

例

Example#1 PostgreSQL NOTIFY メッセージ

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
  echo "An error occured.¥n";
  exit;
}

// 他のプロセスからの 'author_updated' メッセージを監視する
pg_query($conn, 'LISTEN author_updated;');
$notify = pg_get_notify($conn);
if (!$notify) {
  echo "No messages¥n";
} else {
  print_r($notify);
}
?>
```

参考

- pg_get_pid()

---

pg_get_pid

(PHP 4 >= 4.3.0, PHP 5)

pg_get_pid — バックエンドのプロセス ID を得る

説明

```
int pg_get_pid ( resource $connection )
```

pg_get_pid() は、バックエンド(データベースサーバ のプロセス)のプロセス ID を取得します。プロセス ID は、 NOTIFY メッセージが他のプロセスから送信されている かどうかを調べるために有用です。

パラメータ

connection

PostgreSQL データベースの接続リソース。

返り値

バックエンドのデータベースのプロセス ID 。

例

Example#1 PostgreSQL バックエンドの PID

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

// バックエンドのプロセスIDを取得。このプロセスIDをpg_get_notify()で使用
$pid = pg_get_pid($conn);
?>
```

参考

- pg_get_notify()

---

pg_get_result

(PHP 4 >= 4.2.0, PHP 5)

pg_get_result — 非同期クエリの結果を取得する

説明

```
resource pg_get_result ([ resource $connection ] )
```

pg_get_result() は、 pg_send_query() ・pg_send_query_params() あるいは pg_send_execute() で実行した非同期クエリから結果リソースを取得します。

pg_send_query() およびその他の非同期クエリ関数は、 複数のクエリを PostgreSQL サーバに送信することが可能です。クエリの結果を ひとつずつ取得するには、**pg_get_result()** を使用します。

パラメータ

connection

PostgreSQL データベースの接続リソース。

返り値

結果 resource を返します。結果がもうない場合に **FALSE** を返します。

例

Example#1 pg_get_result() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

   if (!pg_connection_busy($dbconn)) {
       pg_send_query($dbconn, "select * from authors; select count(*) from authors;");
   }

   $res1 = pg_get_result($dbconn);
   echo "First call to pg_get_result(): $res1¥n";
   $rows1 = pg_num_rows($res1);
   echo "$res1 has $rows1 records¥n¥n";

   $res2 = pg_get_result($dbconn);
   echo "Second call to pg_get_result(): $res2¥n";
   $rows2 = pg_num_rows($res2);
   echo "$res2 has $rows2 records¥n";
?>
```

上の例の出力は以下となります。

```
First call to pg_get_result(): Resource id #3
Resource id #3 has 3 records

Second call to pg_get_result(): Resource id #4
```

```
Resource id #4 has 1 records
```

参考

- pg_send_query()

---

pg_host

(PHP 4, PHP 5)

pg_host — 接続に関連するホスト名を返す

説明

string **pg_host** ([resource $connection])

pg_host() は、指定した PostgreSQL connection リソースが接続しているホストの 名前を返します。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

connection が接続しているホストの名前を 文字列で返します。エラー時には **FALSE** を返します。

例

Example#1 pg_host() の例

```
<?php
$pgsql_conn = pg_connect("dbname=mark host=localhost");

if ($pgsql_conn) {
    print "Successfully connected to: " . pg_host($pgsql_conn) . "<br/>¥n";
} else {
    print pg_last_error($pgsql_conn);
    exit;
}
?>
```

参考

- pg_connect()
- pg_pconnect()

---

pg_insert

(PHP 4 >= 4.3.0, PHP 5)

pg_insert — テーブルに配列を挿入する

説明

mixed **pg_insert** (resource $connection , string $table_name , array $assoc_array [, int $options])

pg_insert() は、 table_name で指定したテーブルに assoc_array の値をを挿入します。 options が指定されている場合、そのオプションとともに pg_convert() が assoc_array に適用されます。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

行を挿入するテーブルの名前。テーブル table_name は、少なくとも assoc_array の要素数ぶんのカラムを 保持している必要があります。

assoc_array

テーブル table_name のフィールド名をキーに、 挿入する内容を値にもつ配列。

options

PGSQL_CONV_OPTS、 **PGSQL_DML_NO_CONV**、 **PGSQL_DML_EXEC**、 **PGSQL_DML_ASYNC** あるいは **PGSQL_DML_STRING** を組み合わせた数。 **PGSQL_DML_STRING** が options に含まれていた場合、 クエリ文字列が返されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 options で **PGSQL_DML_STRING** が渡された場合は文字列を返します。

例

Example#1 pg_insert() の例

```
<?php
   $dbconn = pg_connect('dbname=foo');
  // これは安全です。なぜなら $_POST は自動的に変換されるからです。
  $res = pg_insert($dbconn, 'post_log', $_POST);
  if ($res) {
      echo "POST data is successfully logged¥n";
  } else {
      echo "User must have sent wrong inputs¥n";
  }
?>
```

参考

- pg_convert()

---

pg_last_error

(PHP 4 >= 4.2.0, PHP 5)

pg_last_error — 特定の接続から直近のエラーメッセージ文字列を取得する

説明

string **pg_last_error** ([resource $connection])

pg_last_error() は、指定した connection から直近のエラーメッセージを返します。

エラーメッセージは、内部的な PostgreSQL(libpq) 関数コールにより 上書きされる可能性があります。PostgreSQL モジュール関数の中で複数のエラーが発生した場合には、この関数は適切なエラーメッセージを返さない 可能性があります。

エラー処理を改善するために pg_result_error()、pg_result_error_field()、 pg_result_status() および pg_connection_status() を使用ください。

> **注意**: この関数は、以前は **pg_errormessage()** と呼ばれていました。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

指定した connection の直近のエラーメッセージを 含む文字列を返します。エラー時には **FALSE** を返します。

例

Example#1 pg_last_error() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("接続できませんでした");

  // 失敗するクエリ
  $res = pg_query($dbconn, "select * from doesnotexist");

  echo pg_last_error($dbconn);
?>
```

参考

- pg_result_error()
- pg_result_error_field()

---

pg_last_notice

(PHP 4 >= 4.0.6, PHP 5)

pg_last_notice — PostgreSQL サーバからの直近の通知メッセージを返す

説明

string **pg_last_notice** (resource $connection)

pg_last_notice() は、connection で指定した PostgreSQL サーバからの直近の通知メッセージを返します。 たとえば、テーブルに SERIAL カラムを作成する場合などに PostgreSQL サーバは通知メッセージを送信します。

pg_last_notice() でトランザクションに関連する通知 メッセージがあるかないかをチェックすることで、無意味なクエリの発行を 避けることが可能です。

php.ini で pgsql.ignore_notice に 1 を指定することで、通知メッセージの追跡をしないようにすることが可能です。

php.ini で pgsql.log_notice に 0 を指定することで、通知メッセージをログに記録しないようにすることが可能です。 pgsql.ignore_notice が 0 に設定されていない限り、 通知メッセージをログに記録することはできません。

パラメータ

connection

PostgreSQL データベースの接続リソース。

返り値

指定された connection の直近の通知を文字列で 返します。エラー時には **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	この関数の実装が完成しました。 以前のバージョンではデータベースの接続パラメータを無視していました。
4.3.0	*php.ini* の設定 *pgsql.ignore_notice* および *pgsql.log_notice* が追加されました。
4.0.6	PHP 4.0.6 には通知メッセージ処理の問題があります。 pg_last_notice() を使用しない場合でも PHP 4.0.6 で PostgreSQL モジュールを使用することは推奨されません。

例

Example#1 pg_last_error() の例

```
<?php
   $pgsql_conn = pg_connect("dbname=mark  host=localhost");

  $res = pg_query("CREATE  TABLE  test  (id  SERIAL)");

  $notice  = pg_last_notice($pgsql_conn);

   echo $notice;
?>
```

上の例の出力は以下となります。

```
CREATE  TABLE  will  create  implicit  sequence  "test_id_seq"  for  "serial"  column  "test.id"
```

参考

- pg_query()
- pg_last_error()

---

pg_last_oid

(PHP 4 >= 4.2.0, PHP 5)

pg_last_oid — 直近の行のオブジェクト ID を返す

説明

string **pg_last_oid** (resource $result)

pg_last_oid() は、挿入された行に割り当てられた OID を取得します。

OID フィールドは PostgreSQL 7.2 からはオプションとなり、 PostgreSQL 8.1 ではデフォルトで存在しません。OID フィールドが テーブルに存在しない場合、プログラマは pg_result_status() を使用して挿入が成功したことを 確かめる必要があります。

挿入された行の SERIAL フィールドの値を取得するには、 PostgreSQL の CURRVAL 関数を使用してシーケンス内で 直近に要求された値を取得する必要があります。シーケンス名がわからない 場合は、PostgreSQL 8.0 の関数 pg_get_serial_sequence が必要になります。

PostgreSQL 8.1 には LASTVAL 関数が存在し、セッション内で 直近に使用されたシーケンスの値を返します。これを用いれば、シーケンスや テーブルやカラムの名前を指定する必要がなくなります。

> **注意**: この関数は、以前は **pg_getlastoid()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

返り値

指定された connection で、直近に挿入された行に 割り当てられた OID を文字列で返します。エラー時や有効な OID のない場合に **FALSE** を返します。

例

Example#1 pg_last_oid() の例

```
<?php
   $pgsql_conn = pg_connect("dbname=mark  host=localhost");

  $res1 = pg_query("CREATE  TABLE  test  (a  INTEGER)  WITH  OIDS");

  $res2 = pg_query("INSERT  INTO  test  VALUES  (1)");

  $oid = pg_last_oid($res2);
?>
```

参考

- pg_query()
- pg_result_status()

---

pg_lo_close

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_close — ラージオブジェクトをクローズする

説明

bool **pg_lo_close** (resource $large_object)

pg_lo_close() はラージオブジェクトをクローズします。 large_object は、pg_lo_open() でオープンされたラージオブジェクトのリソースです。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

注意: This function used to be called **pg_loclose()**.

パラメータ

result

PostgreSQL large object (LOB) resource, returned by pg_lo_open().

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_lo_close() example

```
<?php
    $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $oid = pg_lo_create($database);
    echo "$oid¥n";
   $handle = pg_lo_open($database, $oid, "w");
    echo "$handle¥n";
   pg_lo_write($handle, "large  object  data");
   pg_lo_close($handle);
   pg_query($database, "commit");
?>
```

参考

- pg_lo_open()
- pg_lo_create()
- pg_lo_import()

---

pg_lo_create

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_create — ラージオブジェクトを生成する

説明

int **pg_lo_create** ([resource $connection])

pg_lo_create() はラージオブジェクトを 生成し、そのラージオブジェクトの OID を返します。 PostgreSQL アクセスモード **INV_READ**、

INV_WRITE および **INV_ARCHIVE** はサポートされません。オブジェクトは 常に読み書き可のアクセス権で生成されます。 **INV_ARCHIVE** は PostgreSQL 自身からも削除されました （バージョン 6.3 以降）。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

ラージオブジェクトインターフェース（アクセス制御もできないし使用が面倒） を使用するかわりに、PostgreSQL の bytea カラム型と pg_escape_bytea() を試してください。

> **注意**: この関数は、以前は **pg_locreate()** と呼ばれていました。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

ラージオブジェクトの OID を返します。 エラー時には **FALSE** を返します。

例

Example#1 pg_lo_create() の例

```
<?php
    $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $oid = pg_lo_create($database);
   echo "$oid¥n";
   $handle = pg_lo_open($database, $oid, "w");
   echo "$handle¥n";
   pg_lo_write($handle, "large object data");
   pg_lo_close($handle);
   pg_query($database, "commit");
?>
```

---

pg_lo_export

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_export — ラージオブジェクトをファイルにエクスポートする

説明

```
bool pg_lo_export ( resource $connection , int $oid , string $pathname )
bool pg_lo_export ( int $oid , string $pathname )
```

pg_lo_export() は PostgreSQL データベースから ラージオブジェクトを取得し、その内容をローカルファイルシステム上の ファイルに保存します。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

> **注意**: この関数は、以前は **pg_loexport()** と呼ばれていました。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

oid

データベース内のラージオブジェクトの OID 。

pathname

ラージオブジェクトをクライアントのファイルシステムに書き込む際の フルパスとファイル名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_lo_export() の例

```
<?php
    $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $oid = pg_lo_create($database);
   $handle = pg_lo_open($database, $oid, "w");
   pg_lo_write($handle, "large object data");
   pg_lo_close($handle);
   pg_lo_export($database, $oid, '/tmp/lob.dat');
   pg_query($database, "commit");
?>
```

参考

- pg_lo_import()

---

pg_lo_import

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_import — ファイルからラージオブジェクトをインポートする

説明

```
int pg_lo_import ( resource $connection , string $pathname )
int pg_lo_import ( string $pathname )
```

pg_lo_import() は、ファイルシステム上のファイルの データをもとにして新しいラージオブジェクトをデータベース内に作成します。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

> **注意**: セーフモード が有効の場合、PHP は操作を行うファイル／ディレクトリが実行するスクリプトと同じ UID (所有者)を有しているかどうかを確認します。

> **注意**: この関数は、以前は **pg_loimport()** と呼ばれていました。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

pathname

クライアントのファイルシステムからラージオブジェクト用データを 読み込む際のフルパスとファイル名。

返り値

作成されたラージオブジェクトの OID を返します。 失敗した場合には **FALSE** を返します。

変更履歴

バージョン	説明
4.2.0	関数の構文が変わりました。以前は次のようなものでした。 int pg_lo_import (string $pathname [, resource $connection])

例

Example#1 pg_lo_import() の例

```
<?php
    $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $oid = pg_lo_import($database, '/tmp/lob.dat');
   pg_query($database, "commit");
?>
```

参考

- pg_lo_export()
- pg_lo_open()

---

pg_lo_open

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_open — ラージオブジェクトをオープンする

説明

```
resource pg_lo_open ( resource $connection , int $oid , string $mode )
```

pg_lo_open() はデータベース内にラージオブジェクトを オープンし、それを操作するためのラージオブジェクトリソースを返します。

警告

ラージオブジェクトのリソースを閉じる前にデータベースへの接続を 閉じないでください。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

> **注意**: この関数は、以前は **pg_loopen()** と呼ばれていました。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

oid

データベース内のラージオブジェクトの OID 。

mode

読み込み専用の "r"、書き込み専用の "w"、読み書き可能な "rw" のいずれか。

返り値

*ラージオブジェクトのリソースを返します。エラー時には **FALSE** を返します。*

例

Example#1 pg_lo_open() の例

```
<?php
    $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $oid = pg_lo_create($database);
   echo "$oid¥n";
   $handle = pg_lo_open($database, $oid, "w");
   echo "$handle¥n";
   pg_lo_write($handle, "large object data");
   pg_lo_close($handle);
   pg_query($database, "commit");
?>
```

参考

- pg_lo_close()
- pg_lo_create()

---

pg_lo_read_all

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_read_all — ラージオブジェクト全体を読み込みブラウザに直接送信する

説明

*int **pg_lo_read_all** (resource $large_object)*

***pg_lo_read_all()** は、ラージオブジェクトを読み込み 未送信のヘッダを全て送信した後、ブラウザに直接ラージオブジェクトを 送信します。これは主に、イメージや音などのバイナリデータを送信するために 使用します。*

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

> ***注意**: この関数は、以前は **pg_loreadall()** と呼ばれていました。*

パラメータ

large_object

pg_lo_open() によって返された PostgreSQL ラージオブジェクト (LOB) リソース。

返り値

*読み込んだバイト数を返します。エラー時には **FALSE** を返します。*

例

Example#1 pg_lo_read_all() の例

```
<?php
    header('Content-type: image/jpeg');
   $image_oid = 189762345;
   $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $handle = pg_lo_open($database, $image_oid, "r");
   pg_lo_read_all($handle);
   pg_query($database, "commit");
?>
```

参考

- pg_lo_read()

---

pg_lo_read

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_read — ラージオブジェクトを読み込む

説明

string **pg_lo_read** (resource $large_object [, int $len])

pg_lo_read()は、ラージオブジェクトから最大 len バイト分読み込み、文字列として返します。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

> **注意**: この関数は、以前は **pg_loread()** と呼ばれていました。

パラメータ

large_object

pg_lo_open() によって返された PostgreSQL ラージオブジェクト (LOB) リソース。

len

返すデータの最大バイト数（オプション）。デフォルトは 8192 です。

返り値

ラージオブジェクトから len バイトのデータを 文字列で返します。エラー時には **FALSE** を返します。

例

Example#1 pg_lo_read() の例

```
<?php
    $doc_oid = 189762345;
   $database  = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $handle  = pg_lo_open($database, $doc_oid, "r");
   $data  = pg_lo_read($handle, 50000);
   pg_query($database, "commit");
    echo $data;
?>
```

参考

- pg_lo_read_all()

---

pg_lo_seek

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_seek — ラージオブジェクトの位置をシークする

説明

bool **pg_lo_seek** (resource $large_object , int $offset [, int $whence])

pg_lo_seek() はラージオブジェクトリソースの位置を シークします。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

パラメータ

large_object

pg_lo_open() が返す PostgreSQL の ラージオブジェクト(LOB)リソース。

offset

シークするバイト数。

whence

定数 **PGSQL_SEEK_SET**（オブジェクトの先頭からシークする）、 **PGSQL_SEEK_CUR**（カレントの位置からシークする）、 あるいは **PGSQL_SEEK_END**（オブジェクトの最後からシークする） の中のひとつ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_lo_seek() の例

```
<?php
    $doc_oid = 189762345;
   $database  = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $handle  = pg_lo_open($database, $doc_oid, "r");
   // 最初の 50000 バイトをスキップする
   pg_lo_seek($handle, 50000, PGSQL_SEEK_SET);
   // 次の 10000 バイトを読み込む
```

```
    $data = pg_lo_read($handle, 10000);
    pg_query($database, "commit");
    echo $data;
?>
```

参考

- pg_lo_tell()

---

pg_lo_tell

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_tell — ラージオブジェクトのカレントのシーク位置を返す

説明

int **pg_lo_tell** (resource $large_object)

pg_lo_tell() は、ラージオブジェクトのカレントの位置 （先頭からのオフセット）を返します。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

パラメータ

large_object

pg_lo_open() が返す PostgreSQL の ラージオブジェクト(LOB)リソース。

返り値

ラージオブジェクトの先頭からのカレントのシーク位置（バイト数）を返します。 エラーが発生した場合は、負の数を返します。

例

Example#1 pg_lo_tell() の例

```
<?php
    $doc_oid = 189762345;
    $database = pg_connect("dbname=jacarta");
    pg_query($database, "begin");
    $handle = pg_lo_open($database, $doc_oid, "r");
    // 最初の 50000 バイトをスキップする
    pg_lo_seek($handle, 50000, PGSQL_SEEK_SET);
    // どれだけスキップしたのかを調べる
    $offset = pg_lo_tell($handle);
    echo "Seek position is: $offset";
    pg_query($database, "commit");
?>
```

上の例の出力は以下となります。

```
Seek position is: 50000
```

参考

- pg_lo_seek()

---

pg_lo_unlink

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_unlink — ラージオブジェクトを削除する

説明

bool **pg_lo_unlink** (resource $connection , int $oid)

pg_lo_unlink()は、oid で 関連付けられたラージオブジェクトを削除します。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

> **注意**: この関数は、以前は **pg_lounlink()** と呼ばれていました。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

oid

データベース内のラージオブジェクトの OID 。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_lo_unlink() の例

```
<?php
   // 削除するラージオブジェクトの OID
   $doc_oid = 189762345;
   $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   pg_lo_unlink($database, $doc_oid);
   pg_query($database, "commit");
?>
```

参考

- pg_lo_create()
- pg_lo_import()

---

pg_lo_write

(PHP 4 >= 4.2.0, PHP 5)

pg_lo_write — ラージオブジェクトを書く

説明

int **pg_lo_write** (resource $large_object , string $data [, int $len])

pg_lo_write() は、ラージオブジェクトの カレントのシーク位置にデータを書き込みます。

ラージオブジェクトインターフェースは、トランザクションブロックの中で 使用する必要があります。

> **注意**: この関数は、以前は **pg_lowrite()** と呼ばれていました。

パラメータ

large_object

pg_lo_open() によって返された PostgreSQL ラージオブジェクト (LOB) リソース。

data

ラージオブジェクトに書き込むデータ。len が data の長さより小さく指定されている場合、 len バイトのみが書き込まれます。

len

書き込むデータの最大バイト数（オプション）。0 より大きく、かつ data のサイズ以下でなければなりません。 デフォルトは data の長さです。

返り値

ラージオブジェクトに書き込んだバイト数を返します。 エラー時には **FALSE** を返します。

例

Example#1 pg_lo_write() の例

```
<?php
    $doc_oid = 189762345;
   $data = "This will overwrite the start of the large object.";
   $database = pg_connect("dbname=jacarta");
   pg_query($database, "begin");
   $handle = pg_lo_open($database, $doc_oid, "w");
   $data = pg_lo_write($handle, $data);
   pg_query($database, "commit");
?>
```

参考

- pg_lo_create()
- pg_lo_open()

---

pg_meta_data

(PHP 4 >= 4.3.0, PHP 5)

pg_meta_data — テーブルからメタデータを取得する

説明

array **pg_meta_data** (resource $connection , string $table_name)

pg_meta_data() は、table_name のテーブル定義を配列として返します。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

テーブルの名前。

返り値

テーブル定義の配列を返します。エラー時には **FALSE** を返します。

例

Example#1 テーブルのメタデータを得る

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  $meta = pg_meta_data($dbconn, 'authors');
  if (is_array($meta)) {
       echo '<pre>';
      var_dump($meta);
       echo '</pre>';
  }
?>
```

上の例の出力は以下となります。

```
array(3) {
["author"]=>
array(5) {
   ["num"]=>
   int(1)
   ["type"]=>
   string(7) "varchar"
   ["len"]=>
   int(-1)
   ["not null"]=>
   bool(false)
   ["has default"]=>
   bool(false)
}
["year"]=>
array(5) {
   ["num"]=>
   int(2)
   ["type"]=>
   string(4) "int2"
   ["len"]=>
   int(2)
   ["not null"]=>
   bool(false)
   ["has default"]=>
   bool(false)
}
["title"]=>
array(5) {
   ["num"]=>
   int(3)
   ["type"]=>
   string(7) "varchar"
   ["len"]=>
   int(-1)
   ["not null"]=>
   bool(false)
   ["has default"]=>
   bool(false)
}
}
```

参考

- pg_convert()

---

pg_num_fields

(PHP 4 >= 4.2.0, PHP 5)

pg_num_fields — フィールド数を返す

説明

int **pg_num_fields** (resource $result)

pg_num_fields() は、PostgreSQL 結果リソースから フィールド（カラム）の数を返します。

> **注意**: この関数は、以前は **pg_numfields()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

返り値

結果のフィールド（カラム）数を返します。エラー時には -1 を返します。

例

Example#1 pg_num_fields() の例

```
<?php
$result = pg_query($conn, "SELECT 1, 2");

$num = pg_num_fields($result);

echo $num . " field(s) returned.¥n";
?>
```

上の例の出力は以下となります。

```
2 field(s) returned.
```

参考

- pg_num_rows()
- pg_affected_rows()

---

pg_num_rows

(PHP 4 >= 4.2.0, PHP 5)

pg_num_rows — 行数を返す

説明

int **pg_num_rows** (resource $result)

pg_num_rows() は、PostgreSQL の結果リソースの 行数を返します。

> **注意**: この関数は、以前は **pg_numrows()** と呼ばれていました。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

返り値

結果の行数を返します。エラー時には -1 を返します。

例

Example#1 pg_num_rows() の例

```
<?php
$result = pg_query($conn, "SELECT 1");

$rows = pg_num_rows($result);

echo $rows . " row(s) returned.¥n";
?>
```

上の例の出力は以下となります。

```
1 row(s) returned.
```

参考

- pg_num_fields()
- pg_affected_rows()

---

pg_options

(PHP 4, PHP 5)

pg_options — 接続に関連するオプションを取得する

説明

string **pg_options** ([resource $connection])

pg_options() は、指定した PostgreSQL connection リソースの オプションを保持する文字列を返します。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

connection のオプションを文字列で返します。 エラー時には **FALSE** を返します。

例

Example#1 pg_options() の例

```
<?php
    $pgsql_conn = pg_connect("dbname=mark  host=localhost");
   echo pg_options($pgsql_conn);
?>
```

参考

- pg_connect()

---

pg_parameter_status

(PHP 5)

pg_parameter_status — サーバのパラメータ設定を検索する

説明

string **pg_parameter_status** (resource $connection , string $param_name)
string **pg_parameter_status** (string $param_name)

サーバのパラメータ設定を検索します。

いくつかのパラメータについては、接続の確立時や値の変更時にサーバから 自動的に通知されます。**pg_parameter_status()** は これらの設定問い合わせるために使用可能です。指定したパラメータが存在する 場合にその値を、存在しない場合に **FALSE** を返します。

PostgreSQL 8.0 で指定できるパラメータには以下が含まれます。 server_version、 server_encoding、client_encoding、 is_superuser、session_authorization、 DateStyle、TimeZone および integer_datetimes (server_encoding、TimeZone および integer_datetimes は 8.0 より前のバージョンでは指定できません)。 server_version、server_encoding および integer_datetimes は、PostgreSQL の稼動中には変更できないことに注意しましょう。

PostgreSQL 7.3 以前のサーバはパラメータ設定を通知する機能を持っていません。 しかし、**pg_parameter_status()** には server_version および client_encoding を取得する機能を組み込んでいます。これらの値を取得するためにアプリケーションで アドホックなコードを書くのではなく、**pg_parameter_status()** を使用することを推奨します。

警告

7.4 より前の PostgreSQL サーバでは、接続の確立後に SET を用いて client_encoding を変更しても **pg_parameter_status()** には反映されません。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

param_name

指定可能な param_name には以下が含まれます。 server_version、 server_encoding、client_encoding、 is_superuser、session_authorization、 DateStyle、TimeZone および integer_datetimes。

返り値

パラメータの値を文字列で返します。失敗した場合や param_name が間違っている場合には **FALSE** を返します。

例

Example#1 pg_parameter_status() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

   echo "Server encoding: ", pg_parameter_status($dbconn, "server_encoding");
?>
```

上の例の出力は以下となります。

```
Server encoding: SQL_ASCII
```

---

pg_pconnect

(PHP 4, PHP 5)

pg_pconnect — 持続的な PostgreSQL 接続をオープンする

説明

```
resource pg_pconnect ( string $connection_string [, int $connect_type ] )
```

pg_pconnect() は PostgreSQL への接続を オープンします。この関数は、他の PostgreSQL関数 が必要とする 接続リソースを返します。

既存の接続と同じ connection_string を用いて **pg_pconnect()** が 2 回目にコールされた場合は、 既存の接続を返します。ただし、 connect_type に **PGSQL_CONNECT_FORCE_NEW** を指定している場合は 除きます。

持続的接続を有効にするには、php.ini のディレクティブ pgsql.allow_persistent を "On" に設定する必要があります（デフォルトは On です）。 持続的接続の最大数は php.ini ディレクティブ pgsql.max_persistent で設定可能です（デフォルトは制限無しを意味する -1 です）。 接続全体の数は php.ini ディレクティブ pgsql.max_links で設定可能です。

pg_close() は、 **pg_pconnect()** によりオープンされた持続的接続は 閉じません。

パラメータ

connection_string

すべてデフォルトのパラメータを使用する場合には connection_string を空にすることが可能です。 または 1 つ以上のパラメータを空白で区切って指定することも可能です。 個々のパラメータは keyword = value の形式で 設定します。等号の前後の空白はあってもなくてもかまいません。 空の値や空白を含む値を指定する場合は、その値をシングルクォートで 囲みます（例: keyword = 'a value'）。値の中に シングルクォートやバックスラッシュが含まれる場合は、それらを バックスラッシュでエスケープする必要があります（例: \' および \\）。

現在利用できるパラメータは以下のとおりです。 host , hostaddr , port , dbname , user , password , connect_timeout , options , tty （無視されます）, sslmode , requiressl （非推奨。代わりに sslmode を推奨します） および service 。これらのうち実際にどの パラメータが使えるかは、PostgreSQL のバージョンに依存します。

connect_type

PGSQL_CONNECT_FORCE_NEW が渡された場合は、 たとえ connection_string が既存の接続と まったく同一であっても新しい接続をオープンします。

返り値

成功した場合に PostgreSQL の接続リソース、失敗した場合に **FALSE** を返します。

例

Example#1 pg_pconnect() の使用法

```
<?php
$dbconn = pg_pconnect("dbname=mary");
// "mary"という名前のデータベースに接続

$dbconn2 = pg_pconnect("host=localhost port=5432 dbname=mary");
// "localhost"のポート"5432"にて"mary"という名前のデータベースに接続

$dbconn3 = pg_pconnect("host=sheep port=5432 dbname=mary user=lamb password=foo");
// ユーザ名とパスワードを指定してホスト"sheep"上の"mary"という名前のデータベースに接続

$conn_string = "host=sheep port=5432 dbname=test user=lamb password=bar";
$dbconn4 = pg_pconnect($conn_string);
// ユーザ名とパスワードを指定してホスト"sheep"上の"test"という名前のデータベースへ接続
?>
```

参考

- pg_connect()
- 持続的データベース接続

---

pg_ping

(PHP 4 >= 4.3.0, PHP 5)

pg_ping — データベース接続を調べる

説明

bool **pg_ping** ([resource $connection])

pg_ping() はデータベース接続を調べ、その接続が 壊れている場合には再度接続を試みます。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_ping() の例

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
  echo "エラーが発生しました。\n";
   exit;
}

if (!pg_ping($conn))
  die("接続は壊れています。\n");
?>
```

参考

- pg_connection_status()
- pg_connection_reset()

---

pg_port

(PHP 4, PHP 5)

pg_port — 接続に関連するポート番号を返す

説明

int **pg_port** ([resource $connection])

pg_port() は、指定した PostgreSQL connection リソースが接続している ポートの番号を返します。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

connection が指しているデータベースのポート番号を 含む int を返します。エラー時には **FALSE** を返します。

例

Example#1 pg_port() の例

```
<?php
$pgsql_conn = pg_connect("dbname=mark  host=localhost");

if ($pgsql_conn) {
    print "Successfully connected to port: " . pg_port($pgsql_conn) . "<br/>¥n";
} else {
    print pg_last_error($pgsql_conn);
    exit;
}
?>
```

---

pg_prepare

(PHP 5 >= 5.1.0)

pg_prepare — 指定したパラメータでプリペアドステートメントを作成するリクエストを 送信し、その完了を待つ

説明

```
resource pg_prepare ( resource $connection , string $stmtname , string $query )
resource pg_prepare ( string $stmtname , string $query )
```

pg_prepare() は、 pg_execute() あるいは pg_send_execute() で後に実行するためのプリペアドステートメントを作成します。これにより、繰り返し使用されるコマンドについての構文解析や実行計画作成が最初の 一度だけですみます。**pg_prepare()** は PostgreSQL 7.4 以降の接続でのみ使用可能です。それ以前のバージョンでは失敗します。

この関数は stmtname という名前の プリペアドステートメントを query 文字列から作成します。 この文字列には 1 つの SQL コマンドが含まれている必要があります。 stmtname を "" にすることで無名ステートメントを 作成することが可能で、既存の無名ステートメントは自動的に上書きされます。 それ以外の場合、もしカレントのセッションで既に定義済みのステートメント名を 使用した場合にはエラーとなります。パラメータを使用する際は、 query 内で $1、$2 のような形式で参照されます。

pg_prepare() で使用するプリペアドステートメントは、 SQLの PREPARE 文を実行することでも作成可能です （しかし、パラメータの型を事前に指定する必要がないという点で **pg_prepare()** のほうがより柔軟です）。 また、PHP にはプリペアドステートメントを削除する関数がありませんが、 この目的のためには SQLの DEALLOCATE 文が使用可能です。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

stmtname

プリペアドステートメントにつける名前。接続内で一意である必要があります。 "" が指定された場合は無名ステートメントが作成され、以前に定義された 無名ステートメントを上書きします。

query

パラメータ化した SQL 文。ひとつの文のみである必要があります （複数の文をセミコロンで区切る形式は使用できません）。パラメータを 使用する際は $1、$2 などの形式で参照されます。

返り値

成功した場合にクエリ結果リソース、失敗した場合に **FALSE** を返します。

例

Example#1 pg_prepare() の使用法

```
<?php
// "mary"という名前のデータベースに接続
$dbconn = pg_connect("dbname=mary");

// 実行するクエリの準備
$result = pg_prepare($dbconn, "my_query", 'SELECT * FROM shops WHERE name = $1');

// プリペアドクエリを実行する。文字列 "Joe's Widgets" は
// エスケープの必要がないことに注意
$result = pg_execute($dbconn, "my_query", array("Joe's Widgets"));

// 同一プリペアドクエリを別のパラメータで実行する
$result = pg_execute($dbconn, "my_query", array("Clothes Clothes Clothes"));

?>
```

参考

- pg_execute()
- pg_send_execute()

---

pg_put_line

(PHP 4 >= 4.0.3, PHP 5)

pg_put_line — NULL で終わる文字列を PostgreSQL バックエンドに送信する

説明

```
bool pg_put_line ( string $data )
bool pg_put_line ( resource $connection , string $data )
```

pg_put_line() は、NULL で終わる文字列を PostgreSQL バックエンドサーバに送信します。これは、PostgreSQL の COPY FROM コマンドとともに使用する場合に必要となります。

COPY は、PostgreSQL によってサポートされている 高速なデータ読み込みインターフェースです。データの内容はパースされず、 一度のトランザクションで実行されます。

低レベルな **pg_put_line()** コマンドを用いない別の方法は、 pg_copy_from() を使用することです。これは、はるかに シンプルなインターフェースです。

> **注意**: pg_end_copy() を実行する際には、送信データの最後に 明示的に "\." の 2 文字を送信する必要があります。これによって、バックエンドに対してデータ送信の終了を通知します。

警告

pg_put_line() の使用は、pg_lo_read() や pg_lo_tell() などを含むラージオブジェクトの操作を 発生させ、これが失敗することもあります。そのような場合、かわりに pg_copy_from() および pg_copy_to() が使用可能です。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

data

PostgreSQL バックエンドに直接送信されるテキストデータ。 最後に NULL が自動的に付加されます。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 pg_put_line() の例

```
<?php
  $conn = pg_pconnect("dbname=foo");
  pg_query($conn, "create table bar (a int4, b char(16), d float8)");
  pg_query($conn, "copy bar from stdin");
  pg_put_line($conn, "3¥thello world¥t4.5¥n");
  pg_put_line($conn, "4¥tgoodbye world¥t7.11¥n");
  pg_put_line($conn, "¥¥.¥n");
  pg_end_copy($conn);
?>
```

参考

- pg_end_copy()

---

pg_query_params

(PHP 5 >= 5.1.0)

pg_query_params — SQL コマンドとパラメータを分割してサーバにを送信し、その結果を待つ

説明

```
resource pg_query_params ( resource $connection , string $query , array $params )
resource pg_query_params ( string $query , array $params )
```

コマンドをサーバに送信し、その結果を待ちます。パラメータを SQL コマンド とは別に渡すことが可能です。

***pg_query_params()** は pg_query() と似ていますが、追加の機能を有しています。それはパラメータ値が コマンド文字列と分離しているということです。 **pg_query_params()** は PostgreSQL 7.4 以降の接続でのみ サポートされます。それ以前のバージョンでは失敗します。*

*パラメータを使用する際は、query 文字列内で $1、$2 のように参照されます。params で 実際の値を指定します。**NULL** を指定すると、SQL の NULL とみなされます。*

*pg_query() に対する **pg_query_params()** の最大の利点は、パラメータの値を query 文字列から 分離できることです。そのため、退屈でエラーの元となりやすいクォート・ エスケープなどをしなくてもよくなります。pg_query() と異なり、**pg_query_params()** ではひとつの SQL コマンドしか実行できません（クエリ文字列にセミコロンを含めることは 可能です。しかしそれ以降にコマンドを続けることはできません）。*

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

query

パラメータ化した SQL 文。ひとつの文のみである必要があります （複数の文をセミコロンで区切る形式は使用できません）。パラメータを 使用する際は $1、$2 などの形式で参照されます。

params

プリペアドステートメント中の $1、$2 などのプレースホルダを 置き換えるパラメータの配列。配列の要素数はプレースホルダの 数と一致する必要があります。

返り値

*成功した場合にクエリ結果リソース、失敗した場合に **FALSE** を返します。*

例

Example#1 pg_query_params() の使用法

```
<?php
// "mary"という名前のデータベースに接続
$dbconn = pg_connect("dbname=mary");

// Joe's Widgets という名前の店を探す。"Joe's Widgets" を
// エスケープする必要がないことに注意
$result = pg_query_params($dbconn, 'SELECT * FROM shops WHERE name = $1', array("Joe's Widgets"));

// pg_query を使用した場合と比較
$str = pg_escape_string("Joe's Widgets");
$result = pg_query($dbconn, "SELECT * FROM shops WHERE name = '{$str}'");
```

```
?>
```

参考

- pg_query()

---

pg_query

(PHP 4 >= 4.2.0, PHP 5)

pg_query — クエリを実行する

説明

```
resource pg_query ( string $query )
resource pg_query ( resource $connection , string $query )
```

pg_query() 指定したデータベース connection 上で query を実行します。

エラーが発生して **FALSE** が返された場合、もし接続が正常なら pg_last_error() 関数を使用してエラーの詳細情報が 取得可能です。

> **注意**: connection は省略可能ですが、それは推奨されません。 なぜならスクリプトのバグが発見しにくくなるためです。
>
> **注意**: この関数は、以前は **pg_exec()** と呼ばれていました。 **pg_exec()** は互換性確保のためにまだ使用可能ですが、 新しい名前を使用することが推奨されています。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

query

実行する 1 つまたは複数の SQL 文。複数の文が関数に渡された場合は、 明示的に BEGIN/COMMIT コマンドを指定していない限りはそれらの文はひとつのトランザクションとして実行されます。しかし、1 回のコールで 複数のトランザクションを実行することは推奨されません。

返り値

成功した場合にクエリ結果リソース、失敗した場合に **FALSE** を返します。

例

Example#1 pg_query() の例

```
<?php

$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

$result = pg_query($conn, "SELECT author, email FROM authors");
if (!$result) {
   echo "An error occured.¥n";
   exit;
}

while ($row = pg_fetch_row($result)) {
   echo "Author: $row[0]  E-mail: $row[1]";
   echo "<br />¥n";
}

?>
```

Example#2 pg_query() で複数の文を使用する例

```
<?php

$conn = pg_pconnect("dbname=publisher");

// これらの文がひとつのトランザクションとして実行されます

$query = "UPDATE authors SET author=UPPER(author) WHERE id=1;";
$query .= "UPDATE authors SET author=LOWER(author) WHERE id=2;";
$query .= "UPDATE authors SET author=NULL WHERE id=3;";

pg_query($conn, $query);

?>
```

参考

- pg_connect()
- pg_pconnect()
- pg_fetch_array()
- pg_fetch_object()

- pg_num_rows()
- pg_affected_rows()

---

pg_result_error_field

(PHP 5 >= 5.1.0)

pg_result_error_field — エラー報告の各フィールドを返す

説明

string **pg_result_error_field** (resource $result , int $fieldcode)

pg_result_error_field() は、result リソースに関するエラーメッセージの特定のフィールドを返します。 PostgreSQL 7.4 以降のサーバでのみ有効です。エラーフィールドは fieldcode で定義します。

pg_query() や pg_query_params() はクエリが失敗した場合に **FALSE** を返すので、結果のハンドルを得るには pg_send_query() および pg_get_result() を使用する必要があります。

失敗した pg_query() クエリから詳細なエラー情報を 取得する必要がある場合は、pg_set_error_verbosity() および pg_last_error() を使用してその結果をパース します。

パラメータ

result

実行したステートメントのクエリ結果リソース。

fieldcode

以下の fieldcode が使用可能です。 **PGSQL_DIAG_SEVERITY**、 **PGSQL_DIAG_SQLSTATE**、**PGSQL_DIAG_MESSAGE_PRIMARY**、**PGSQL_DIAG_MESSAGE_DETAIL**、 **PGSQL_DIAG_MESSAGE_HINT**、**PGSQL_DIAG_STATEMENT_POSITION**、 **PGSQL_DIAG_INTERNAL_POSITION** (PostgreSQL 8.0+ のみ)、 **PGSQL_DIAG_INTERNAL_QUERY** (PostgreSQL 8.0+ のみ)、 **PGSQL_DIAG_CONTEXT**、**PGSQL_DIAG_SOURCE_FILE**、 **PGSQL_DIAG_SOURCE_LINE** あるいは **PGSQL_DIAG_SOURCE_FUNCTION** 。

返り値

エラーフィールドの内容を文字列で返します。 フィールドが存在しない場合に **NULL** 、失敗した場合に **FALSE** を返します。

例

Example#1 pg_result_error_field() の例

```
<?php
  $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  if (!pg_connection_busy($dbconn)) {
      pg_send_query($dbconn, "select * from doesnotexist;");
  }

  $res1 = pg_get_result($dbconn);
  echo pg_result_error_field($res1, PGSQL_DIAG_SQLSTATE);
?>
```

参考

- pg_result_error()

---

pg_result_error

(PHP 4 >= 4.2.0, PHP 5)

pg_result_error — 結果に関連するエラーメッセージを取得する

説明

string **pg_result_error** (resource $result)

pg_result_error() は、結果リソース (result)に関連したエラーメッセージを返します。 このため、ユーザーが pg_last_error() よりも適切な エラーメッセージを得る可能性があります。

pg_result_error_field() は、 **pg_result_error()** よりもさらに詳細なエラー情報を 返します。

pg_query() はクエリが失敗した場合に **FALSE** を返すので、 結果ハンドルを取得するには pg_send_query() および pg_get_result() を使用する必要があります。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

返り値

result パラメータに関連付けられたエラーがある場合は それを文字列で返し、それ以外の場合は **FALSE** を返します。

例

Example#1 pg_result_error() の例

```
<?php
  $dbconn = pg_connect("dbname=publisher") or die("接続できませんでした");

  if (!pg_connection_busy($dbconn)) {
      pg_send_query($dbconn, "select * from doesnotexist;");
  }

  $res1 = pg_get_result($dbconn);
  echo pg_result_error($res1);
?>
```

参考

- pg_result_error_field()
- pg_query()
- pg_send_query()
- pg_get_result()
- pg_last_error()
- pg_last_notice()

---

pg_result_seek

(PHP 4 >= 4.3.0, PHP 5)

pg_result_seek — 結果リソースの内部行オフセットを設定する

説明

bool **pg_result_seek** (resource $result , int $offset)

pg_result_seek()は、結果リソースの行の位置を 指定された offset にセットします。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

offset

結果リソース内で、内部オフセットを移動させる行。 行番号はゼロから始まります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_result_seek() の例

```
<?php

// データベースに接続する
$conn = pg_pconnect("dbname=publisher");

// クエリを実行する
$result = pg_query($conn, "SELECT author, email FROM authors");

// 3 行目をシークする（結果が 3 行あると仮定する）
pg_result_seek($result, 2);

// 3 行目を取得する
$row = pg_fetch_row($result);

?>
```

参考

- pg_fetch_row()
- pg_fetch_assoc()
- pg_fetch_array()
- pg_fetch_object()
- pg_fetch_result()

---

pg_result_status

(PHP 4 >= 4.2.0, PHP 5)

pg_result_status — クエリ結果のステータスを取得する

説明

mixed **pg_result_status** (resource $result [, int $type])

pg_result_status() は、結果リソースのステータス あるいは結果に関するコマンド補完タグを返します。

パラメータ

result

pg_query(), pg_query_params() あるいは pg_execute() から返される PostgreSQL の クエリ結果リソース。

type

result の数値ステータスを返す **PGSQL_STATUS_LONG**、あるいは result のコマンドタグを返す **PGSQL_STATUS_STRING** のいずれかです。 指定しない場合は **PGSQL_STATUS_LONG** がデフォルトです。

返り値

PGSQL_STATUS_LONG が指定された場合の返り値は 以下のいずれかです。 **PGSQL_EMPTY_QUERY**、 **PGSQL_COMMAND_OK**、**PGSQL_TUPLES_OK**、**PGSQL_COPY_OUT**、 **PGSQL_COPY_IN**、**PGSQL_BAD_RESPONSE**、**PGSQL_NONFATAL_ERROR** および **PGSQL_FATAL_ERROR** 。 それ以外の場合は、PostgreSQL コマンドタグを含む文字列を返します。

変更履歴

バージョン	説明
4.3.0	パラメータ type が追加されました。

例

Example#1 pg_result_status() の例

```
<?php

// データベースに接続する
$conn = pg_pconnect("dbname=publisher");

// COPY を実行する
$result = pg_query($conn, "COPY authors FROM STDIN;");

// 結果ステータスを得る
$status = pg_result_status($result);

// ステータスの内容を調べる
if ($status == PGSQL_COPY_IN)
    echo "Copy began.";
else
    echo "Copy failed.";

?>
```

上の例の出力は以下となります。

```
Copy began.
```

参考

- pg_connection_status()

---

pg_select

(PHP 4 >= 4.3.0, PHP 5)

pg_select — レコードを選択する

説明

mixed **pg_select** (resource $connection , string $table_name , array $assoc_array [, int $options])

pg_select() は、field=>value 形式の assoc_array で指定したレコードを選択します。 クエリに成功した場合、assoc_array で指定した条件に マッチする全てのレコードとフィールドを含む配列が返されます。

options が指定された場合、 指定したフラグとともに pg_convert() が assoc_array に適用されます。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

行を選択するテーブルの名前。

assoc_array

テーブル *table_name* のフィールド名をキーに、 そして取得対象となる行にマッチするデータを値にもつ配列。

options

PGSQL_CONV_FORCE_NULL、 ***PGSQL_DML_NO_CONV***、 ***PGSQL_DML_EXEC***、 ***PGSQL_DML_ASYNC*** あるいは ***PGSQL_DML_STRING*** の組み合わせ。 *options* の一部に ***PGSQL_DML_STRING*** が含まれていた場合、 クエリ文字列が返されます。

返り値

成功した場合に ***TRUE*** を、失敗した場合に ***FALSE*** を返します。 *options* に ***PGSQL_DML_STRING*** が渡された場合は文字列を返します。

例

Example#1 pg_select() の例

```
<?php
  $db = pg_connect('dbname=foo');
  // これは安全です。なぜなら $_POST は自動的に変換されるからです。
  $rec = pg_select($db, 'post_log', $_POST);
  if ($rec) {
      echo "選択されたレコード:\n";
      var_dump($rec);
  } else {
      echo "ユーザが誤った入力を送信しました。\n";
  }
?>
```

参考

- pg_convert()

---

pg_send_execute

(PHP 5 >= 5.1.0)

pg_send_execute — 指定したパラメータでプリペアドステートメントを実行するリクエストを 送信し、その結果を待たない

説明

bool ***pg_send_execute*** *(resource $connection , string $stmtname , array $params)*

指定したパラメータでプリペアドステートメントを実行するリクエストを 送信し、その結果を待ちません。

これは pg_send_query_params() と似ています。 しかし、実行するコマンドは指定したクエリ文字列で決まるのではなく 事前に準備されたステートメントの名前で決まります。関数のパラメータは pg_execute() と同じように処理されます。 pg_execute() と同様に、7.4 より前のバージョンの PostgreSQL では動作しません。

パラメータ

connection

PostgreSQL データベース接続リソース。*connection* が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

stmtname

実行するプリペアドステートメントの名前。"" が指定された場合は 無名ステートメントが実行されます。名前は、事前に pg_prepare() ・ pg_send_prepare() あるいは *PREPARE SQL* コマンドで準備されたものである 必要があります。

params

プリペアドステートメント中の *$1*、*$2* などのプレースホルダを 置き換えるパラメータの配列。配列の要素数はプレースホルダの 数と一致する必要があります。

返り値

成功した場合に ***TRUE*** 、失敗した場合に ***FALSE*** を返します。 クエリの結果を確認するには pg_get_result() を使用します。

例

Example#1 pg_send_execute() の使用法

```
<?php
  $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  // 実行するクエリを準備する
  if (!pg_connection_busy($dbconn)) {
    pg_send_prepare($dbconn, "my_query", 'SELECT * FROM shops WHERE name = $1');
    $res1 = pg_get_result($dbconn);
  }

  // プリペアドクエリを実行する。 文字列 "Joe's Widgets" は
  // エスケープの必要がないことに注意
```

```
  if (!pg_connection_busy($dbconn)) {
    pg_send_execute($dbconn, "my_query", array("Joe's  Widgets"));
    $res2 = pg_get_result($dbconn);
  }

  // 同じプリペアドクエリを異なるパラメータで実行する
  if (!pg_connection_busy($dbconn)) {
    pg_send_execute($dbconn, "my_query", array("Clothes  Clothes  Clothes"));
    $res3 = pg_get_result($dbconn);
  }

?>
```

参考

- pg_prepare()
- pg_send_prepare()
- pg_execute()

---

pg_send_prepare

(PHP 5 >= 5.1.0)

pg_send_prepare — 指定したパラメータでプリペアドステートメントを作成するリクエストを 送信し、その結果を待たない

説明

bool **pg_send_prepare** (resource $connection , string $stmtname , string $query)

指定したパラメータでプリペアドステートメントを作成するリクエストを 送信し、その結果を待ちません。

これは pg_prepare() の非同期バージョンです。 リクエストが受け付けられた場合に **TRUE** 、そうでない場合に **FALSE** を返します。コールが成功した後、実際にプリペアドステートメントが 作成されたかどうかを調べるには pg_get_result() を使用します。関数のパラメータは pg_prepare() と同じように処理されます。pg_prepare() と同様、 7.4 より前の PostgreSQL のバージョンでは正しく動作しません。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

stmtname

プリペアドステートメントにつける名前。接続内で一意である必要があります。 "" が指定された場合は無名ステートメントが作成され、以前に定義された 無名ステートメントを上書きします。

query

パラメータ化した SQL 文。ひとつの文のみである必要があります （複数の文をセミコロンで区切る形式は使用できません）。パラメータを 使用する際は $1、$2 などの形式で参照されます。

返り値

成功した場合に **TRUE** 、失敗した場合に **FALSE** を返します。 クエリの結果を確かめるには pg_get_result() を使用します。

例

Example#1 pg_send_prepare() の使用法

```
<?php
  $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  // 実行するクエリを準備する
  if (!pg_connection_busy($dbconn)) {
    pg_send_prepare($dbconn, "my_query", 'SELECT * FROM shops WHERE name = $1');
    $res1 = pg_get_result($dbconn);
  }

  // プリペアドクエリを実行する。文字列 "Joe's Widgets" は
  // エスケープの必要がないことに注意
  if (!pg_connection_busy($dbconn)) {
    pg_send_execute($dbconn, "my_query", array("Joe's  Widgets"));
    $res2 = pg_get_result($dbconn);
  }

  // 同一プリペアドクエリを別のパラメータで実行する
  if (!pg_connection_busy($dbconn)) {
    pg_send_execute($dbconn, "my_query", array("Clothes  Clothes  Clothes"));
    $res3 = pg_get_result($dbconn);
  }

?>
```

参考

- pg_connect()
- pg_pconnect()
- pg_execute()

- pg_send_execute()
- pg_send_query_params()

---

pg_send_query_params

(PHP 5 >= 5.1.0)

pg_send_query_params — コマンドとパラメータを分割してサーバに送信し、その結果を待たない

説明

bool **pg_send_query_params** (resource $connection , string $query , array $params)

コマンドとパラメータを分割してサーバに送信します。その結果を待つことは しません。

これは pg_send_query() とほぼ同じですが、パラメータが query とは別に分かれている点が違います。 関数のパラメータは、pg_query_params() によって 完全に制御されます。pg_query_params() と同様、 7.4 より前の PostgreSQL では動作しません。またクエリ文字列には ひとつのコマンドのみを含めることができます。

パラメータ

connection

PostgreSQL データベースの接続リソース。

query

パラメータ化した SQL 文。ひとつの文のみである必要があります （複数の文をセミコロンで区切る形式は使用できません）。パラメータを 使用する際は $1、$2 などの形式で参照されます。

params

プリペアドステートメント中の $1、$2 などのプレースホルダを 置き換えるパラメータの配列。配列の要素数はプレースホルダの 数と一致する必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

クエリの結果を判断するには pg_get_result() を使用します。

例

Example#1 pg_send_query_params() の使用法

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

  // パラメータを使用する。パラメータにはクォートやエスケープが
  // 必要ないことに注意
  pg_send_query_params($dbconn, 'select count(*) from authors where city = $1', array('Perth'));

  // 基本的な pg_send_query の使用法との比較
  $str = pg_escape_string('Perth');
  pg_send_query($dbconn, "select count(*) from authors where city = '${str}'");
?>
```

参考

- pg_send_query()

---

pg_send_query

(PHP 4 >= 4.2.0, PHP 5)

pg_send_query — 非同期クエリを送信する

説明

bool **pg_send_query** (resource $connection , string $query)

pg_send_query() は、connection に非同期クエリを送信します。pg_query() とは異なり、 PostgreSQL へ一度に複数のクエリを送信することができ、 pg_get_result() を用いて結果を一つずつ取得することが可能です。

スクリプトの実行は、クエリを実行中もブロックされません。 接続がビジーである(すなわち、クエリが実行中である)ことを調べるには、pg_connection_busy() を使用してください。 クエリは、pg_cancel_query() をコールすることによりキャンセルすることが可能です。

ユーザは複数のクエリを一度に送信することができますが、 複数のクエリをビジー状態の接続に送信することはできません。 クエリがビジー状態の接続に送信された場合、 最後のクエリが終了するまで待ち、全ての結果は破棄されます。

パラメータ

connection

PostgreSQL データベース接続リソース。

query

実行するひとつまたは複数の SQL 文。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

クエリの結果を利用するには pg_get_result() を使用します。

例

Example#1 pg_send_query() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

   if (!pg_connection_busy($dbconn)) {
       pg_send_query($dbconn, "select * from authors; select count(*) from authors;");
   }

   $res1 = pg_get_result($dbconn);
   echo "First call to pg_get_result(): $res1¥n";
   $rows1 = pg_num_rows($res1);
   echo "$res1 has $rows1 records¥n¥n";

   $res2 = pg_get_result($dbconn);
   echo "Second call to pg_get_result(): $res2¥n";
   $rows2 = pg_num_rows($res2);
   echo "$res2 has $rows2 records¥n";
?>
```

上の例の出力は以下となります。

```
First call to pg_get_result(): Resource id #3
Resource id #3 has 3 records

Second call to pg_get_result(): Resource id #4
Resource id #4 has 1 records
```

参考

- pg_query()
- pg_cancel_query()
- pg_get_result()
- pg_connection_busy()

---

pg_set_client_encoding

(PHP 4 >= 4.0.3, PHP 5)

pg_set_client_encoding — クライアントのエンコーディングを設定する

説明

```
int pg_set_client_encoding ( string $encoding )
int pg_set_client_encoding ( resource $connection , string $encoding )
```

***pg_set_client_encoding()** はクライアントの エンコーディングを設定し、成功した場合に 0、エラー時に -1 を 返します。*

PostgreSQL は、バックエンドのデータベースエンコーディングを自動的に フロントエンドのエンコーディングに変換します。

> ***注意**: この関数は、以前は **pg_setclientencoding()** と呼ばれていました。*

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

encoding

クライアントエンコーディング。以下のうちのひとつです。SQL_ASCII、EUC_JP、 EUC_CN、EUC_KR、EUC_TW、 UNICODE、MULE_INTERNAL、 LATINX (X=1...9)、 KOI8、WIN、ALT、SJIS、 BIG5 あるいは WIN1250 。

利用可能なエンコーディングの完全なリストは、使用している PostgreSQL のバージョンに依存します。詳細な情報については PostgreSQL のマニュアルを 参照ください。

返り値

成功した場合に 0 、エラー時に -1 を返します。

例

Example#1 pg_set_client_encoding() の例

```
<?php

$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
   echo "An error occured.¥n";
   exit;
}

// クライアントのエンコーディングを UNICODE にする。
// データは、バックエンドのエンコーディングから自動的に変換される。
pg_set_client_encoding($conn, UNICODE);

$result = pg_query($conn, "SELECT author, email FROM authors");
if (!$result) {
   echo "An error occured.¥n";
   exit;
}

// UTF-8 データを書き出す。
while ($row = pg_fetch_row($result)) {
   echo "Author: $row[0]  E-mail: $row[1]";
   echo "<br />¥n";
}

?>
```

参考

- pg_client_encoding()

---

pg_set_error_verbosity

(PHP 5 >= 5.1.0)

pg_set_error_verbosity — pg_last_error() および pg_result_error() が返すメッセージの詳細度を指定する

説明

```
int pg_set_error_verbosity ( resource $connection , int $verbosity )
int pg_set_error_verbosity ( int $verbosity )
```

pg_last_error() および pg_result_error() が返すメッセージの詳細度を指定します。

pg_set_error_verbosity() は詳細度を設定し、その接続の これまでの設定を返します。**PGSQL_ERRORS_TERSE** モードでは、返されるメッセージは深刻度・概要 および 発生位置のみです。 これはたいていの場合 1 行に収まります。デフォルトのモード （**PGSQL_ERRORS_DEFAULT**）では、それに加えて何らかの 詳細情報・ヒントあるいは状況フィールドを含みます（これらは複数行に またがる可能性があります）。**PGSQL_ERRORS_VERBOSE** モードは、有効なフィールドをすべて含みます。詳細度の設定変更内容は それ以降に新しく作成した結果オブジェクトにのみ反映され、既存の 結果オブジェクトには影響を与えません。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

verbosity

指定する詳細度。**PGSQL_ERRORS_TERSE**、 **PGSQL_ERRORS_DEFAULT** あるいは **PGSQL_ERRORS_VERBOSE** 。

返り値

変更前の詳細度レベル。**PGSQL_ERRORS_TERSE**、 **PGSQL_ERRORS_DEFAULT** あるいは **PGSQL_ERRORS_VERBOSE** のいずれかを 返します。

例

Example#1 pg_set_error_verbosity() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");

   if (!pg_connection_busy($dbconn)) {
       pg_send_query($dbconn, "select * from doesnotexist;");
   }

   pg_set_error_verbosity($dbconn, PGSQL_ERRORS_VERBOSE);
   $res1 = pg_get_result($dbconn);
   echo pg_result_error($res1);
?>
```

参考

- pg_last_error()
- pg_result_error()

---

pg_trace

(PHP 4 >= 4.0.1, PHP 5)

pg_trace — PostgreSQL 接続のトレースを有効にする

説明

bool **pg_trace** (string $pathname [, string $mode [, resource $connection]])

pg_trace() は、PostgreSQL フロントエンド/ バックエンド間の通信をデバック用のファイルにトレースすることを有効にします。 このトレース結果を完全に理解するためには、PostgreSQL 通信プロトコルの詳細に 精通している必要があります。

そうでない人にとっても、サーバに送られたクエリのエラーを トレースすることは有用です。試しに **grep '^To backend' trace.log** を実行し、実際に PostgreSQL サーバに送信される クエリを見てみるとよいでしょう。詳細な情報は » PostgreSQL ドキュメントを 参照ください。

パラメータ

pathname

トレースログを書き込むファイルの名前（フルパスを含む）。 fopen() と同じ。

mode

オプションのファイルアクセスモード。fopen() と同じです。デフォルトは "w" です。

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pg_trace() の例

```
<?php
$pgsql_conn = pg_connect("dbname=mark host=localhost");

if ($pgsql_conn) {
   pg_trace('/tmp/trace.log', 'w', $pgsql_conn);
   pg_query("SELECT 1");
   pg_untrace($pgsql_conn);
   // /tmp/trace.log にバックエンドの通信が記録される
} else {
    print pg_last_error($pgsql_conn);
    exit;
}
?>
```

参考

- fopen()
- pg_untrace()

---

pg_transaction_status

(PHP 5 >= 5.1.0)

pg_transaction_status — サーバ上で実行中のトランザクションの状態を返す

説明

int **pg_transaction_status** (resource $connection)

サーバ上で実行中のトランザクションの状態を返します。

警告

PostgreSQL 7.3 サーバで autocommit を off に 設定している場合、**pg_transaction_status()** は 不正確な値を返します。サーバ側での自動コミット機能は非推奨と されており、最近のバージョンのサーバでは存在しません。

パラメータ

connection

PostgreSQL データベースの接続リソース。

返り値

PGSQL_TRANSACTION_IDLE（アイドル状態）、 **PGSQL_TRANSACTION_ACTIVE**（コマンドの実行中）、 **PGSQL_TRANSACTION_INTRANS**（正常なトランザクション内でアイドル状態）、 あるいは **PGSQL_TRANSACTION_INERROR**（失敗したトランザクション内でアイドル状態） のいずれかを返します。 接続が異常な場合は **PGSQL_TRANSACTION_UNKNOWN** を返します。 **PGSQL_TRANSACTION_ACTIVE** が返されるのは、クエリを サーバに送信した後まだそれが完了していない場合のみです。

例

Example#1 pg_transaction_status() の例

```
<?php
   $dbconn = pg_connect("dbname=publisher") or die("Could not connect");
  $stat = pg_transaction_status($dbconn);
  if ($stat === PGSQL_TRANSACTION_UNKNOWN) {
      echo 'Connection is bad';
  } else if ($stat === PGSQL_TRANSACTION_IDLE) {
      echo 'Connection is currently idle';
  } else {
      echo 'Connection is in a transaction state';
  }
?>
```

---

pg_tty

(PHP 4, PHP 5)

pg_tty — 接続に関する TTY 名を返す

説明

string **pg_tty** ([resource $connection])

pg_tty() は、指定した PostgreSQL connection リソースで、サーバ側のデバッグ出力が 送られる tty 名を返します。

> **注意**: **pg_tty()** は現在は使用されません。今はサーバが TTY 設定を気にする必要がないからです。しかし、過去との互換性を 保持するためにこの関数は残されています。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

connection のデバッグ TTY を文字列で返します。 エラー時には **FALSE** を返します。

例

Example#1 pg_tty() の例

```
<?php
$pgsql_conn = pg_connect("dbname=mark host=localhost");

if ($pgsql_conn) {
    print "Server debug TTY is: " . pg_tty($pgsql_conn) . "<br/>¥n";
} else {
    print pg_last_error($pgsql_conn);
    exit;
}
?>
```

---

pg_unescape_bytea

(PHP 4 >= 4.3.0, PHP 5)

pg_unescape_bytea — bytea 型のバイナリをアンエスケープする

説明

string **pg_unescape_bytea** (string $data)

pg_unescape_bytea() は、bytea 型のデータ文字列を アンエスケープし、エスケープしていない文字列(バイナリ)を返します。

> **注意**: bytea 型を SELECT した場合、PostgreSQL は '\' で 始まる 8 進数のバイト値（例: \032）を返します。これをユーザが手動で コンバートすることを期待されています。
> この関数は PostgreSQL 7.2 以降のバージョンを必要とします。PostgreSQL 7.2.0 および 7.2.1 では、マルチバイトのサポートを有効にした場合は bytea の値をキャストする必要があります。例: INSERT INTO test_table (image) VALUES ('$image_escaped'::bytea); PostgreSQL 7.2.2 以降ではキャストする必要はありません。 クライアントとバックエンドの文字エンコーディングが一致しない場合は 例外で、この場合はマルチバイトストリームエラーが発生します。この エラーを避けるためには bytea 型へのキャストが必要になります。

パラメータ

data

PHP のバイナリ文字列に変換する PostgreSQL の bytea データを含む 文字列。

返り値

アンエスケープされたデータを文字列で返します。

例

Example#1 pg_unescape_bytea() の例

```php
<?php
  // データベースに接続する
  $dbconn = pg_connect('dbname=foo');

  // bytea データを取得する
  $res = pg_query("SELECT data FROM gallery WHERE name='Pine trees'");
  $raw = pg_fetch_result($res, 'data');

  // バイナリに変換し、ブラウザに送信する
  header('Content-type: image/jpeg');
  echo pg_unescape_bytea($raw);
?>
```

参考

- pg_escape_bytea()
- pg_escape_string()

---

pg_untrace

(PHP 4 >= 4.0.1, PHP 5)

pg_untrace — PostgreSQL 接続のトレースを無効にする

説明

bool **pg_untrace** ([resource $connection])

pg_trace() で開始したトレースを停止します。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

Always returns **TRUE**.

例

Example#1 pg_untrace() の例

```php
<?php
$pgsql_conn = pg_connect("dbname=mark host=localhost");

if ($pgsql_conn) {
   pg_trace('/tmp/trace.log', 'w', $pgsql_conn);
   pg_query("SELECT 1");
   pg_untrace($pgsql_conn);
   // バックエンド通信のトレースを無効にする
} else {
    print pg_last_error($pgsql_conn);
    exit;
}
?>
```

参考

- pg_trace()

---

pg_update

(PHP 4 >= 4.3.0, PHP 5)

pg_update — テーブルを更新する

説明

mixed **pg_update** (resource $connection , string $table_name , array $data , array $condition [, int $options])

pg_update() は、data に関して condition にマッチするレコードを更新します。 options が指定された場合、指定したオプションとともに pg_convert() が data に適用されます。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

connection

PostgreSQL データベースの接続リソース。

table_name

行を更新するテーブルの名前。

data

テーブル table_name のフィールド名をキーに、 そしてマッチした対象を更新するデータを値にもつ配列。

condition

テーブル table_name のフィールド名をキーに、 そして取得対象となる行にマッチするデータを値にもつ配列。

options

PGSQL_CONV_OPTS、 **PGSQL_DML_NO_CONV**、 **PGSQL_DML_EXEC** あるいは **PGSQL_DML_STRING** の組み合わせ。 options の一部に **PGSQL_DML_STRING** が含まれていた場合、 クエリ文字列が返されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 options に **PGSQL_DML_STRING** が渡された場合は文字列を返します。

例

Example#1 pg_update() の例

```
<?php
  $db = pg_connect('dbname=foo');
  $data = array('field1'=>'AA', 'field2'=>'BB');

  // これは安全です。なぜなら $_POST は自動的に変換されるからです。
  $res = pg_update($db, 'post_log', $_POST, $data);
  if ($res) {
      echo "データが更新されました: $res\n";
  } else {
      echo "ユーザが誤った入力を送信しました。\n";
  }
?>
```

参考

- pg_convert()

---

pg_version

(PHP 5)

pg_version — クライアント・プロトコル・サーバのバージョンを配列で返す

説明

array **pg_version** ([resource $connection])

pg_version() はクライアント・プロトコル およびサーバのバージョンを配列で返します。プロトコルおよびサーバの バージョンは、PHP が PostgreSQL 7.4 以降とともにコンパイルされている 場合のみ有効です。

詳細なサーバ情報を取得するには pg_parameter_status() を参照ください。

パラメータ

connection

PostgreSQL データベース接続リソース。connection が指定されていない場合はデフォルトの接続が使用されます。 デフォルトの接続は、直近の pg_connect() あるいは pg_pconnect() によって作成されたものです。

返り値

client, protocol および server_version のキーとその値を持つ配列を 返します（有効な場合）。エラー時や接続が正常でない場合に **FALSE** を返します。

例

Example#1 pg_version() の例

```
<?php
   $dbconn = pg_connect("host=localhost port=5432 dbname=mary")
      or die("Could not connect");

  $v = pg_version($dbconn);

   echo $v['client'];
?>
```

上の例の出力は以下となります。

```
7.4
```

参考

- pg_parameter_status()

目次

PostgreSQL 関数 (PDO_PGSQL)

導入

PDO_PGSQL は、PHP から PostgreSQL データベースへのアクセスを可能にするための PHP Data Objects (PDO) インターフェース を実装したドライバです。

リソース型

この拡張モジュールでは、ストリームリソースを定義しています。これは **PDO::pgsqlLOBOpen()** が返すものです。

PDO_PGSQL DSN

(No version information available, might be only in CVS)

PDO_PGSQL DSN — PostgreSQL データベースに接続する

説明

PDO_PGSQL データソース名 (DSN) は以下の要素で構成され、 スペースで区切られます:

DSN 接頭辞

DSN 接頭辞は **pgsql:** です。

host

データベースサーバが存在するホスト名を指定します。

port

データベースサーバが起動しているポートを指定します。

dbname

データベース名を指定します。

user

接続するユーザー名を指定します。もし DSN でユーザー名を指定する場合、 PDO は PDO コンストラクタにおけるユーザー名引数の値を無視します。

password

接続するユーザーのパスワードを指定します。もし DSN でパスワードを指定する場合、 PDO は PDO コンストラクタにおけるパスワード引数の値を無視します。

> **注意**: bytea フィールドはストリームとして返されます。

例

Example#1 PDO_PGSQL DSN の例

以下の例は、PostgreSQL データベースに接続するための PDO_PGSQL DSN を表します:

```
pgsql:host=localhost port=5432 dbname=testdb user=bruce password=mypass
```

PDO::pgsqlLOBCreate

(PHP 5 >= 5.1.2, PECL pdo_pgsql:1.0.1-1.0.2)

PDO::pgsqlLOBCreate — 新しいラージオブジェクトを作成する

説明

```
string PDO::pgsqlLOBCreate ( void )
```

PDO::pgsqlLOBCreate() は、 ラージオブジェクトを作成してその OID を返します。 このオブジェクトに対するデータの読み書きを行う際には、 **PDO::pgsqlLOBOpen()** を使用してストリームをオープンします。 OID は OID 型のカラムに格納され、ラージオブジェクトを参照するために使用されます。 これにより、行のサイズがどんどん拡大してしまうことを防ぎます。 **PDO::pgsqlLOBUnlink()** をコールして削除するまで、 ラージオブジェクトはデータベース内に残り続けます。

ラージオブジェクトの大きさは最大 2GB まで拡大できますが、扱い方はめんどうです。 オブジェクトの OID を参照している行をデータベースから削除する際には、 必ず事前に **PDO::pgsqlLOBUnlink()** がコールされていなければなりません。 さらに、ラージオブジェクトにはアクセス権の設定がありません。 ラージオブジェクトの代替策として、bytea 型のカラムも検討ください。 最近のバージョンの PostgreSQL では bytea 型のカラムに最大 1GB まで保存でき、行サイズを最適化したうえでデータを透過的に扱うことができます。

> **注意**: この関数は、トランザクション内でコールしなければなりません。

パラメータ

PDO::pgsqlLOBCreate() は、パラメータを受け取りません。

返り値

新しく作成されたラージオブジェクトの OID、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 PDO::pgsqlLOBCreate() の例

この例では、新しいラージオブジェクトを作成し、 ファイルの内容をそこにコピーします。その後、OID がテーブルに保存されます。

```php
<?php
$db = new PDO('pgsql:dbname=test host=localhost', $user, $pass);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->beginTransaction();
$oid = $db->pgsqlLOBCreate();
$stream = $db->pgsqlLOBOpen($oid, 'w');
$local = fopen($filename, 'rb');
stream_copy_to_stream($local, $stream);
$local = null;
$stream = null;
$stmt = $db->prepare("INSERT INTO BLOBS (ident, oid) VALUES (?, ?)");
$stmt->execute(array($some_id, $oid));
$db->commit();
?>
```

参考

- **PDO::pgsqlLOBOpen()**
- **PDO::pgsqlLOBUnlink()**
- pg_lo_create()

PDO::pgsqlLOBOpen

(PHP 5 >= 5.1.2, PECL pdo_pgsql:1.0.1-1.0.2)

PDO::pgsqlLOBOpen — 既存のラージオブジェクトのストリームをオープンする

説明

```
resource PDO::pgsqlLOBOpen ( string $oid [, string $mode ] )
```

PDO::pgsqlLOBOpen() は、oid が指すデータにアクセスするためのストリームをオープンします。 mode が r の場合、 ストリームは読み込み用にオープンされます。 mode が w の場合、 ストリームは書き込み用にオープンされます。 fread()、fwrite() および fgets() のような通常のファイルシステム関数を使用して、 ストリームの内容を操作することができます。

> **注意**: この関数およびラージオブジェクトに対するすべての操作は、 トランザクション内で処理される必要があります。

パラメータ

oid

ラージオブジェクトの ID。

mode

モードが r の場合、読み込み用のストリームをオープンします。 モードが w の場合、書き込み用のストリームをオープンします。

返り値

成功した場合にストリームリソース、失敗した場合に **FALSE** を返します。

例

Example#1 PDO::pgsqlLOBOpen() の例

PDO::pgsqlLOBCreate() の例に引き続き、 このコード片はデータベースからラージオブジェクトを取得して それをブラウザに出力します。

```
<?php
$db = new PDO('pgsql:dbname=test host=localhost', $user, $pass);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->beginTransaction();
$stmt = $db->prepare("select oid from BLOBS where ident = ?");
$stmt->execute(array($some_id));
$stmt->bindColumn('oid', $lob, PDO::PARAM_LOB);
$stmt->fetch(PDO::FETCH_BOUND);
fpassthru($lob);
?>
```

参考

- **PDO::pgsqlLOBCreate()**
- **PDO::pgsqlLOBUnlink()**
- pg_lo_open()

PDO::pgsqlLOBUnlink

(PHP 5 >= 5.1.2, PECL pdo_pgsql:1.0.1-1.0.2)

PDO::pgsqlLOBUnlink — ラージオブジェクトを削除する

説明

bool **PDO::pgsqlLOBUnlink** (string $oid)

OID が指すラージオブジェクトをデータベースから削除します。

> **注意**: この関数は、トランザクション内でコールしなければなりません。

パラメータ

oid

ラージオブジェクトの ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 PDO::pgsqlLOBUnlink() の例

PDO::pgsqlLOBCreate() および **PDO::pgsqlLOBOpen()** の例で使用した blob テーブルからラージオブジェクトを参照している行を削除する前に、 この例ではラージオブジェクトをデータベースから削除します。

```
<?php
$db = new PDO('pgsql:dbname=test host=localhost', $user, $pass);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->beginTransaction();
$db->pgsqlLOBUnlink($oid);
$stmt = $db->prepare("DELETE FROM BLOBS where ident = ?");
$stmt->execute(array($some_id));
$db->commit();
?>
```

参考

- **PDO::pgsqlLOBOpen()**
- **PDO::pgsqlLOBCreate()**

目次

プリンタ関数

導入

以下の関数は、Windows 9.x, ME, NT4, 2000 でのみ利用可能です。これらの 関数は、PHP 4.0.4 で追加されました。

インストール手順

この » PECL 拡張モジュールは、 PHP にバンドルされていません。

Windows ユーザは、これらの関数を使用するには php.ini の中で php_printer.dll を有効にする必要があります。

実行時設定

php.ini の設定により動作が変化します。

プリンタ設定オプション

名前	デフォルト	変更の可否	変更履歴
printer.default_printer	""	PHP_INI_ALL	PHP 4.0.6 以降で使用可能です。PHP 4.1.1 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

この拡張モジュールでは、プリンタ接続や ブラシ、フォント、ペンへのハンドルを定義しています。

定義済み定数

定義済みの定数 を参照ください。

定数

(No version information available, might be only in CVS)

定数 — プリンタの定義済み定数

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

PRINTER_COPIES (integer)
PRINTER_MODE (integer)
PRINTER_TITLE (integer)
PRINTER_DEVICENAME (integer)
PRINTER_DRIVERVERSION (integer)
PRINTER_OUTPUT_FILE (integer)
PRINTER_RESOLUTION_Y (integer)
PRINTER_RESOLUTION_X (integer)
PRINTER_SCALE (integer)
PRINTER_BACKGROUND_COLOR (integer)
PRINTER_PAPER_LENGTH (integer)
PRINTER_PAPER_WIDTH (integer)
PRINTER_PAPER_FORMAT (integer)
PRINTER_FORMAT_CUSTOM (integer)
PRINTER_FORMAT_LETTER (integer)
PRINTER_FORMAT_LEGAL (integer)
PRINTER_FORMAT_A3 (integer)
PRINTER_FORMAT_A4 (integer)
PRINTER_FORMAT_A5 (integer)
PRINTER_FORMAT_B4 (integer)
PRINTER_FORMAT_B5 (integer)
PRINTER_FORMAT_FOLIO (integer)
PRINTER_ORIENTATION (integer)
PRINTER_ORIENTATION_PORTRAIT (integer)
PRINTER_ORIENTATION_LANDSCAPE (integer)
PRINTER_TEXT_COLOR (integer)
PRINTER_TEXT_ALIGN (integer)
PRINTER_TA_BASELINE (integer)
PRINTER_TA_BOTTOM (integer)
PRINTER_TA_TOP (integer)
PRINTER_TA_CENTER (integer)
PRINTER_TA_LEFT (integer)
PRINTER_TA_RIGHT (integer)
PRINTER_PEN_SOLID (integer)
PRINTER_PEN_DASH (integer)
PRINTER_PEN_DOT (integer)
PRINTER_PEN_DASHDOT (integer)
PRINTER_PEN_DASHDOTDOT (integer)
PRINTER_PEN_INVISIBLE (integer)
PRINTER_BRUSH_SOLID (integer)
PRINTER_BRUSH_CUSTOM (integer)
PRINTER_BRUSH_DIAGONAL (integer)
PRINTER_BRUSH_CROSS (integer)
PRINTER_BRUSH_DIAGCROSS (integer)
PRINTER_BRUSH_FDIAGONAL (integer)
PRINTER_BRUSH_HORIZONTAL (integer)

PRINTER_BRUSH_VERTICAL (integer)
PRINTER_FW_THIN (integer)
PRINTER_FW_ULTRALIGHT (integer)
PRINTER_FW_LIGHT (integer)
PRINTER_FW_NORMAL (integer)
PRINTER_FW_MEDIUM (integer)
PRINTER_FW_BOLD (integer)
PRINTER_FW_ULTRABOLD (integer)
PRINTER_FW_HEAVY (integer)
PRINTER_ENUM_LOCAL (integer)
PRINTER_ENUM_NAME (integer)
PRINTER_ENUM_SHARED (integer)
PRINTER_ENUM_DEFAULT (integer)
PRINTER_ENUM_CONNECTIONS (integer)
PRINTER_ENUM_NETWORK (integer)
PRINTER_ENUM_REMOTE (integer)

---

printer_abort

(No version information available, might be only in CVS)

printer_abort — プリンタのスプールファイルを削除する

説明

void **printer_abort** (resource $printer_handle)

この関数は、プリンタのスプールファイルを削除します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

返り値

値を返しません。

例

Example#1 printer_abort() の例

```
<?php
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
?>
```

---

printer_close

(No version information available, might be only in CVS)

printer_close — プリンタへの接続を閉じる

説明

void **printer_close** (resource $printer_handle)

この関数は、プリンタへの接続をクローズします。 **printer_close()** は、アクティブな デバイスコンテキストもクローズします。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

返り値

値を返しません。

例

Example#1 printer_close() の例

```
<?php
$handle = printer_open();
printer_close($handle);
?>
```

---

printer_create_brush

(No version information available, might be only in CVS)

printer_create_brush — 新規ブラシを作成する

説明

resource **printer_create_brush** (int $style , string $color)

この関数は、新しいブラシを作成してブラシへのハンドルを返します。 ブラシは、図形を塗りつぶすために使用されます。使用例は printer_select_brush() を参照ください。

パラメータ

style

style は、以下の定数のうちのいずれかで ある必要があります。

- **PRINTER_BRUSH_SOLID**: 実線のブラシを作成します。
- **PRINTER_BRUSH_DIAGONAL**: 45 度右上がりハッチ(/)のブラシを作成します。
- **PRINTER_BRUSH_CROSS**: クロスハッチ(+)のブラシを作成します。
- **PRINTER_BRUSH_DIAGCROSS**: 45 度回転したクロスハッチ(x)のブラシを作成します。
- **PRINTER_BRUSH_FDIAGONAL**: 45 度右下がりハッチ(\ )のブラシを作成します。
- **PRINTER_BRUSH_HORIZONTAL**: 水平ハッチ(-)のブラシを作成します。
- **PRINTER_BRUSH_VERTICAL**: 垂直ハッチ(|)のブラシを作成します。
- **PRINTER_BRUSH_CUSTOM**: BMP ファイルからカスタムブラシを作成します。2 番目のパラメータには RGB 色コードのかわりに BMP を指定します。

color

color は RGB 16 進フォーマットである必要があります。 たとえば黒は "000000" となります。

返り値

ブラシのハンドル、あるいはエラー時に **FALSE** を返します。

---

printer_create_dc

(No version information available, might be only in CVS)

printer_create_dc — 新規デバイスコンテキストを作成する

説明

void **printer_create_dc** (resource $printer_handle)

この関数は、新しいデバイスコンテキストを作成します。 デバイスコンテキストは、ドキュメントのグラフィックオブジェクトを カスタマイズするために使用されます。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

返り値

値を返しません。

例

Example#1 printer_create_dc() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* dc に関する操作を行います */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);

/* 別の dc を作成します */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* dc に関する操作を行います */

printer_delete_dc($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_create_font

(No version information available, might be only in CVS)

printer_create_font — 新規フォントを作成する

説明

resource **printer_create_font** (string $face , int $height , int $width , int $font_weight , bool $italic , bool $underline , bool $strikeout , int $orientation)

この関数は、新しいフォントを作成してそのハンドルを返します。作成したフォントは テキストの描画に使用されます。使用例は printer_select_font() を参照ください。

パラメータ

face

face はフォントフェースを指定する文字列である必要があります。

height

height でフォントの高さを指定します。

width

width でフォントの幅を指定します。

font_weight

font_weight ではフォントのウェイト (通常は 400)を指定し、以下の定数のうちのいずれかを使用することが可能です。

- **PRINTER_FW_THIN**: フォントウェイトを thin (100) に設定します。
- **PRINTER_FW_ULTRALIGHT**: フォントウェイトを ultra light (200) に設定します。
- **PRINTER_FW_LIGHT**: フォントウェイトを light (300) に設定します。
- **PRINTER_FW_NORMAL**: フォントウェイトを normal (400) に設定します。
- **PRINTER_FW_MEDIUM**: フォントウェイトを medium (500) に設定します。
- **PRINTER_FW_BOLD**: フォントウェイトを bold (700) に設定します。
- **PRINTER_FW_ULTRABOLD**: フォントウェイトを ultra bold (800) に設定します。
- **PRINTER_FW_HEAVY**: フォントウェイトをheavy (900) に設定します。

italic

italic には、 フォントを斜体にするかどうかを **TRUE** あるいは **FALSE** で指定します。

underline

underline には、 フォントに下線を引くかどうかを **TRUE** あるいは **FALSE** で指定します。

strikeout

strikeout には、 フォントに打ち消し線を引くかどうかを **TRUE** あるいは **FALSE** で指定します。

orientaton

orientation ではフォントの回転を指定します。

返り値

成功した場合にフォントのハンドル、エラー時に **FALSE** を返します。

---

printer_create_pen

(No version information available, might be only in CVS)

printer_create_pen — 新規ペンを作成する

説明

resource **printer_create_pen** (int $style , int $width , string $color)

この関数は、新しいペンを作成してそのハンドルを返します。作成したペンは、 直線や曲線を描く際に使用されます。使用例は printer_select_pen() を参照ください。

パラメータ

style

style は、以下の定数のうちのいずれかで ある必要があります。

- **PRINTER_PEN_SOLID**: 実線のペンを作成します。
- **PRINTER_PEN_DASH**: 破線のペンを作成します。
- **PRINTER_PEN_DOT**: 点線のペンを作成します。
- **PRINTER_PEN_DASHDOT**: ダッシュとドットで構成されるペンを作成します。
- **PRINTER_PEN_DASHDOTDOT**: ダッシュと 2 つのドットで構成されるペンを作成します。
- **PRINTER_PEN_INVISIBLE**: 不可視のペンを作成します。

width

width はペンの幅を指定しす。

color

color は RGB 16 進フォーマットである必要が あります。たとえば黒は "000000" となります。

返り値

*Returns a pen handle or **FALSE** on error.*

---

printer_delete_brush

(No version information available, might be only in CVS)

printer_delete_brush — ブラシを削除する

説明

*void **printer_delete_brush** (resource $brush_handle)*

この関数は、選択したブラシを削除します。使用例は printer_select_brush() を参照ください。

パラメータ

brush_handle

brush_handle は、 有効なブラシのハンドルである必要があります。

返り値

値を返しません。

---

printer_delete_dc

(No version information available, might be only in CVS)

printer_delete_dc — デバイスコンテキストを削除する

説明

*bool **printer_delete_dc** (resource $printer_handle)*

この関数はデバイスコンテキストを削除します。 使用例は printer_create_dc() を参照ください。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

---

printer_delete_font

(No version information available, might be only in CVS)

printer_delete_font — フォントを削除する

説明

*void **printer_delete_font** (resource $font_handle)*

この関数は選択したフォントを削除します。使用例は printer_select_font() を参照ください。

パラメータ

font_handle

font_handle は、 有効なフォントのハンドルである必要があります。

返り値

値を返しません。

---

printer_delete_pen

(No version information available, might be only in CVS)

printer_delete_pen — ペンを削除する

説明

*void **printer_delete_pen** (resource $pen_handle)*

この関数は、選択したペンを削除します。使用例は printer_select_pen() を参照ください。

パラメータ

pen_handle

pen_handle は、 有効なペンのハンドルである必要があります。

返り値

値を返しません。

---

printer_draw_bmp

(No version information available, might be only in CVS)

printer_draw_bmp — ビットマップを描画する

説明

bool **printer_draw_bmp** (resource $printer_handle , string $filename , int $x , int $y [, int $width], int $height)

この関数はビットマップを描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

filename

ビットマップへのパス。

x

x はビットマップの左上の x 座標です。

y

y はビットマップの左上の y 座標です。

width

ビットマップの幅。

height

ビットマップの高さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 printer_draw_bmp() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:¥¥image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_draw_chord

(No version information available, might be only in CVS)

printer_draw_chord — 弦を描画する

説明

void **printer_draw_chord** (resource $printer_handle , int $rec_x , int $rec_y , int $rec_x1 , int $rec_y1 , int $rad_x , int $rad_y , int $rad_x1 , int $rad_y1)

この関数は、弦を描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

rec_x

rec_x は、弦を囲む長方形の左上の *x* 座標です。

rec_y

rec_y は、弦を囲む長方形の左上の *y* 座標です。

rec_x1

rec_x1 は、弦を囲む長方形の右下の *x* 座標です。

rec_y1

rec_y1 は、弦を囲む長方形の右下の *y* 座標です。

rad_x

rad_x は、弦の開始位置を指定する半径の *x* 座標です。

rad_y

rad_y は、弦の開始位置を指定する半径の *y* 座標です。

rad_x1

rad_x1 は、弦の終了位置を指定する半径の *x* 座標です。

rad_y1

rad_y1 は、弦の終了位置を指定する半径の *y* 座標です。

返り値

値を返しません。

例

Example#1 printer_draw_chord() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_draw_elipse

(No version information available, might be only in CVS)

printer_draw_elipse — 楕円を描画する

説明

void ***printer_draw_elipse*** *(resource $printer_handle , int $ul_x , int $ul_y , int $lr_x , int $lr_y)*

この関数は、楕円を描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

ul_x

ul_x は、楕円の左上の *x* 座標です。

ul_y

ul_y は、楕円の左上の *y* 座標です。

lr_x

lr_x は、楕円の右下の *x* 座標です。

lr_y

lr_y は、楕円の右下の y 座標です。

返り値

値を返しません。

例

Example#1 printer_draw_elipse() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_draw_line

(No version information available, might be only in CVS)

printer_draw_line — 線を描画する

説明

void ***printer_draw_line*** *(resource $printer_handle , int $from_x , int $from_y , int $to_x , int $to_y)*

この関数は、選択したペンで直線を描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

from_x

from_x は開始点の x 座標です。

from_y

from_y は開始点の y 座標です。

to_x

to_x は終了点の x 座標です。

to_y

to_y は終了点の y 座標です。

返り値

値を返しません。

例

Example#1 printer_draw_line() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_pie

(No version information available, might be only in CVS)

printer_draw_pie — 円弧を描画する

説明

void **printer_draw_pie** (resource $printer_handle , int $rec_x , int $rec_y , int $rec_x1 , int $rec_y1 , int $rad1_x , int $rad1_y , int $rad2_x , int $rad2_y)

この関数は円弧を描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

rec_x

rec_x は、円弧を囲む長方形の左上の x 座標です。

rec_y

rec_y は、円弧を囲む長方形の左上の y 座標です。

rec_x1

rec_x1 は、円弧を囲む長方形の右下の x 座標です。

rec_y1

rec_y1 は、円弧を囲む長方形の右下の y 座標です。

rad1_x

rad1_x は、最初の半径の終了点の x 座標です。

rad1_y

rad1_y は、最初の半径の終了点の y 座標です。

rad2_x

rad2_x は、2 番目の半径の終了点の x 座標です。

rad2_y

rad2_y は、2 番目の半径の終了点の y 座標です。

返り値

値を返しません。

例

Example#1 printer_draw_pie() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

printer_draw_rectangle

(No version information available, might be only in CVS)

printer_draw_rectangle — 矩形を描画する

説明

void **printer_draw_rectangle** (resource $printer_handle , int $ul_x , int $ul_y , int $lr_x , int $lr_y)

この関数は矩形を描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

ul_x

ul_x は、矩形の左上の x 座標です。

ul_y

ul_y は、矩形の左上の y 座標です。

lr_x

lr_x は、矩形の右下の x 座標です。

lr_y

lr_y は、矩形の右下の y 座標です。

返り値

値を返しません。

例

Example#1 printer_draw_rectangle() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_draw_roundrect

(No version information available, might be only in CVS)

printer_draw_roundrect — 角が丸い矩形を描画する

説明

void **printer_draw_roundrect** (resource $printer_handle , int $ul_x , int $ul_y , int $lr_x , int $lr_y , int $width , int $height)

この関数は、角が丸い矩形を描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

ul_x

ul_x は、矩形の左上の x 座標です。

ul_y

ul_y は、矩形の左上の y 座標です。

lr_x

lr_x は、矩形の右下の x 座標です。

lr_y

lr_y は、矩形の右下の y 座標です。

width

width は、楕円の幅です。

height

height は、楕円の高さです。

返り値

値を返しません。

例

Example#1 printer_draw_roundrect() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_draw_text

(No version information available, might be only in CVS)

printer_draw_text — テキストを描画する

説明

void **printer_draw_text** (resource $printer_handle , string $text , int $x , int $y)

この関数は、選択したフォントを使用して位置 x , y に text を描画します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

text

描画したいテキスト。

x

x は描画する位置の x 座標です。

y

y は描画する位置の y 座標です。

返り値

値を返しません。

例

Example#1 printer_draw_text() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 72, 48, 400, false, false, false, 0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_end_doc

(No version information available, might be only in CVS)

printer_end_doc — ドキュメントを閉じる

説明

bool **printer_end_doc** (resource $printer_handle)

プリンタスプーラ内の新規ドキュメントを閉じます。 これにより、ドキュメントの印刷準備が完了します。使用例は printer_start_doc() を参照ください。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

printer_end_page

(No version information available, might be only in CVS)

printer_end_page — アクティブなページを閉じる

説明

bool **printer_end_page** (resource $printer_handle)

この関数は、アクティブなドキュメントのアクティブなページを閉じます。 使用例は printer_start_doc() を参照ください。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

printer_get_option

(No version information available, might be only in CVS)

printer_get_option — プリンタ設定データを取得する

説明

mixed **printer_get_option** (resource $printer_handle , string $option)

この関数は、option の設定値を取得します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

option

取得可能な設定については printer_set_option() を参照ください。それに加えてさらに以下の設定も取得可能です。

- **PRINTER_DEVICENAME** プリンタのデバイス名を返します。
- **PRINTER_DRIVERVERSION** プリンタドライバのバージョンを返します。

返り値

option の値を返します。

例

Example#1 printer_get_option() の例

```
<?php
$handle = printer_open();
echo printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
?>
```

---

printer_list

(No version information available, might be only in CVS)

printer_list — サーバに登録されたプリンタの配列を返す

説明

array **printer_list** (int $enumtype [, string $name [, int $level]])

この関数は、利用可能なプリンタとその機能を調べます。

パラメータ

enumtype

enumtype は次の定義済みの定数のどれかとする必要があります。

- **PRINTER_ENUM_LOCAL**: ローカルにインストールされたプリンタを列挙します。
- **PRINTER_ENUM_NAME**: name のサーバ、 ドメインまたはプリントプロバイダになることができるものを列挙します。
- **PRINTER_ENUM_SHARED**: このパラメータは単独では使用できず、 他のパラメータとともに使用する必要があります。 たとえば PRINTER_ENUM_LOCAL とともに使用することで、 ローカルの共有プリンタを検出します。
- **PRINTER_ENUM_DEFAULT**: (Win9.x のみ) デフォルトのプリンタを列挙します。
- **PRINTER_ENUM_CONNECTIONS**: (WinNT/2000 のみ) ユーザが接続済みのプリンタを列挙します。
- **PRINTER_ENUM_NETWORK**: (WinNT/2000 のみ) コンピュータのドメインにあるネットワークプリンタを列挙します。 level が 1 の場合のみ有効です。
- **PRINTER_ENUM_REMOTE**: (WinNT/2000 のみ) コンピュータのドメインにある ネットワークプリンタとプリンタサーバを列挙します。 level が 1 の場合のみ有効です。

name

PRINTER_ENUM_NAME とともに使用します。

level

level には、情報要求のレベルを設定します。 これは、1,2,4 または 5 とすることが可能です、

返り値

プリンタの配列を返します。

例

Example#1 printer_list() の例

```
<?php
/* ローカルな共有プリンタを検出 */
var_dump(printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED));
?>
```

---

printer_logical_fontheight

(No version information available, might be only in CVS)

printer_logical_fontheight — 論理フォントの高さを取得する

説明

int **printer_logical_fontheight** (resource $printer_handle , int $height)

この関数は height の論理フォントの高さを計算します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

height

フォントの高さ。

返り値

論理フォントの高さ、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 printer_logical_fontheight() の例

```
<?php
$handle = printer_open();
echo printer_logical_fontheight($handle, 72);
printer_close($handle);
?>
```

---

printer_open

(No version information available, might be only in CVS)

printer_open — プリンタへの接続をオープンする

説明

resource **printer_open** ([string $printername])

この関数は、指定したプリンタへの接続を試みます。

printer_open() は、 デバイスコンテキストも開始します。

パラメータ

printername

プリンタ名。このパラメータが指定されない場合、 この関数はデフォルトのプリンタへの接続をオープンします(php.ini に printer.default_printer として指定されていない場合は、 PHP はデフォルトのプリンタの検出を試みます)。

返り値

成功時にハンドル、失敗時に **FALSE** を返します。

例

Example#1 printer_open() の例

```
<?php
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
?>
```

---

printer_select_brush

(No version information available, might be only in CVS)

printer_select_brush — ブラシを選択する

説明

void **printer_select_brush** (resource $printer_handle , resource $brush_handle)

この関数は、実際のデバイスコンテキストのアクティブな描画オブジェクトと してブラシを選択します。矩形を描画する場合、図形の描画には ブラシが使用され、輪郭の描画にはペンが使用されます。

図形を描く前にブラシが選択されていない場合、その図形は塗りつぶされません。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

brush_handle

brush_handle は、 ブラシの有効なハンドルである必要があります。

返り値

値を返しません。

例

Example#1 printer_select_brush() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:¥¥brush.bmp");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1, 501, 500, 1001);
printer_delete_brush($brush);

printer_delete_pen($pen);

printer_end_page($handle);
```

```
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_select_font

(No version information available, might be only in CVS)

printer_select_font — フォントを選択する

説明

void **printer_select_font** (resource $printer_handle , resource $font_handle)

この関数は、テキストを描画するフォントを選択します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

font_handle

font_handle は、フォントの 有効なハンドルである必要があります。

返り値

値を返しません。

例

Example#1 printer_select_font() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -50);
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_select_pen

(No version information available, might be only in CVS)

printer_select_pen — ペンを選択する

説明

void **printer_select_pen** (resource $printer_handle , resource $pen_handle)

この関数は、実際のデバイスコンテキストのアクティブな描画オブジェクトと してペンを選択します。ペンは、直線や曲線の描画に使用されます。たとえば 一本の直線を引く際にはペンが使用されます。矩形を描く際には、その輪郭を ペンが描画し、ブラシがその図形を塗りつぶします。 図形を描く前にペンが選択されていない場合、その図形の輪郭は描かれません。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

pen_handle

pen_handle は、 ペンの有効なハンドルである必要があります。

返り値

値を返しません。

例

Example#1 printer_select_pen() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
```

```
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_set_option

(No version information available, might be only in CVS)

printer_set_option — プリンタの接続を設定する

説明

bool **printer_set_option** (resource $printer_handle , int $option , mixed $value)

この関数は、現在の接続のオプションを設定します。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

option

オプションに指定できるのは、以下の定数のいずれかです。

- **PRINTER_COPIES**: 印刷する部数を指定します。value は整数値である必要があります。
- **PRINTER_MODE**: データの種類 (text、raw あるいは emf) を指定します。 value は文字列である必要があります。
- **PRINTER_TITLE**: 文書の名前を指定します。value は文字列である必要があります。
- **PRINTER_ORIENTATION**: 用紙の向きを指定します。value は PRINTER_ORIENTATION_PORTRAIT あるいは PRINTER_ORIENTATION_LANDSCAPE のいずれかです。
- **PRINTER_RESOLUTION_Y**: y 方向の解像度を DPI で指定します。value は整数値である必要があります。
- **PRINTER_RESOLUTION_X**: x 方向の解像度を DPI で指定します。value は整数値である必要があります。
- **PRINTER_PAPER_FORMAT**: 定義済みの用紙フォーマットを指定します。 PRINTER_PAPER_WIDTH および PRINTER_PAPER_LENGTH を使用して独自の用紙を指定したい場合には、 value には PRINTER_FORMAT_CUSTOM を指定します。 value は、以下の定数のいずれかです。
  - **PRINTER_FORMAT_CUSTOM**: 独自のフォーマットを指定します。
  - **PRINTER_FORMAT_LETTER**: 標準 letter フォーマット (8 1/2 インチ x 11 インチ) を指定します。
  - **PRINTER_FORMAT_LEGAL**: 標準 legal フォーマット (8 1/2 インチ x 14 インチ) を指定します。
  - **PRINTER_FORMAT_A3**: 標準 A3 フォーマット (297 mm x 420 mm) を指定します。
  - **PRINTER_FORMAT_A4**: 標準 A4 フォーマット (210 mm x 297 mm) を指定します。
  - **PRINTER_FORMAT_A5**: 標準 A5 フォーマット (148 mm x 210 mm) を指定します。
  - **PRINTER_FORMAT_B4**: 標準 B4 フォーマット (250 mm x 354 mm) を指定します。
  - **PRINTER_FORMAT_B5**: 標準 B5 フォーマット (182 mm x 257 mm) を指定します。
  - **PRINTER_FORMAT_FOLIO**: 標準 FOLIO フォーマット (8 1/2 インチ x 13 インチ) を指定します。
- **PRINTER_PAPER_LENGTH**: PRINTER_PAPER_FORMAT が PRINTER_FORMAT_CUSTOM に指定されている場合、 ミリ単位での用紙の長さを PRINTER_PAPER_LENGTH で指定します。 value は整数値である必要があります。
- **PRINTER_PAPER_WIDTH**: PRINTER_PAPER_FORMAT が PRINTER_FORMAT_CUSTOM に指定されている場合、 ミリ単位での用紙の幅を PRINTER_PAPER_WIDTH で指定します。 value は整数値である必要があります。
- **PRINTER_SCALE**: 印刷時の倍率を指定します。実際の大きさに scale/100 を掛けた大きさでページが印刷されます。例えば、値を 50 に設定した場合には、出力内容は実際の大きさの半分となります。 value は整数値である必要があります。
- **PRINTER_BACKGROUND_COLOR**: デバイスコンテキストの背景色を指定します。 value は rgb 情報を十六進表現で表した 文字列である必要があります。つまり、"005533" のようになります。
- **PRINTER_TEXT_COLOR**: デバイスコンテキストの文字色を指定します。 value は rgb 情報を十六進表現で表した 文字列である必要があります。つまり、"005533" のようになります。
- **PRINTER_TEXT_ALIGN**: デバイスコンテキストのテキストの配置を指定します。 value には、 以下の定数を ORで組み合わせて指定することができます。
  - **PRINTER_TA_BASELINE**: テキストをベースライン上に配置します。
  - **PRINTER_TA_BOTTOM**: テキストを下詰めにします。
  - **PRINTER_TA_TOP**: テキストを上詰めにします。
  - **PRINTER_TA_CENTER**: テキストを中央揃えにします。
  - **PRINTER_TA_LEFT**: テキストを左詰めにします。
  - **PRINTER_TA_RIGHT**: テキストを右詰めにします。

value

option の値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 printer_set_option() の例

```
<?php
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
?>
```

---

printer_start_doc

(No version information available, might be only in CVS)

printer_start_doc — 新規ドキュメントを開始する

説明

bool **printer_start_doc** (resource $printer_handle [, string $document])

この関数は、プリンタスプーラに新しいドキュメントを作成します。 ドキュメントには複数のページを含めることが可能で、スプーラの中で 印刷ジョブを予約するために使用されます。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

document

オプションのパラメータ document は、ドキュメントの別名を指定するために使用可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 printer_start_doc() の例

```
<?php
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
?>
```

---

printer_start_page

(No version information available, might be only in CVS)

printer_start_page — 新規ページを開始する

説明

bool **printer_start_page** (resource $printer_handle)

この関数は、アクティブなドキュメントに新しいページを作成します。 使用例は printer_start_doc() を参照ください。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

printer_write

(No version information available, might be only in CVS)

printer_write — プリンタへデータを書き込む

説明

bool **printer_write** (resource $printer_handle , string $content)

content を直接プリンタに書き込みます。

パラメータ

printer_handle

printer_handle には、プリンタへの有効なハンドルを指定する必要があります。

content

書き込むデータ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 printer_write() の例

```
<?php
$handle = printer_open();
printer_write($handle, "Text to print");
printer_close($handle);
?>
```

目次

プログラム実行関数

導入

以下の関数は、システム自体の上でコマンドを実行したり、 こうしたコマンドの安全に実行する手段を提供します。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールが定義するリソースは process リソースで、これは proc_open() が返します。

定義済み定数

定数は定義されていません。

注意

警告

ロックしてオープンしたファイル（特に、オープンしたセッション）は、プログラムをバックグラウンドで実行する前に閉じておく必要があります。

参考

以下の関数は、 バックティック演算子 にも関係します。 また、セーフモード においては、 safe_mode_exec_dir ディレクティブの使用を考慮する必要があります。

---

escapeshellarg

(PHP 4 >= 4.0.3, PHP 5)

escapeshellarg — シェル引数として使用される文字列をエスケープする

説明

string **escapeshellarg** (string $arg)

escapeshellarg() は、文字列をシングルクオート で括り、既存のシングルクオートを全てクオート/エスケープします。こ れにより、文字列を直接シェル関数に渡し、単一の安全な引数として処 理することを可能にします。この関数は、ユーザー入力からの入力を シェル関数への引数として渡す際にエスケープするために使用する必要 があります。シェル関数には、exec(), system()そして バックティック演算子 を含むシェル関数が含まれます。

パラメータ

arg

エスケープされる引数

返り値

エスケープされた文字列

例

Example#1 escapeshellarg() の例

```
<?php
system('ls '.escapeshellarg($dir));
?>
```

参考

- escapeshellcmd()
- exec()
- popen()
- system()
- バックティック演算子

---

escapeshellcmd

(PHP 4, PHP 5)

escapeshellcmd — シェルのメタ文字をエスケープする

説明

string **escapeshellcmd** (string $command)

escapeshellcmd() は、文字列中においてシェルコマンドを だまして勝手なコマンドを実行する可能性がある文字をエスケープします。 この関数は、ユーザに入力されたデータを関数 exec() または system() または、 バックティック演算子 に渡す前に全てエスケープを行う場合に使用するべきです。

#&;`|*?~<>^()[]{}$¥、¥x0A および ¥xFF については、その文字の前にバックスラッシュが 追加されます。' および " は、対になっていない場合にのみエスケープされます。Windows では、 これらの文字に加えて % がスペースに置き換えられます。

パラメータ

command

エスケープされるコマンド

返り値

エスケープされた文字列

例

Example#1 escapeshellcmd() の例

```
<?php
$e = escapeshellcmd($userinput);

// ここでは $e がスペースを含んでいても関係ない
system("echo $e");
$f = escapeshellcmd($filename);

// ここでは気を遣い、クォートを使用する
system("touch ¥"/tmp/$f¥"; ls -l ¥"/tmp/$f¥"");
?>
```

参考

- escapeshellarg()
- exec()
- popen()
- system()
- バックティック演算子

---

exec

(PHP 4, PHP 5)

exec — 外部プログラムを実行する

説明

```
string exec ( string $command [, array &$output [, int &$return_var ]] )
```

exec() は指定されたコマンド command を実行します。

パラメータ

command

実行するコマンド

output

引数 output が存在する場合、指定した配列は、 コマンドからの出力の各行で埋められます。 ¥n のような後に続く空白は、この配列には含まれません。 配列に既に何らかの要素が 含まれる場合は、**exec()** は配列の最後に追加される ことに注意してください。関数が要素を追加することを望まないのなら、 それが **exec()** に渡される前に、配列の unset() を呼び出してください。

return_var

引数return_var が、引数 output と共に存在する場合、実行したコマンドの ステータスがこの変数に書かれます。

返り値

コマンド結果の最後の行を返します。コマンドを実行し、 一切干渉を受けずに直接コマンドから全てのデータを受けとる必要が あるならば、PassThru() 関数を使ってください。

実行されたコマンドの出力を取得するには、必ず output パラメータを設定・使用してください。

例

Example#1 exec() の例

```
<?php
// ("whoami" コマンドをパスに有するシステム上で)
// 実行中のphp/httpdプロセスを所有するユーザの名前を出力
echo exec('whoami');
?>
```

注意

警告

ユーザが入力したデータをこの関数に渡すことを許可する場合、ユーザが任意のコマンドを実行できるようシステムを欺くことができないように escapeshellarg() または escapeshellcmd() を適用する必要があります。

> **注意**: この関数を用いてプログラムを開始し、バックグラウンドで実行させたままにしたい場合、このプログラムの出力をファイルまたは他の出力ストリームにリダイレクトするようにする必要があります。さもないと、 PHP はプログラム実行終了までにハングしてしまいます。
>
> **注意**: セーフモード が有効な場合、safe_mode_exec_dirの中にある実行プログラムのみ実行可能です。実際的な理由により、現在、実行プログラムへのパスに ..を含めることはできません。

警告

セーフモードが有効な場合、コマンド文字列は escapeshellcmd() でエスケープされます。つまり、echo y | echo x は、echo y ¥| echo x となります。

参考

- system()
- passthru()
- escapeshellcmd()
- pcntl_exec()
- バックティック演算子

---

passthru

(PHP 4, PHP 5)

passthru — 外部プログラムを実行し、未整形の出力を表示する

説明

void **passthru** (string $command [, int &$return_var])

passthru()関数はexec()関数と同様、 command を実行します。 引数 return_var を指定した場合、 Unix コマンドのステータスで置換されます。 この関数は Unix コマンドからの出力がバイナリデータであり、 ブラウザーへ直接返す必要がある場合、exec() もしくはsystem()の代わりに使用する必要があります。 よく使うのは、直接画像ストリームを出力することができる pbmplus ユーティリティの様なものを実行する場合です。 content-type を image/gifに設定して、gifを出力するpbmplus プログラムを呼び出すことにより、直接画像を出力する PHP スクリプトを作成することができます。

パラメータ

command

実行するコマンド

return_var

引数return_var が存在する場合、 Unix コマンドのステータスがこの変数に書かれます。

返り値

値を返しません。

注意

警告

ユーザが入力したデータをこの関数に渡すことを許可する場合、ユーザが任意のコマンドを実行できるようシステムを欺くことができないように escapeshellarg() または escapeshellcmd() を適用する必要があります。

> **注意**: この関数を用いてプログラムを開始し、バックグラウンドで実行させたままにしたい場合、このプログラムの出力をファイルまたは他の出力ストリームにリダイレクトするようにする必要があります。さもないと、 PHP はプログラム実行終了までにハングしてしまいます。
>
> **注意**: セーフモード が有効な場合、 safe_mode_exec_dirの中にある実行プログラムのみ実行可能です。実際的な理由により、現在、実行プログラムへのパスに ..を含めることはできません。

警告

セーフモードが有効な場合、コマンド文字列は escapeshellcmd() でエスケープされます。つまり、echo y | echo x は、echo y ¥| echo x となります。

参考

- exec()
- system()
- popen()
- escapeshellcmd()
- バックティック演算子

---

proc_close

(PHP 4 >= 4.3.0, PHP 5)

proc_close — proc_open() で開かれたプロセスを閉じ、 そのプロセスの終了コードを返す

説明

int **proc_close** (resource $process)

proc_close() は pclose() と似ていますが、proc_open() で開かれたプロセスに対してのみ機能するという点で異なります。 **proc_close()** は、プロセスが終了するまで待った後で、 終了コードを返します。もし、そのプロセスに対してパイプが開かれていた場合は、 デッドロックを避けるため、fclose() 関数で、 この関数を呼び出す前にそれらを閉じておかなくてはなりません - パイプが開いている間、子プロセスは終了できないからです。

パラメータ

process

閉じられる proc_open() リソース

返り値

実行していたプロセスの終了状態を返します。

---

proc_get_status

(PHP 5)

proc_get_status — proc_open() で開かれたプロセスに関する情報を取得する

説明

*array **proc_get_status** (resource $process)*

***proc_get_status()** は、proc_open() で開かれたプロセスに関する情報を取得します。*

パラメータ

process

評価される proc_open() リソース

返り値

*成功時は集められた情報の配列、失敗時は **FALSE** 。 返される配列は次のような要素を持ちます:*

要素	型	説明
command	string	proc_open() に指定されたコマンド文字列。
pid	int	プロセス ID
running	bool	もしプロセスがまだ動いている場合は、**TRUE** 、すでに終了している場合は **FALSE**。
signaled	bool	子プロセスが、キャッチされていないシグナルにより終了した場合に **TRUE** となります。Windows では常に **FALSE** にセットされます。
stopped	bool	子プロセスが、シグナルにより停止した時に **TRUE** となります。 Windows では常に **FALSE** にセットされます。
exitcode	int	プロセスが返した終了コード (*running* が **FALSE** の時のみ意味を持ちます)。 正しい値を返すのは関数を最初にコールした時のみで、次に コールした際には *-1* を返します。
termsig	int	プロセスを終了させたシグナルの番号です (*signaled* が **TRUE** の時のみ意味を持ちます)。
stopsig	int	プロセスを停止させたシグナルの番号です (*stopped* が **TRUE** の時のみ意味を持ちます)。

参考

- *proc_open()*

---

proc_nice

(PHP 5)

proc_nice — 現在のプロセスの優先度を変更する

説明

*bool **proc_nice** (int $increment)*

***proc_nice()** は、現在のプロセスの優先度を increment で指定された値に変更します。 increment が正数の場合、 現在のプロセスの優先度をより低くし、increment が負数の場合は優先度が上がります。*

***proc_nice()** は、proc_open() やそれに関連する関数とは関係ありません。*

パラメータ

increment

変更する優先度の増加値

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 ユーザーが優先度を変更する権限を持っていないなど、 エラーが発生した場合は **E_WARNING** レベルのエラーも発行されます。*

注意

> ***注意**: **可用性** **proc_nice()** は、使用しているシステムが 'nice' の機能を持っている場合のみ利用可能です。 'nice' は次のシステムに準拠しています: SVr4, SVID EXT, AT&T, X/OPEN, BSD 4.3 。 これは Windows では **proc_nice()** を利用できないことを意味します。*

---

proc_open

(PHP 4 >= 4.3.0, PHP 5)

proc_open — コマンドを実行し、入出力用にファイルポインタを開く

説明

resource **proc_open** (string $cmd , array $descriptorspec , array &$pipes [, string $cwd [, array $env [, array $other_options]]])

proc_open() は popen() と よく似ていますが、プログラムの実行をさらに細かく制御できる点で違います。

パラメータ

cmd

実行されるコマンド。

descriptorspec

数値添字の配列で、ディスクリプタ番号をキーとし、PHP がその ディスクリプタをどのように子プロセスに渡すかを表すのが 対応する値となります。 0 が標準入力 (stdin)、1 が標準出力 (stdout) で、 2 が標準エラー出力 (stderr) となります。

現在サポートされているパイプ形式は file および pipe です。

ファイルディスクリプタの番号は、特に 0, 1, 2 に限られているわけでは ありません。有効であるどのようなファイルディスクリプタの番号も指定でき、 それは子プロセスに渡されます。これにより、あるスクリプトと、 子プロセスとして起動している別のスクリプトとの間で通信ができます。特に、これは PGP や GPG、openssl といったプログラムにパスフレーズを より安全な方法で渡したいとき威力を発揮します。 補助的なファイルディスクリプタを介して、そのようなプログラムの 状態を取得するのにも便利です。

pipes

PHP 側で生成されたパイプの終端にあたる ファイルポインタの配列。

cwd

コマンドの初期作業ディレクトリ。 完全パスである必要があります。 デフォルト値（現在の PHP プロセスの作業ディレクトリ）を使用したい場合は **NULL** を指定します。

env

実行するコマンドのための環境変数の配列。 現在の PHP プロセスと同じ環境変数を使用する場合は **NULL** を指定します。

other_options

その他の追加オプションを指定することが可能です。 現在サポートされているオプションは次の通りです。

- suppress_errors (windows のみ): **TRUE** にすると、この関数が出力するエラーを抑制します。
- bypass_shell (windows のみ): **TRUE** にすると、cmd.exe シェルをバイパスします。
- context: ファイルをオープンする際に (stream_context_create() で作成した) ストリームコンテキストを使用します。
- binary_pipes: パイプをバイナリモードでオープンします。 通常の stream_encoding は使用しません。

返り値

プロセスを表すリソースを返します。このリソースは、使用し終えた際に proc_close() を使用して開放する必要があります。 失敗した場合は **FALSE** を返します。

変更履歴

バージョン	説明
6.0.0	other_options パラメータに オプション context および binary_pipes が追加されました。
5.2.1	other_options パラメータに オプション bypass_shell が追加されました。
5.0.0	パラメータ cwd 、env および other_options が追加されました。

例

Example#1 A proc_open() の例

```
<?php
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin は、子プロセスが読み込むパイプです。
   1 => array("pipe", "w"),  // stdout は、子プロセスが書き込むパイプです。
   2 => array("file", "/tmp/error-output.txt", "a") //  はファイルで、そこに書き込みます。
);

$cwd = '/tmp';
$env = array('some_option' => 'aeiou');

$process = proc_open('php', $descriptorspec, $pipes, $cwd, $env);

if (is_resource($process)) {
    // $pipes はこの時点で次のような形を取っています。
     // 0 => 子プロセスの stdin に繋がれた書き込み可能なハンドル
     // 1 => 子プロセスの stdout に繋がれた読み込み可能なハンドル
      // すべてのエラー出力は /tmp/error-output.txt に書き込みされます。

    fwrite($pipes[0], '<?php print_r($_ENV); ?>');
    fclose($pipes[0]);

    echo stream_get_contents($pipes[1]);
```

```
    fclose($pipes[1]);

    // デッドロックを避けるため、proc_close を呼ぶ前に
    // すべてのパイプを閉じることが重要です。
    $return_value = proc_close($process);

    echo "command returned $return_value¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [some_option] => aeiou
    [PWD] => /tmp
    [SHLVL] => 1
    [_] => /usr/local/bin/php
)
command returned 0
```

注意

注意: Windows における互換性: 2 (stderr) よりも大きな番号のディスクリプタは 子プロセスに継承可能なハンドルとして渡されますが、 Windows のアーキテクチャは、ファイルディスクリプタの番号と より低レベルなハンドルを関連付けないので、子プロセスは、それらのハンドルにアクセスする術を持ちません。stdin, stdout, stderr は期待通り動きます。

注意: もし単方向(一方向)のパイプを利用したいだけでしたら、 popen() を使うほうがより簡単です。

参考

- popen()
- exec()
- system()
- passthru()
- stream_select()
- バックティック演算子

---

proc_terminate

(PHP 5)

proc_terminate — proc_open でオープンされたプロセスを強制終了する

説明

bool **proc_terminate** (resource $process [, int $signal])

終了させる (proc_open() を用いて生成された) process にシグナルを送信します。 **proc_terminate()** は即座に返され、 プロセスの終了を待ちません。

proc_terminate() により、プロセスを終了させ、 他のタスクを継続することができます。proc_get_status() 関数を使用して (停止したかどうかを確認するために) プロセスをポーリングすることができます。 しかし、これは PHP 5.2.2 以降でしか行えません。それより前のバージョンでは指定したプロセス resource を破棄していました。

パラメータ

process

終了させる proc_open() リソース。

signal

このオプションパラメータは POSIX オペレーティングシステムでのみ有用です。 kill(2) システムコールを使用して、 プロセスに送信するシグナルを指定することができます。 デフォルトは SIGTERM です。

返り値

実行していたプロセスの終了状態を返します。

参考

- proc_open()
- proc_close()
- proc_get_status()

---

shell_exec

(PHP 4, PHP 5)

shell_exec — シェルによりコマンドを実行し、文字列として出力全体を返す

説明

string **shell_exec** (string $cmd)

この関数は バックティック演算子 と等価です。

パラメータ

cmd

実行するコマンド

返り値

実行されたコマンドからの出力

例

Example#1 shell_exec() の例

```
<?php
$output = shell_exec('ls -lart');
echo "<pre>$output</pre>";
?>
```

注意

> **注意**: この関数は、safe-mode では無効となります。

参考

- exec()
- escapeshellcmd()

---

system

(PHP 4, PHP 5)

system — 外部プログラムを実行し、出力を表示する

説明

string **system** (string $command [, int &$return_var])

system() は、指定した command を実行し、結果を出力する C 言語の system 関数に似ています。

PHP をサーバモジュールとして実行している場合、 **system()** のコールにより、各行を出力した後、 Web サーバの出力バッファが自動的にクリアされます。

コマンドを実行し、何の加工もせずに全てのデータをコマンドから直接 返す必要がある場合、passthru() 関数を使用してください。

パラメータ

command

実行するコマンド。

return_var

引数 return_var が存在する場合、 実行したコマンドの返すステータスがこの変数に書かれます。

返り値

成功時はコマンド出力の最後の行を返し、失敗時は **FALSE** を返します。

例

Example#1 system() の例

```
<?php
echo '<pre>';

// シェルコマンド "ls" の全ての結果を出力し、出力の最後の
// 行を $last_line に格納します。シェルコマンドの戻り値は
// $retval に格納されます。
$last_line = system('ls', $retval);

// 追加情報を表示します。
echo '
</pre>
<hr />Last line of the output: ' . $last_line . '
<hr />Return value: ' . $retval;
?>
```

注意

警告

ユーザが入力したデータをこの関数に渡すことを許可する場合、ユーザが任意のコマンドを実行できるようシステムを欺くことができないように escapeshellarg() または escapeshellcmd() を適用する必要があります。

> **注意**: この関数を用いてプログラムを開始し、バックグラウンドで実行させたままにしたい場合、このプログラムの出力をファイルまたは他の出力ストリームにリダイレクトするようにする必要があります。さもないと、 PHP はプログラム実行終了までにハングしてしまいます。
>
> **注意**: セーフモード が有効な場合、safe_mode_exec_dirの中にある実行プログラムのみ実行可能です。実際的な理由により、現在、実行プログラムへのパスに ..を含めることはできません。

警告

セーフモードが有効な場合、コマンド文字列は escapeshellcmd() でエスケープされます。つまり、echo y | echo x は、echo y ¥| echo x となります。

参考

- exec()
- passthru()
- popen()
- escapeshellcmd()
- pcntl_exec()
- バックティック演算子

目次

PostScript ドキュメントの作成

導入

このモジュールは、PostScript ドキュメントを作成するためのものです。 PDF 拡張と多くの共通点を持っています。実際のところ API はほぼ同じで、 多くは関数名の最初の pdf_ を ps_ に変えるだけで使えます。 PostScript ドキュメントには直接関係のない機能（ハイパーリンクの追加など） も含まれていますが、これらは PostScript ドキュメントを PDF に変換した場合に有効となります。

この拡張によって作られたドキュメントは、いくつかの点で pdf 拡張に よって作られたドキュメントより優れています。なぜなら pslib のレンダリング関数はカーニングやハイフネーション、そしてリゲチャに対応しており、 よりよい出力が得られるからです。

要件

少なくとも PHP 4.3.0 以降と pslib >= 0.1.12 が必要です。 ps ライブラリ（pslib）は、 » http://pslib.sourceforge.net/ にあります。

インストール手順

簡単なインストール方法: コンソールで以下のコマンドを入力します。

```
$ pecl install ps
```

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールでは PostScript ドキュメントリソースを定義しています。これは ps_new() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

以下の 2 つの表は、ps 拡張で定義されている定数の一覧です。

線端の形状をあらわす定数

名前	意味
ps_LINECAP_BUTT	
ps_LINECAP_ROUND	
ps_LINECAP_SQUARED	

線の連結方法をあらわす定数

名前	意味
ps_LINEJOIN_MITER	
ps_LINEJOIN_ROUND	
ps_LINEJOIN_BEVEL	

連絡先

もしこの拡張や pslib についての意見・バグフィックス・機能拡張などがあれば、 私 » steinm@php.net にメールをください。お待ちしています。

---

ps_add_bookmark

(PECL ps:1.1.0-1.3.5)

ps_add_bookmark — 現在のページにブックマークを追加する

説明

int **ps_add_bookmark** (resource $psdoc , string $text [, int $parent [, int $open]])

現在のページにブックマークを追加します。たいてい、ブックマークは PDFビューア上でページの左側にツリー形式で表示されます。 ブックマークをクリックすると、該当ページにジャンプします。

ドキュメントを印刷したり表示したりする場合にはブックマークは何の意味も 持ちません。ブックマークが使われるのは、Acrobat Distiller™ や Ghostview を用いてドキュメントを PDF に変換した場合です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

ブックマークの表示に使用するテキスト。

parent

新しいブックマークの親となる、 事前にこの関数で作成されたブックマーク。

open

open がゼロでない場合、ブックマークは PDF ビューア上で開いた形で表示されます。

返り値

返り値はブックマークへの参照です。この値は、作成したブックマークを 他のブックマークの親として使用する場合にのみ使われます。 成功した場合には、この値はゼロより大きくなります。 エラーが発生した場合にはゼロを返します。

参考

- ps_add_launchlink()
- ps_add_pdflink()
- ps_add_weblink()

---

ps_add_launchlink

(PECL ps:1.1.0-1.3.5)

ps_add_launchlink — ファイルを実行するためのリンクを追加する

説明

bool **ps_add_launchlink** (resource $psdoc , float $llx , float $lly , float $urx , float $ury , string $filename)

クリックした際にファイルが実行されるハイパーリンクを、 指定した場所に設定します。ハイパーリンクのリンク元は矩形で表され、 その左下の座標は (llx, lly)、そして右上の座標が (urx, ury) となります。 この矩形はデフォルトでは細い青線で表されます。

ドキュメントを印刷したり表示したりする場合には、 ハイパーリンクは見えません。ハイパーリンクが現れるのは、Acrobat Distiller™ や Ghostview を用いてドキュメントを PDF に変換した場合です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

llx

左下角の x 座標。

lly

左下角の y 座標。

urx

右上角の x 座標。

ury

右上角の y 座標。

filename

リンクがクリックされた際に開始するプログラムのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ps_add_locallink()*
- *ps_add_pdflink()*
- *ps_add_weblink()*

---

ps_add_locallink

(PECL ps:1.1.0-1.3.5)

ps_add_locallink — 同一ドキュメント内のページへのリンクを追加する

説明

bool ***ps_add_locallink*** *(resource $psdoc , float $llx , float $lly , float $urx , float $ury , int $page , string $dest)*

指定した位置に、同一ドキュメント内のページへのハイパーリンクを設定します。 リンクをクリックすると、そのページに移動します。ドキュメントの最初の ページのページ番号は 1 です。

ハイパーリンクのリンク元は矩形で表され、その左下の座標は (llx , lly)、 そして右上の座標が (urx , ury) となります。この矩形はデフォルトでは細い青線で表されます。

ドキュメントを印刷したり表示したりする場合には、 ハイパーリンクは見えません。ハイパーリンクが現れるのは、Acrobat Distiller™ や Ghostview を用いてドキュメントを PDF に変換した場合です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

llx

左下角の x 座標。

lly

左下角の y 座標。

urx

右上角の x 座標。

ury

右上角の y 座標。

page

リンクがクリックされた際に開始するページ。

dest

dest は、ドキュメントがどのように 表示されるかを指定します。とりうる値は fitpage または fitwidth、 fitheight、fitbbox のいずれかです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_add_launchlink()
- ps_add_pdflink()
- ps_add_weblink()

---

ps_add_note

(PECL ps:1.1.0-1.3.5)

ps_add_note — 現在のページに注釈を追加する

説明

bool **ps_add_note** (resource $psdoc , float $llx , float $lly , float $urx , float $ury , string $contents , string $title , string $icon , int $open)

ページの特定の位置に注釈を追加します。注釈は、 テキストが記入された小さな矩形で、ページ上のどこにでも配置できます。 折りたたんで保持することもでき、折りたたまれていない場合は、 指定したアイコンがプレースホルダとして使用されます。

ドキュメントを印刷したり表示したりする場合には、 注釈は見えません。注釈が現れるのは、Acrobat Distiller™ や Ghostview を用いてドキュメントを PDF に変換した場合です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

llx

左下角の x 座標。

lly

左下角の y 座標。

urx

右上角の x 座標。

ury

右上角の y 座標。

contents

注釈のテキスト。

title

注釈のヘッダに表示される、タイトル。

icon

注釈が折りたたまれた際に表示されるアイコン。 comment、insert、 note、paragraph、 newparagraph、key あるいは help のいずれかが設定できます。

open

open がゼロでない場合、 PDF ビューア上でドキュメントをオープンすると、 注釈は折りたたまれずに表示されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_add_pdflink()
- ps_add_launchlink()
- ps_add_locallink()
- ps_add_weblink()

---

ps_add_pdflink

(PECL ps:1.1.0-1.3.5)

ps_add_pdflink — 別の PDF ドキュメントのページへのリンクを追加する

説明

bool **ps_add_pdflink** (resource $psdoc , float $llx , float $lly , float $urx , float $ury , string $filename , int $page , string $dest)

指定した位置に、別の PDF ドキュメントへのハイパーリンクを設定します。 リンクをクリックすると、そのドキュメントの指定されたページに移動します。 ドキュメントの最初のページのページ番号は 1 です。

ハイパーリンクのリンク元は矩形で表され、その左下の座標は (llx , lly)、 そして右上の座標が (urx , ury) となります。この矩形はデフォルトでは細い青線で表されます。

ドキュメントを印刷したり表示したりする場合には、 ハイパーリンクは見えません。ハイパーリンクが現れるのは、Acrobat Distiller™ や Ghostview を用いてドキュメントを PDF に変換した場合です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

llx

左下角の x 座標。

lly

左下角の y 座標。

urx

右上角の x 座標。

ury

右上角の y 座標。

filename

リンクがクリックされた際に開かれる PDF ドキュメントの名前。

page

対象 PDF ドキュメントのページ番号。

dest

dest は、ドキュメントがどのように 表示されるかを指定します。とりうる値は fitpage または fitwidth、 fitheight、fitbbox のいずれかです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_add_launchlink()
- ps_add_locallink()
- ps_add_weblink()

---

ps_add_weblink

(PECL ps:1.1.0-1.3.5)

ps_add_weblink — Web 上の場所へのリンクを追加する

説明

bool **ps_add_weblink** (resource $psdoc , float $llx , float $lly , float $urx , float $ury , string $url)

指定した位置に、Web ページを指すハイパーリンクを設定します。 ハイパーリンクのリンク元は矩形で表され、その左下の座標は (llx , lly)、そして右上の座標が (urx , ury) となります。 この矩形はデフォルトでは細い青線で表されます。

ドキュメントを印刷したり表示したりする場合には、ハイパーリンクは 見えません。ハイパーリンクが現れるのは、Acrobat Distiller™ や Ghostview を用いてドキュメントを PDF に変換した場合です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

llx

左下角の x 座標。

lly

左下角の y 座標。

urx

右上角の x 座標。

ury

右上角の y 座標。

url

リンクがクリックされた際に開かれるハイパーリンクの URL。 例えば http://www.php.net

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ps_add_launchlink()*
- *ps_add_locallink()*
- *ps_add_pdflink()*

---

ps_arc

(PECL ps:1.1.0-1.3.5)

ps_arc — 反時計回りに円弧を描く

説明

*bool **ps_arc** (resource $psdoc , float $x , float $y , float $radius , float $alpha , float $beta)*

(x , y) を中心とした円の一部を描きます。円弧は alpha で指定された角度からはじまり beta で指定された角度で終わります。反時計回りに描かれます (時計回りに描くには ps_arcn() を使用します)。 また、alpha で指定された角度からはじまって beta で指定された角度で終わるサブパスが、 現在のパスに追加されます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

円の中心の x 座標。

y

円の中心の y 座標。

radius

円の半径。

alpha

開始位置の角度を度単位で指定。

beta

終了位置の角度を度単位で指定。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ps_arcn()*

---

ps_arcn

(PECL ps:1.1.0-1.3.5)

ps_arcn — 時計回りに円弧を描く

説明

*bool **ps_arcn** (resource $psdoc , float $x , float $y , float $radius , float $alpha , float $beta)*

(x , y) を中心とした円の一部を描きます。円弧は alpha で指定された角度からはじまり beta で指定された角度で終わります。 時計回りに描かれます (反時計回りに描くには ps_arc() を使用します)。 また、beta で指定された角度からはじまって alpha で指定された角度で終わるサブパスが、カレントパスに追加されます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

円の中心の x 座標。

y

円の中心の y 座標。

radius

円の半径。

alpha

開始位置の角度を度単位で指定。

beta

終了位置の角度を度単位で指定。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ps_arc()*

---

ps_begin_page

(PECL ps:1.1.0-1.3.5)

ps_begin_page — 新しいページを開始する

説明

*bool **ps_begin_page** (resource $psdoc , float $width , float $height)*

*新しいページを開始します。width や height というパラメータを見ると いかにもページ単位でサイズを変えられるように思えますが、PostScript ではこれは不可能です。最初に **ps_begin_page()** を呼んだ際に指定されたサイズが、ドキュメント全体に適用されます。 2 回目以降に呼ばれた場合はこのパラメータは意味を持たず、 単に新しいページが作られるだけです。しかし、もしあなたが PostScript ドキュメントを PDF に変換しようと考えているのなら話は別です。 この関数はドキュメントの各ページごとに pdfmark を設定するので、 PDF に変換した後はページごとにサイズを変えることができます。*

PostScript はさまざまなページサイズを知らないので、pslib ではドキュメントの各ページに対してバウンディングボックスを設定します。 このサイズは PostScript ビューアによって評価され、 ドキュメントヘッダの BoundingBox より優先されます。 左下角が (0, 0) ではない BoundingBox を設定すると、 予期せぬ結果を引き起こすことがあります。なぜなら、 ページ単位のバウンディングボックスは常に左下角が (0, 0) であり、 これが全体の設定より優先されるからです。

個々のページの情報はカプセル化して保存されています。どういうことかというと、 あるページで設定した情報のほとんどは次のページに引き継がれないということです。

*最初の **ps_begin_page()** 呼び出しまでに一度も ps_findfont() が呼び出されていなかった場合は、 PostScript ドキュメントのヘッダが出力され、 バウンディングボックスが最初のページのサイズに設定されます。 バウンディングボックスの左下角は (0, 0) に設定されます。もしすでに ps_findfont() が呼ばれていた場合は既にヘッダが出力されてしまっており、 ドキュメントに適切なバウンディングボックスが設定されていないことになります。 このようなことを避けるために、ps_findfont() や **ps_begin_page()** を実行する前にはいつも ps_set_info() を実行し、BoundingBox や Orientation の値を設定しておくべきです。*

> ***注意**: pslib のバージョン 0.2.6 まででは、 ps_set_info() によって事前に設定されていたり ps_findfont() が事前にコールされていない限り、 この関数は常に BoundingBox および Orientation を上書きします。*

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

width

ピクセル単位のページの幅 (例: A4 の場合は 596)。

height

ピクセル単位のページの高さ (例: A4 の場合は 842)。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ps_end_page()*
- *ps_findfont()*
- *ps_set_info()*

---

ps_begin_pattern

(PECL ps:1.2.0-1.3.5)

ps_begin_pattern — 新しいパターンを開始する

説明

int **ps_begin_pattern** (resource $psdoc , float $width , float $height , float $xstep , float $ystep , int $painttype)

新しいパターンを開始します。パターンとは、 領域を塗りつぶすための描画データを含むページのようなものです。 ps_setcolor() をコールする際に pattern に色空間を指定すると、 これは色のように使用されます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

width

ピクセル単位のパターンの幅。

height

ピクセル単位のパターンの高さ。

x-step

パターンを水平方向に並べる際のピクセル単位の間隔。

y-step

パターンを垂直方向に並べる際のピクセル単位の間隔。

painttype

1 あるいは 2 でなければなりません。

返り値

パターンの ID を返します。失敗した場合は **FALSE** を返します。

例

Example#1 パターンの作成および使用

```
<?php
$ps = ps_new();

if (!ps_open_file($ps, "pattern.ps")) {
  print "PostScript ファイルをオープンできません\n";
  exit;
}

ps_set_parameter($ps, "warning", "true");
ps_set_info($ps, "Creator", "pattern.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Pattern example");


$pspattern = ps_begin_pattern($ps, 10.0, 10.0, 10.0, 10.0, 1);
ps_setlinewidth($ps, 0.2);
ps_setcolor($ps, "stroke", "rgb", 0.0, 0.0, 1.0, 0.0);
ps_moveto($ps, 0, 0);
ps_lineto($ps, 7, 7);
ps_stroke($ps);
ps_moveto($ps, 0, 7);
ps_lineto($ps, 7, 0);
ps_stroke($ps);
ps_end_pattern($ps);

ps_begin_page($ps, 596, 842);
ps_setcolor($ps, "both", "pattern", $pspattern, 0.0, 0.0, 0.0);
ps_rect($ps, 50, 400, 200, 200);
ps_fill($ps);
ps_end_page($ps);

ps_close($ps);
ps_delete($ps);
?>
```

参考

- ps_end_pattern()
- ps_setcolor()
- ps_shading_pattern()

---

ps_begin_template

(PECL ps:1.2.0-1.3.5)

ps_begin_template — 新しいテンプレートを開始する

説明

```
int ps_begin_template ( resource $psdoc , float $width , float $height )
```

新しいテンプレートを開始します。テンプレートは、postscript 言語からコールされます。パターンと似ていますが、画像のように使用されます。テンプレートは、ドキュメント内で何回も使用される描画内容、 例えば会社のロゴなどに対して使用されることが多いでしょう。 テンプレート内では、すべての描画関数が使用されます。 テンプレートは、ps_place_image() によって配置されるまでは描画されません。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

width

ピクセル単位のテンプレートの幅。

height

ピクセル単位のテンプレートの高さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 テンプレートの作成および使用

```
<?php
$ps = ps_new();

if (!ps_open_file($ps, "template.ps")) {
  print "PostScript ファイルをオープンできません\n";
   exit;
}

ps_set_parameter($ps, "warning", "true");
ps_set_info($ps, "Creator", "template.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Template example");

$pstemplate = ps_begin_template($ps, 30.0, 30.0);
ps_moveto($ps, 0, 0);
ps_lineto($ps, 30, 30);
ps_moveto($ps, 0, 30);
ps_lineto($ps, 30, 0);
ps_stroke($ps);
ps_end_template($ps);

ps_begin_page($ps, 596, 842);
ps_place_image($ps, $pstemplate, 20.0, 20.0, 1.0);
ps_place_image($ps, $pstemplate, 50.0, 30.0, 0.5);
ps_place_image($ps, $pstemplate, 70.0, 70.0, 0.6);
ps_place_image($ps, $pstemplate, 30.0, 50.0, 1.3);
ps_end_page($ps);

ps_close($ps);
ps_delete($ps);
?>
```

参考

- ps_end_template()

---

ps_circle

(PECL ps:1.1.0-1.3.5)

ps_circle — 円を描く

説明

```
bool ps_circle ( resource $psdoc , float $x , float $y , float $radius )
```

(x , y) を中心とする円を描きます。描画開始位置は (x +radius , y) です。もしパスの外部でこの関数が呼び出された場合、 新しいパスを開始します。パスの内部で呼び出された場合は、 円をサブパスとして追加します。ひとつ前の描画処理の終点が (x +radius , y) でなかった場合、パスに隙間ができることになります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

円の中心の x 座標。

y

円の中心の y 座標。

radius

円の半径。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ps_arc()*
- *ps_arcn()*

---

ps_clip

(PECL ps:1.1.0-1.3.5)

ps_clip — 現在のパスを描画範囲として指定する

説明

*bool **ps_clip** (resource $psdoc)*

現在のパスを、描画範囲の境界として定義します。 この領域の外に描画された内容は、見えなくなります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *ps_closepath()*

---

ps_close_image

(PECL ps:1.1.0-1.3.5)

ps_close_image — 画像を閉じ、メモリを開放する

説明

*void **ps_close_image** (resource $psdoc , int $imageid)*

画像を閉じ、そのリソースを開放します。 いったん閉じられた後は、画像を使用することはできません。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

imageid

ps_open_image() あるいは ps_open_image_file() が返す、 画像のリソース ID。

返り値

*成功した場合に **NULL** を、失敗した場合に **FALSE** を返します。*

参考

- *ps_open_image()*
- *ps_open_image_file()*

---

ps_close

(PECL ps:1.1.0-1.3.5)

ps_close — PostScript ドキュメントを閉じる

説明

bool ***ps_close*** *(resource $psdoc)*

PostScript ドキュメントを閉じます。

*この関数は、PostScript ドキュメントのトレーラーを書き込みます。 また、ブックマークツリーも書き込みます。**ps_close()** は、リソースの開放は行いません。これは ps_delete() が行います。*

もし事前にコールされていない場合、この関数は ps_delete() からもコールされます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- ps_open_file()
- ps_delete()

---

ps_closepath_stroke

(PECL ps:1.1.0-1.3.5)

ps_closepath_stroke — パスを閉じ、描画する

説明

bool ***ps_closepath_stroke*** *(resource $psdoc)*

パスの終了点を開始点とつなげ、出来上がった線を描画します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- ps_closepath()

---

ps_closepath

(PECL ps:1.1.0-1.3.5)

ps_closepath — パスを閉じる

説明

bool ***ps_closepath*** *(resource $psdoc)*

パスの終了点と開始点をつなげます。出来上がったパスは、 描画、塗りつぶし、切り取りなどに使用されます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- ps_clip()
- ps_closepath_stroke()

---

ps_continue_text

(PECL ps:1.1.0-1.3.5)

ps_continue_text — 次の行にテキストを続ける

説明

bool **ps_continue_text** (resource $psdoc , string $text)

最終行の次の行からテキストを出力します。行間隔は "leading" の値が用いられます。この値は ps_set_value() を用いて指定する必要があります。 実際のテキストの出力位置は、"textx" と "texty" の値によって決まります。 この値は ps_get_value() を用いて取得することができます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

出力するテキスト。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_show()
- ps_show_xy()
- ps_show_boxed()

---

ps_curveto

(PECL ps:1.1.0-1.3.5)

ps_curveto — 曲線を描く

説明

bool **ps_curveto** (resource $psdoc , float $x1 , float $y1 , float $x2 , float $y2 , float $x3 , float $y3)

現在のパスに対して、3 つの制御点を指定して 3 次ベジエ曲線を追加します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x1

最初の制御点の x 座標。

y1

最初の制御点の y 座標。

x2

2 番目の制御点の x 座標。

y2

2 番目の制御点の y 座標。

x3

3 番目の制御点の x 座標。

y3

3 番目の制御点の y 座標。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_lineto()

---

ps_delete

(PECL ps:1.1.0-1.3.5)

ps_delete — PostScript ドキュメントの全リソースを削除する

説明

bool **ps_delete** (resource $psdoc)

ドキュメントが使用していたメモリを開放します。また、 もし事前に ps_close() でファイルが閉じられていない場合はファイルを閉じます。 どんな場合でも、事前に ps_close() でファイルを閉じておくべきです。なぜなら ps_close() はファイルをただ閉じるだけではなく、トレーラーを出力するからです。 ここには、ドキュメントのページ数やブックマーク階層といった情報が含まれます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_close()

---

ps_end_page

(PECL ps:1.1.0-1.3.5)

ps_end_page — ページを終了する

説明

bool **ps_end_page** (resource $psdoc)

ps_begin_page() によって開始したページを終了します。 ページを終了しても、現在の描画コンテキストはそのままになります。つまり、 もしページ内部でフォントが読み込まれたのなら、 それを再読み込みしてしなければなりません。 また、行の幅や色などのその他の描画パラメータも再設定する必要があります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_begin_page()

---

ps_end_pattern

(PECL ps:1.2.0-1.3.5)

ps_end_pattern — パターンを終了する

説明

bool **ps_end_pattern** (resource $psdoc)

ps_begin_pattern() によって開始したパターンを終了します。 パターンが終了すると、領域の塗りつぶし時にそれを使用できるようになります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_begin_pattern()

ps_end_template

(PECL ps:1.2.0-1.3.5)

ps_end_template — テンプレートを終了する

説明

bool **ps_end_template** (resource $psdoc)

ps_begin_template() によって開始したテンプレートを終了します。 テンプレートが終了すると、それを画像のように使用できるようになります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_begin_template()

---

ps_fill_stroke

(PECL ps:1.1.0-1.3.5)

ps_fill_stroke — 現在のパスを塗りつぶし、線を引く

説明

bool **ps_fill_stroke** (resource $psdoc)

ps_lineto() のような描画関数によって事前に作成されたパスを塗りつぶし、描画します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_fill()
- ps_stroke()

---

ps_fill

(PECL ps:1.1.0-1.3.5)

ps_fill — 現在のパスを塗りつぶす

説明

bool **ps_fill** (resource $psdoc)

ps_lineto() のような描画関数によって事前に作成されたパスを塗りつぶします。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_fill_stroke()
- ps_stroke()

---

ps_findfont

(PECL ps:1.1.0-1.3.5)

ps_findfont — フォントを読み込む

説明

int **ps_findfont** (resource $psdoc , string $fontname , string $encoding [, bool $embed])

あとで使用するために、フォントを読み込みます。読み込んだフォントを実際に利用するためには、 ps_setfont() で設定しなければなりません。 文字の間隔を計算するため、この関数は adobe フォントメトリックファイルを必要とします。 ページの内部で読み込まれたフォントは、 そのページ内でのみ有効となります。ドキュメント全体で使われるフォントは、 最初の ps_begin_page() の実行より前に読み込まれなければなりません。 ページとページの間で **ps_findfont()** が呼ばれた場合は、それ以降のページでフォントが有効になります。

afm ファイルの名前は、 fontname .afm でなければなりません。 フォントを埋め込む場合は、フォントのアウトラインを含む fontname .pfb が存在しなければなりません。

最初のページを処理する前に **ps_findfont()** をコール際、 postscript ヘッダが出力されます。ここには、ドキュメント全体に適用される BoundingBox が含まれます。 通常は、BoundingBox を設定するのは最初に ps_begin_page() がコールされたときで、これは **ps_findfont()** をコールした後になります。 したがって、**ps_findfont()** のコール時にはまだ BoundingBox が設定されておらず、警告が発生してしまいます。 こうなることを避けるため、**ps_findfont()** をコールする前に ps_set_parameter() をコールし、BoundingBox を設定しておくべきです。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

fontname

フォントの名前。

encoding

ps_findfont() は、encoding で渡されたファイルを読み込もうと試みます。 エンコーディングファイルは、 **dvips(1)** で使われるものと同形式です。 そこにはフォントエンコーディングベクタ（現在は利用されていませんが、 存在する必要があります）が含まれており、また afm ファイルから 生成されたリゲチャのリストを拡張するための拡張リゲチャが含まれています。

encoding は **NULL** または空文字列とすることも可能で、 その場合はデフォルトエンコーディング（TeXBase1）が用いられます。

encoding が builtin と指定された場合は、 エンコード処理は行われずにフォント固有のエンコーディングがそのまま用いられます。 これは、記号フォントを扱う場合に便利です。

embed

0 より大きい値を設定すると、フォントがドキュメントに埋め込まれます。 これを使用するには、フォントのアウトライン（.pfb ファイル）が必要です。

返り値

成功した場合にフォントの ID を、失敗した場合にゼロを返します。 ID は正の数値です。

参考

- ps_begin_page()
- ps_setfont()

---

ps_get_buffer

(PECL ps:1.1.0-1.3.5)

ps_get_buffer — 生成された PS データを含むバッファの内容を取得する

説明

string **ps_get_buffer** (resource $psdoc)

この関数は、まだ実装されていません。常に空の文字列を返します。 メモリ内でのデータ作成が求められる場合に、postscript ファイルを内部バッファに読み込み、バッファの内容をこの関数で取得する、 という実装を計画中です。 現在は、メモリ内で作成されたデータはバッファリングされず直接ブラウザに送られます。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

参考

- ps_open_file()

---

ps_get_parameter

(PECL ps:1.1.0-1.3.5)

ps_get_parameter — パラメータを取得する

説明

string **ps_get_parameter** (resource $psdoc , string $name [, float $modifier])

ps_set_parameter() で直接設定されたか、 あるいは他の関数で間接的に設定されたパラメータの値を取得します。 パラメータは文字列値として定義されます。この関数は、 同じく ps_set_parameter() で設定されたリソースを取得するためには利用できません。

パラメータ name は、以下の値をとります。

fontname

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントの名前。

fontencoding

現在アクティブなフォントのエンコーディング。

dottedversion

元になる pslib ライブラリのバージョン。書式は <major>.<minor>.<subminor>

scope

現在の描画スコープ。object, document, null, page, pattern, path, template, prolog, font, glyph。

ligaturedisolvechar

リゲチャを溶かす文字。たとえば、リゲチャ `ff' を含むフォントで `|' がこの文字に指定されていた場合は、 `f|f' はリゲチャ `ff' ではなく 2 つの独立した `f' となります。

imageencoding

画像の符号化に使用するエンコーディング。 hex あるいは 85。 hex エンコーディングは、画像の各バイトを postscript ファイルの 2 バイトであらわします。85 は Ascii85 エンコーディングを表します。

linenumbermode

行番号を段落単位でつける場合は paragraph、 ボックス内でつける場合は box を設定します。

linebreak

テキストを ps_show_boxed() で出力する場合にのみ使用します。 true を設定すると、 改行時にキャリッジリターンが付加されます。

parbreak

テキストを ps_show_boxed() で出力する場合にのみ使用します。 true を設定すると、 キャリッジリターンで新しい段落を開始します。

hyphenation

テキストを ps_show_boxed() で出力する場合にのみ使用します。 true を設定すると、 ハイフネーション辞書が設定されており存在する場合に、段落のハイフネーション処理が行われます。

hyphendict

ハイフネーションパターンの辞書のファイル名。

返り値

成功した場合にパラメータの値を、エラー時には **FALSE** を返します。

参考

- ps_set_parameter()

---

ps_get_value

(PECL ps:1.1.0-1.3.5)

ps_get_value — 値を取得する

説明

float **ps_get_value** (resource $psdoc , string $name [, float $modifier])

ps_set_value() で設定された値を取得します。 値は浮動小数点数値として定義されます。

パラメータ name には以下の値を設定できます。

fontsize

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントの名前。

font

現在アクティブなフォント自身。

imagewidth

パラメータ modifier で渡された ID の画像の幅。

imageheight

パラメータ modifier で渡された ID の画像の高さ。

capheight

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントにおける、 大文字の M の高さ。

ascender

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントの ascender (上に突き出している部分の長さ)。

descender

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントの descender (下に突き出している部分の長さ)。

italicangle

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントの斜体の角度。

underlineposition

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントの下線の位置。

underlinethickness

現在アクティブなフォント、 あるいはパラメータ modifier で指定された ID を持つフォントの下線の太さ。

textx

現在のテキスト出力位置の x 座標。

texty

現在のテキスト出力位置の y 座標。

textrendering

現在のテキストのレンダリングモード。

textrise

ベースライン上部の空間。

leading

テキストの行間隔をポイント数で指定。

wordspacing

単語と単語の間の空白の幅を、空白文字の幅を基準として指定。

charspacing

文字と文字の間の空白。0.0 でない場合は、リゲチャは常に分解されます。

hyphenminchars

単語の末尾で、ハイフネーションの対象となる最小文字数。

parindent

各段落の最初の n 行を字下げします。

numindentlines

parindent が 0.0 でない場合に、段落内で字下げする行数。

parskip

段落と段落の間隔。

linenumberspace

行番号を表示するために、各行の先頭に置く空白。

linenumbersep

行番号と行の内容の間隔。

major

pslib のメジャーバージョン番号。

minor

pslib のマイナーバージョン番号。

subminor, revision

pslib の詳細バージョン番号。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

name

値の名前。

modifier

modifier は、値を取得しようとしている リソースを指定します。たとえばフォントや画像の ID が考えられます。

返り値

*成功した場合にパラメータの値を、失敗した場合に **FALSE** を返します。*

参考

- *ps_set_value()*

---

ps_hyphenate

(PECL ps:1.1.1-1.3.5)

ps_hyphenate — 単語をハイフネーションする

説明

*array **ps_hyphenate** (resource $psdoc , string $text)*

*渡された単語をハイフネーションします。**ps_hyphenate()** は、(ps_set_value() で設定した) hyphenminchars の値と (ps_set_parameter() で設定した) hyphendict パラメータの内容を評価します。 hyphendict は、この関数を呼ぶ前に必ず設定しておかなければなりません。*

この関数を利用するためには、LC_CTYPE が適切に設定されている必要があります。 PostScript 拡張が初期化される際、環境変数の値が利用されます。 Unix 環境を利用している方は、詳細な情報は locale の man ページを見てください。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

text にはアルファベット以外の文字を 含むべきではありません。ハイフンで分割できる位置が、 整数値の配列として返されます。配列の個々の要素の内容は、 text 内でハイフンを入れられる場所の 次の文字の位置を表します。

返り値

*テキスト内で改行を入れられる位置を表す整数値の配列、 あるいはエラー時には **FALSE** を返します。*

例

Example#1 テキストのハイフネーション

```
<?php
$word = "Koordinatensystem";
$psdoc = ps_new();
ps_set_parameter($psdoc, "hyphendict", "hyph_de.dic");
$hyphens = ps_hyphenate($psdoc, $word);
for($i=0; $i<strlen($word); $i++) {
  echo $word[$i];
  if(in_array($i, $hyphens))
    echo "-";
}
ps_delete($psdoc);
?>
```

上の例の出力は以下となります。

```
Ko-ordi-na-ten-sys-tem
```

参考

- *ps_show_boxed()*
- *locale(1)*

---

ps_include_file

(PECL ps:1.3.4-1.3.5)

ps_include_file — 外部ファイルを生の PostScript コードとして読み込む

説明

bool **ps_include_file** (resource $psdoc , string $file)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

psdoc

ps_new() が返す postscript ファイルの ID リソース。

file

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ps_lineto

(PECL ps:1.1.0-1.3.5)

ps_lineto — 直線を描く

説明

bool **ps_lineto** (resource $psdoc , float $x , float $y)

現在の位置から指定された座標までの直線を現在のパスに追加します。 直線の開始位置を指定するには ps_moveto() を利用します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

直線の終了地点の x 座標。

y

直線の終了地点の y 座標。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 矩形を描画する

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "rectangle.ps")) {
  print "PostScript ファイルをオープンできません\n";
   exit;
}

ps_set_info($ps, "Creator", "rectangle.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Lineto example");

ps_begin_page($ps, 596, 842);
ps_moveto($ps, 100, 100);
ps_lineto($ps, 100, 200);
ps_lineto($ps, 200, 200);
ps_lineto($ps, 200, 100);
ps_lineto($ps, 100, 100);
ps_stroke($ps);
ps_end_page($ps);

ps_delete($ps);
?>
```

参考

- ps_moveto()

---

ps_makespotcolor

(PECL ps:1.1.0-1.3.5)

ps_makespotcolor — スポット色を作成する

説明

int **ps_makespotcolor** (resource $psdoc , string $name [, float $reserved])

現在の塗りつぶし色からスポット色を作成します。塗りつぶし色の色空間は rgb, cmyk または グレースケールのいずれかでなければなりません。スポット色には任意の名前をつけることができます。 スポット色には、ps_setcolor() を用いてどんな色でも設定できます。 ドキュメントが印刷されずに postscript ビューアで表示される場合にも、 指定した色空間の色が使用されます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

name

スポット色の名前、例えば Pantone 5565。

返り値

成功した場合にスポット色の ID を、失敗した場合にゼロを返します。

例

Example#1 スポット色の作成と使用

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "spotcolor.ps")) {
  print "PostScript ファイルをオープンできません\n";
   exit;
}

ps_set_info($ps, "Creator", "spotcolor.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Spot color example");

ps_begin_page($ps, 596, 842);
ps_setcolor($ps, "fill", "cmyk", 0.37, 0.0, 0.34, 0.34);
$spotcolor = ps_makespotcolor($ps, "PANTONE 5565 C", 0);
ps_setcolor($ps, "fill", "spot", $spotcolor, 0.5, 0.0, 0.0);
ps_moveto($ps, 100, 100);
ps_lineto($ps, 100, 200);
ps_lineto($ps, 200, 200);
ps_lineto($ps, 200, 100);
ps_lineto($ps, 100, 100);
ps_fill($ps);
ps_end_page($ps);

ps_delete($ps);
?>
```

この例では、深緑（オリーブ）色で 50 % の濃度で矩形を塗りつぶすスポット色 "PANTONE 5565 C" を作成します。

参考

- ps_setcolor()

---

ps_moveto

(PECL ps:1.1.0-1.3.5)

ps_moveto — 現在位置を設定する

説明

bool **ps_moveto** (resource $psdoc , float $x , float $y)

現在位置を新しい座標に指定します。前回のパスが終了してからはじめて **ps_moveto()** が呼ばれた場合、新しいパスを開始します。 もしすでにパスが開始されているときに呼ばれた場合は、 単に現在位置を設定し、サブパスを開始します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

移動する位置の x 座標。

y

移動する位置の y 座標。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_lineto()

---

ps_new

(PECL ps:1.1.0-1.3.5)

ps_new — 新しい PostScript ドキュメントオブジェクトを作成する

説明

resource **ps_new** (void)

新しいドキュメントのインスタンスを生成します。 ディスク上やメモリ内にファイルを作ることはありません。 ドキュメントのインスタンスがすべてを設定します。 **ps_new()** の後には、通常は ps_open_file() がコールされます。これにより、 実際に postscript ドキュメントが作成されます。

返り値

成功した場合に PostScript ドキュメントのリソースを、 失敗した場合に **FALSE** を返します。 この返り値は、その他のすべての関数の最初の引数として渡されます。

参考

- ps_delete()

---

ps_open_file

(PECL ps:1.1.0-1.3.5)

ps_open_file — 出力用のファイルを開く

説明

bool **ps_open_file** (resource $psdoc [, string $filename])

ディスク上に新しいファイルを作成し、PostScript ドキュメントをその中に書き込みます。ps_close() が呼ばれると、ファイルは閉じられます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

filename

postscript ファイルの名前。 filename が渡されなかった場合、 ドキュメントはメモリ内に作成され、内容は直接ブラウザに出力されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_close()

---

ps_open_image_file

(PECL ps:1.1.0-1.3.5)

ps_open_image_file — ファイルから画像を開く

説明

int **ps_open_image_file** (resource $psdoc , string $type , string $filename [, string $stringparam [, int $intparam]])

後で使用するために、画像を読み込みます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

type

画像の形式。png、 jpeg あるいは eps。

filename

画像データを含むファイルの名前。

stringparam

使用されません。

intparam

使用されません。

返り値

成功した場合に画像の ID を、失敗した場合にゼロを返します。 ID は、0 より大きい正の数値です。

参考

- ps_open_image()
- ps_place_image()
- ps_close_image()

---

ps_open_image

(PECL ps:1.1.0-1.3.5)

ps_open_image — 後で配置するために画像を読み込む

説明

int **ps_open_image** (resource $psdoc , string $type , string $source , string $data , int $lenght , int $width , int $height , int $components , int $bpc , string $params)

すでにメモリ上にある画像を読み込みます。現在は source は評価されず、常に memory と仮定されます。 画像データは、左上から右下に向かってピクセルデータを順に並べたものです。 各ピクセルは、色コンポーネント components で構成されており、このコンポーネントの大きさは bpc ビットです。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

type

画像の形式。png、 jpeg あるいは eps。

source

使用されません。

data

画像データ。

length

画像データの長さ。

width

画像の幅。

height

画像の高さ。

components

各ピクセルのコンポーネントの数。 1 (グレースケール画像)、 3 (rgb 画像) あるいは 4 (cmyk, rgba 画像)。

bpc

コンポーネント単位のビット数 (ほとんどの場合は 8)

params

返り値

成功した場合に画像の ID を、失敗した場合にゼロを返します。 ID は、0 より大きい正の数値です。

参考

- ps_open_image_file()
- ps_place_image()
- ps_close_image()

---

ps_open_memory_image

(PECL ps:1.1.0-1.3.5)

ps_open_memory_image — GD 画像を受け取り、PS ドキュメントにはめ込む画像を返す

説明

int **ps_open_memory_image** (resource $psdoc , int $gd)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

psdoc

ps_new() が返す postscript ファイルの ID リソース。

gd

---

ps_place_image

(PECL ps:1.1.0-1.3.5)

ps_place_image — 画像をページに配置する

説明

bool **ps_place_image** (resource $psdoc , int $imageid , float $x , float $y , float $scale)

以前に読み込まれている画像をページに配置します。画像の縮尺は変更できます。 画像を回転させる場合は、事前に ps_rotate() で座標系を回転させておきます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

imageid

ps_open_image() あるいは ps_open_image_file() が返す、 画像のリソース ID。

x

画像の左下角の x 座標。

y

画像の左下角の y 座標。

scale

画像の拡大率。1.0 を指定すると、解像度は 72 dpi となります。 これにより、各ピクセルが 1 ポイントと等しくなるからです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_open_image()
- ps_open_image_file()

---

ps_rect

(PECL ps:1.1.0-1.3.5)

ps_rect — 矩形を描く

説明

bool **ps_rect** (resource $psdoc , float $x , float $y , float $width , float $height)

(x , y) を左下の角とする 矩形を描きます。描画は左下の角から始まって同じ位置で終わります。 もしパスの外部でこの関数が呼び出された場

合、新しいパスを開始します。 パスの内部で呼び出された場合は、矩形をサブパスとして追加します。一つ前の描画処理が 左下角の位置で終わっていなかった場合は、パスに隙間ができることになります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

矩形の左下角の x 座標。

y

矩形の左下角の y 座標。

width

画像の幅。

height

画像の高さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ps_arc()*
- ***ps_cirle()***
- *ps_lineto()*

---

ps_restore

(PECL ps:1.1.0-1.3.5)

ps_restore — 以前に保存されたコンテキストを復元する

説明

bool ***ps_restore*** *(resource $psdoc)*

以前に保存されている描画コンテキストを復元します。 すべての *ps_save()* 呼び出しには、対応する ***ps_restore()*** の呼び出しが必要です。 座標系変換、線の設定、色の設定などは、すべて *ps_save()* のコール前の状態に戻ります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ps_save()*

---

ps_rotate

(PECL ps:1.1.0-1.3.5)

ps_rotate — 回転の程度を設定する

説明

bool ***ps_rotate*** *(resource $psdoc , float $rot)*

座標系の回転を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

rot

回転角度を度単位で指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 座標系の回転

```
<?php
function rectangle($ps) {
    ps_moveto($ps, 0, 0);
    ps_lineto($ps, 0, 50);
    ps_lineto($ps, 50, 50);
    ps_lineto($ps, 50, 0);
    ps_lineto($ps, 0, 0);
    ps_stroke($ps);
}

$ps = ps_new();
if (!ps_open_file($ps, "rotation.ps")) {
  print "PostScript ファイルをオープンできません\n";
  exit;
}

ps_set_info($ps, "Creator", "rotation.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Rotation example");
ps_set_info($ps, "BoundingBox", "0 0 596 842");

$psfont = ps_findfont($ps, "Helvetica", "", 0);

ps_begin_page($ps, 596, 842);
ps_set_text_pos($ps, 100, 100);
ps_save($ps);
ps_translate($ps, 100, 100);
ps_rotate($ps, 45);
rectangle($ps);
ps_restore($ps);
ps_setfont($ps, $psfont, 8.0);
ps_show($ps, "Text without rotation");
ps_end_page($ps);

ps_delete($ps);
?>
```

上の例では、座標系を回転させることによって、図形（ここでは単なる矩形）を回転させるという一般的な方法を説明しています。 図形の座標系は (0,0) を起点とみなしているため、 図形がページの端にない場合には座標系の変換も行います。 ps_translate() および **ps_rotate()** の順番には気をつけてください。 上の例では、変換前の座標系での (100, 100) を基準として矩形を回転させています。 2 つの文の順序を入れ替えると、まったく異なる結果となります。

これ以降のテキストをもとの位置に出力させるには、 ps_save() および ps_restore() を使用して、座標系に関する変更内容をカプセル化します。

参考

- ps_scale()
- ps_translate()

---

ps_save

(PECL ps:1.1.0-1.3.5)

ps_save — 現在のコンテキストを保存する

説明

bool **ps_save** (resource $psdoc)

現在の描画コンテキストを保存します。 そこには色や変換・回転その他の情報が含まれています。保存されたコンテキストは ps_restore() で復元できます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_restore()

---

ps_scale

(PECL ps:1.1.0-1.3.5)

ps_scale — 縮尺を設定する

説明

bool **ps_scale** (resource $psdoc , float $x , float $y)

座標系の、水平方向・垂直方向の縮尺をそれぞれ設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

水平方向の倍率。

y

垂直方向の倍率。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_rotate()
- ps_translate()

---

ps_set_border_color

(PECL ps:1.1.0-1.3.5)

ps_set_border_color — 注記の枠線の色を設定する

説明

bool **ps_set_border_color** (resource $psdoc , float $red , float $green , float $blue)

ps_add_weblink() や ps_add_pdflink() などの関数で追加されたリンクは、 pdf に変換して pdf ビューアで見た際には矩形で囲まれて表示されます。 この矩形は、postscript ドキュメントでは表示されません。 この関数は、矩形の枠線の色を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

red

枠線の色の red 要素。

green

枠線の色の green 要素。

blue

枠線の色の blue 要素。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_set_border_dash()
- ps_set_border_style()

---

ps_set_border_dash

(PECL ps:1.1.0-1.3.5)

ps_set_border_dash — 注記の枠線の、破線の状態を設定する

説明

bool **ps_set_border_dash** (resource $psdoc , float $black , float $white)

ps_add_weblink() や ps_add_pdflink() などの関数で追加されたリンクは、 pdf に変換して pdf ビューアで見た際には矩形で囲まれて表示さ

れます。 この矩形は、postscript ドキュメントでは表示されません。 この関数は、破線の「線」の部分と「空白」の部分の長さを設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

black

破線の長さ。

white

破線と破線の間隔。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- ps_set_border_color()
- ps_set_border_style()

---

ps_set_border_style

(PECL ps:1.1.0-1.3.5)

ps_set_border_style — 注記の枠線の形式を設定する

説明

bool **ps_set_border_style** (resource $psdoc , string $style , float $width)

ps_add_weblink() や ps_add_pdflink() などの関数で追加されたリンクは、 pdf に変換して pdf ビューアで見た際には矩形で囲まれて表示されます。 この矩形は、postscript ドキュメントでは表示されません。 この関数は、その線の形式と幅を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

style

style は、 solid（実線） または dashed（破線）のいずれかを設定できます。

width

枠線の幅。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- ps_set_border_color()
- ps_set_border_dash()

---

ps_set_info

(PECL ps:1.1.0-1.3.5)

ps_set_info — ドキュメントの情報を設定する

説明

bool **ps_set_info** (resource $p , string $key , string $val)

ドキュメントの情報フィールドを設定します。この情報は、PostScript ファイルのヘッダにコメントとして表示されます。 ドキュメントが pdf に変換された場合にも、 このフィールドはドキュメントの情報として使用されます。

通常 BoundingBox は最初のページの設定と同じ値を 設定しておきます。 ps_findfont() が事前に呼ばれていなかった場合のみ、この設定が利用されます。そのような場合、 もしこの関数で明示的に設定していなければ BoundingBox は空白のままとなります。

postscript ファイルのヘッダが既に書き込まれている場合には、 この関数は何の意味も持ちません。この関数は、最初のページの前、 あるいは ps_findfont() が最初にコールされる前にコールされる必要があります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

key

設定する情報フィールドの名前。設定できる値は Keywords、Subject、 Title、Creator、 Author、BoundingBox および Orientation です。このうちのいくつかは、PostScript ビューアに対して意味を持つものであることに注意しましょう。

value

情報フィールドの値。 Orientation フィールドには Portrait あるいは Landscape が設定できます。BoundingBox は 4 つの数値からなる文字列で、最初の 2 つがページの左下角の座標を表します。 残りの 2 つが右上角の座標です。

> **注意**: pslib のバージョン 0.2.6 まででは、 ps_findfont() が事前にコールされていない限り、 この関数は常に BoundingBox および Orientation を上書きします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_findfont()
- ps_begin_page()

---

ps_set_parameter

(PECL ps:1.1.0-1.3.5)

ps_set_parameter — パラメータを設定する

説明

bool **ps_set_parameter** (resource $psdoc , string $name , string $value)

多くの関数で利用されるパラメータを設定します。 パラメータは文字列値として定義されます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

name

使用できる名称については ps_get_parameter() を参照ください。

value

パラメータの値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_get_parameter()
- ps_set_value()

---

ps_set_text_pos

(PECL ps:1.1.0-1.3.5)

ps_set_text_pos — テキストの出力位置を設定する

説明

bool **ps_set_text_pos** (resource $psdoc , float $x , float $y)

テキストを出力する位置を設定します。別の方法として、x と y の値を別々に ps_set_value() で設定することもできます。 この場合の設定項目は、それぞれ textx と texty となります。

テキストを所定の位置に出力したいのなら、テキストの位置を指定してから ps_show() をコールするよりも、 ps_show_xy() を使用するほうが便利です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

新しいテキスト位置の x 座標。

y

新しいテキスト位置の *y* 座標。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 指定した位置にテキストを配置する

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "text.ps")) {
  print "PostScript ファイルをオープンできません\n";
  exit;
}

ps_set_info($ps, "Creator", "rectangle.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Text placement example");

ps_begin_page($ps, 596, 842);
$psfont = ps_findfont($ps, "Helvetica", "", 0);
ps_setfont($ps, $psfont, 8.0);
ps_show_xy($ps, "Some text at (100, 100)", 100, 100);

ps_set_value($ps, "textx", 100);
ps_set_value($ps, "texty", 120);
ps_show($ps, "Some text at (100, 120)");
ps_end_page($ps);

ps_delete($ps);
?>
```

参考

- ps_set_value()
- ps_show()

---

ps_set_value

(PECL ps:1.1.0-1.3.5)

ps_set_value — 値を設定する

説明

bool **ps_set_value** (resource $psdoc , string $name , float $value)

多くの関数で利用される値を設定します。 パラメータは浮動小数点数値として定義されます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

name

name は以下のうちのひとつです。

textrendering

テキストをどのように見せるか。

textx

出力テキストの *x* 座標。

texty

出力テキストの *y* 座標。

wordspacing

単語の間隔を空白文字に対する比率で設定する。

leading

行間をピクセルで指定する。

value

パラメータの値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_get_value()
- ps_set_parameter()

---

ps_setcolor

(PECL ps:1.1.0-1.3.5)

ps_setcolor — 色を設定する

説明

bool **ps_setcolor** (resource $psdoc , string $type , string $colorspace , float $c1 , float $c2 , float $c3 , float $c4)

描画色、塗りつぶし色、あるいはその両方を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

type

type は both、fill、 あるいは fillstroke のいずれかです。

colorspace

colorspace は gray、 rgb、cmyk、 spot、pattern のいずれかです。 colorspace の値によって、以下のパラメータのうち最初のひとつ・最初の 3 つ あるいはすべてが利用されます。

c1

colorspace の値によって、red 成分 (rgb の場合)、 cyan 成分 (cmyk の場合)、グレースケール (gray の場合)、 スポット色の ID あるいはパターンの ID のいずれかとなります。

c2

colorspace の値によって、green 成分 (rgb の場合)、 magenta 成分 (cmyk の場合) のいずれかとなります。

c3

colorspace の値によって、blue 成分 (rgb の場合)、 yellow 成分 (cmyk の場合) のいずれかとなります。

c4

colorspace が cmyk の場合にのみ設定され、black 成分の値を指定します。

バグ

2 番目のパラメータは、現在は常に評価されるわけではありません。 設定内容にかかわらず、fillstroke が渡されたときのような振る舞いをすることがあります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ps_setdash

(PECL ps:1.1.0-1.3.5)

ps_setdash — 破線の形状を設定する

説明

bool **ps_setdash** (resource $psdoc , float $on , float $off)

破線の「線」の部分と「空白」の部分の長さを設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

on

破線の長さ。

off

破線と破線の間隔の長さ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_setpolydash()

---

ps_setflat

(PECL ps:1.1.0-1.3.5)

ps_setflat — 平面度を設定する

説明

bool **ps_setflat** (resource $psdoc , float $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

value

value は 0.2 から 1 までの間でなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ps_setfont

(PECL ps:1.1.0-1.3.5)

ps_setfont — 以降の出力で用いるフォントを設定する

説明

bool **ps_setfont** (resource $psdoc , int $fontid , float $size)

フォントを設定します。このフォントは、事前に ps_findfont() で読み込まれている必要があります。 フォントを設定せずにテキストを出力するとエラーとなります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

fontid

ps_findfont() が返す、フォント ID。

size

フォントのサイズ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_findfont()

---

ps_setgray

(PECL ps:1.1.0-1.1.1)

ps_setgray — グレー値を設定する

説明

bool **ps_setgray** (resource $psdoc , float $gray)

以降の描画処理で使われるグレー値を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

gray

値は 0（白）から 1（黒）までの間でなければなりません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ps_setcolor()*

---

ps_setlinecap

(PECL ps:1.1.0-1.3.5)

ps_setlinecap — 線端の形状を設定する

説明

bool **ps_setlinecap** (resource $psdoc , int $type)

線端をどのように表示するかを設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

type

線端の形式。 PS_LINECAP_BUTT、 PS_LINECAP_ROUND あるいは PS_LINECAP_SQUARED のいずれか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ps_setlinejoin()*
- *ps_setlinewidth()*
- *ps_setmiterlimit()*

---

ps_setlinejoin

(PECL ps:1.1.0-1.3.5)

ps_setlinejoin — 線の連結方法を設定する

説明

bool **ps_setlinejoin** (resource $psdoc , int $type)

線がどのように連結されるかを設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

type

線の連結方式。 PS_LINEJOIN_MITER、 PS_LINEJOIN_ROUND あるいは PS_LINEJOIN_BEVEL のいずれか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *ps_setlinecap()*
- *ps_setlinewidth()*
- *ps_setmiterlimit()*

ps_setlinewidth

(PECL ps:1.1.0-1.3.5)

ps_setlinewidth — 線幅を設定する

説明

bool **ps_setlinewidth** (resource $psdoc , float $width)

以降の描画処理での線幅を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

width

線の幅（ポイント数）。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_setlinecap()
- ps_setlinejoin()
- ps_setmiterlimit()

---

ps_setmiterlimit

(PECL ps:1.1.0-1.3.5)

ps_setmiterlimit — miter limit 値を設定する

説明

bool **ps_setmiterlimit** (resource $psdoc , float $value)

2 本の直線が小さい角度で連結され、かつ連結方法が PS_LINEJOIN_MITER に設定されている場合、 出来上がる線の角の部分が非常に長くなります。miter limit は、miter length（角の長さ）と線幅の比率の最大値です。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

value

miter length と線幅の比率の最大値。大きな値（> 10）を設定すると、2 本の直線が狭い角度で交わる際に、 角の部分が非常に長くなってしまいます。 よくわからない場合はデフォルトのままにしておきましょう。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_setlinecap()
- ps_setlinejoin()
- ps_setlinewidth()

---

ps_setoverprintmode

(PECL ps:1.3.0-1.3.5)

ps_setoverprintmode — overprint モードを設定する

説明

bool **ps_setoverprintmode** (resource $psdoc , int $mode)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

psdoc

ps_new() が返す postscript ファイルの ID リソース。

mode

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

---

ps_setpolydash

(PECL ps:1.1.0-1.3.5)

ps_setpolydash — 破線の形状を設定する

説明

bool **ps_setpolydash** (resource $psdoc , float $arr)

破線の「線」の部分と「空白」の部分の長さを設定します。 **ps_setpolydash()** は、 より複雑な破線パターンを設定するために用いられます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

arr

arr は「線」の部分の長さと 「空白」の部分の長さを互い違いにならべたリストです。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 破線を描画する

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "polydash.ps")) {
   print "PostScript ファイルをオープンできません\n";
      exit;
}

ps_set_info($ps, "Creator", "polydash.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Poly dash example");

ps_begin_page($ps, 596, 842);
ps_setpolydash($ps, array(10, 5, 2, 5));
ps_moveto($ps, 100, 100);
ps_lineto($ps, 200, 200);
ps_stroke($ps);
ps_end_page($ps);

ps_delete($ps);
?>
```

この例は、長さ 10 ポイントと 2 ポイントの線を、 5 ポイントの空白でつなげた破線を描画します。

参考

- ps_setdash()

---

ps_shading_pattern

(PECL ps:1.3.0-1.3.5)

ps_shading_pattern — シェーディング用のパターンを作成する

説明

int **ps_shading_pattern** (resource $psdoc , int $shadingid , string $optlist)

シェーディング用のパターンを作成します。これは ps_shading() を呼ぶ前に行わなければなりません。 シェーディングパターンは、標準のパターンと同じように用いられます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

shadingid

事前に ps_shading() で作成したシェーディングの ID。

optlist

このパラメータは、現在使用されていません。

返り値

*成功した場合にパターンの ID を、失敗した場合に **FALSE** を返します。*

参考

- *ps_shading()*
- *ps_shfill()*

---

ps_shading

(PECL ps:1.3.0-1.3.5)

ps_shading — 以降の出力で用いるシェーディングを作成する

説明

*int **ps_shading** (resource $psdoc , string $type , float $x0 , float $y0 , float $x1 , float $y1 , float $c1 , float $c2 , float $c3 , float $c4 , string $optlist)*

シェーディングを作成します。これは ps_shfill() や ps_shading_pattern() で用いられます。

シェーディングの色には、pattern 以外の任意の色空間が利用できます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

type

シェーディングの種類で、radial (放射状) または axial (直線状) のいずれかです。 シェーディングは現在の塗りつぶし色で始まり、 c1 から c4 までのパラメータで指定される色で終わります (パラメータの意味は ps_setcolor() を参照ください)。

x0, x1, y0, y1

x0 , y0 , x1 , y1 は、 シェーディングの開始位置・終了位置の座標です。 シェーディングの形式が radial の場合は、 それぞれ開始円・終了円の中心となります。

c1, c2, c3, c4

このパラメータの意味は ps_setcolor() を参照してください。

optlist

type が radial の場合、optlist にはパラメータ r0 と r1 が含まれる必要があります。これらはそれぞれ開始位置と終了位置を表す 円の半径です。

返り値

*成功した場合にパターンの ID を、失敗した場合に **FALSE** を返します。*

参考

- *ps_shading_pattern()*
- *ps_shfill()*

---

ps_shfill

(PECL ps:1.3.0-1.3.5)

ps_shfill — 範囲をシェーディングで塗りつぶす

説明

*bool **ps_shfill** (resource $psdoc , int $shadingid)*

範囲をシェーディングで塗りつぶします。シェーディングは事前に ps_shading() で作られている必要があります。 この関数は、ps_shading_pattern() でシェーディングからパターンを作成し、 それを塗りつぶし色として使用するのと同じことを行います。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

shadingid

事前に ps_shading() で作成したシェーディングの ID。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- ps_shading()
- ps_shading_pattern()

---

ps_show_boxed

(PECL ps:1.1.0-1.3.5)

ps_show_boxed — テキストをボックス内に出力する

説明

*int **ps_show_boxed** (resource $psdoc , string $text , float $left , float $bottom , float $width , float $height , string $hmode [, string $feature])*

与えられたボックスの中にテキストを出力します。ボックスの左下の座標が (left , bottom) となります。 必要に応じて改行が挿入されます。連続する空白文字は、ひとつとして扱われ、 タブ文字は空白文字として扱われます。

パラメータ hyphenation が true に設定されており、またパラメータ hyphendict に有効なハイフネーションファイルが設定されている場合に、テキストのハイフネーションが行われます。 行間隔の設定は、値 leading で行います。 TeX と同様に、段落と段落の間には空行が挿入されます。値 parindent が 0.0 より大きく設定されている場合、 最初の n 行は字下げされます。n の行数は、値 numindentlines で設定します。 最初の m 段落を字下げしないようにするには、 値 parindentskip に正の数値を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

指定したボックスの中に出力するテキスト。

left

ボックスの左下角の x 座標。

bottom

ボックスの左下角の y 座標。

width

ボックスの幅。

height

ボックスの高さ。

hmode

パラメータ hmode は "justify"（均等割付）・ "fulljustify"（完全な均等割付）・"right"（右寄せ）・"left"（左寄せ）・ "center"（センタリング）のどれかの値を設定します。 "justify" と "fulljustify" の違いは、ボックスの最終行の処理です。 fulljustify モードでは、 段落の最後の行である場合を除いて最終行も均等割付をします。 justify モードでは、最終行は常に左寄せとなります。

feature

用いられるパラメータ

***ps_show_boxed()** の出力は、パラメータや値（それぞれ ps_set_parameter() や ps_set_value() で設定されたもの）で設定できます。 テキスト出力に影響するパラメータや値のうち、以下のものが評価されます。*

leading (value)

連続する 2 つの行の間隔です。

linebreak (parameter)

キャリッジリターンを、空白ではなく改行として扱いたい場合は "true" に設定します。デフォルトは "false" です。

parbreak (parameter)

単一行のデータ中のキャリッジリターンを、空白ではなく段落区切りとして 扱いたい場合は "true" に設定します。デフォルトは "true" です。

hyphenation (parameter)

ハイフネーションを有効にする場合は "true" に設定します。 その際、パラメータ "hyphendict" に辞書を設定しておく必要があります。 デフォルトは "false" です。

hyphendict (parameter)

ハイフネーションパターンの辞書ファイル名です（以下を参照）。

hyphenminchar (value)

ハイフンの前後に最低限必要な文字数です。つまり、少なくともこの値の 2 倍以上の文字数の単語でないとハイフネーションされないということです。 デフォルトは 3 です。値をゼロに設定すると、デフォルトの値が使われます。

parindent (value)

段落の最初の m 行に対する字下げの量をピクセル単位で設定します。 m は、値 "numindentlines" で設定します。

parskip (value)

段落間の空白の追加量をピクセル単位で設定します。デフォルトは 0 で、 通常の行間設定をそのまま利用します。

numindentlines (value)

段落の最初の部分で字下げ対象となる行数です。デフォルトは 1 です。

parindentskip (value)

字下げ処理を行わない段落数を設定します。デフォルトは 0 です。 この設定が便利に使えるのは、章の始めの段落・2 つのボックスにまたがる テキストなどです。このような場合には値を 1 に設定します。

linenumbermode (parameter)

行番号のつけ方を設定します。"box" を設定するとボックス全体に 番号をつけ、"paragraph" を設定するとその中の段落単位で番号をつけます。

linenumberspace (value)

行番号を表示するための、左側のスペースの幅です。行番号は、 このスペースに右詰めで表示されます。デフォルトは 20 です。

linenumbersep (value)

行番号欄と行本文欄の間の空白です。デフォルトは 5 です。

ハイフネーション

パラメータ hyphenation が true に設定され、かつ 有効なハイフネーション辞書が設定されている場合に、テキストが ハイフネーションされます。pslib は独自のハイフネーション辞書を持っておらず、 openoffice または scribus, koffice の辞書のうちひとつを利用します。 これらのソフトウェアがインストールされている環境では、さまざまな言語の辞書が 以下のディレクトリで見つけられます。

- /usr/share/apps/koffice/hyphdicts/
- /usr/lib/scribus/dicts/
- /usr/lib/openoffice/share/dict/ooo/

現時点では、scribus が最も完璧なハイフネーション辞書を持っているようです。

返り値

書くことができなかった文字数を返します。

参考

- ps_continue_text()

---

ps_show_xy2

(PECL ps:1.1.0-1.3.5)

ps_show_xy2 — テキストを指定した位置に出力する

説明

bool **ps_show_xy2** (resource $psdoc , string $text , int $len , float $xcoor , float $ycoor)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ps_show_xy

(PECL ps:1.1.0-1.3.5)

ps_show_xy — 指定された位置にテキストを出力する

説明

bool **ps_show_xy** (resource $psdoc , string $text , float $x , float $y)

指定された位置にテキストを出力します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

出力するテキスト。

x

テキストを囲むボックスの左下角の x 座標。

y

テキストを囲むボックスの左下角の y 座標。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_continue_text()
- ps_show()

---

ps_show2

(PECL ps:1.1.0-1.3.5)

ps_show2 — テキストを現在の位置に出力する

説明

bool **ps_show2** (resource $psdoc , string $text , int $len)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

psdoc

ps_new() が返す postscript ファイルの ID リソース。

text

len

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

ps_show

(PECL ps:1.1.0-1.3.5)

ps_show — テキストを出力する

説明

bool **ps_show** (resource $psdoc , string $text)

現在の位置にテキストを出力します。テキストの位置を設定するには、 関数 ps_set_value() の値 textx および texty に x 座標と y 座標を設定します。 事前に ps_setfont() でフォントが設定されていない場合、この関数はエラーを返します。

ps_show() は、以下のようなパラメータや値 （それぞれ ps_set_parameter() や ps_set_value() で設定されたもの）の内容を評価します。

charspacing (value)

連続する 2 つのグリフ間の距離です。値がゼロ以外の場合、リゲチャは 解消されます。ゼロ未満の値を指定することも可能です。

kerning (parameter)

このパラメータを "false" に設定するとカーニングが無効になります。 デフォルトではカーニングが有効になっています。

ligatures (parameter)

このパラメータを "false" に設定するとリゲチャを使用しないようになります。 デフォルトではリゲチャを使用するようになっています。

underline (parameter)

このパラメータを "true" に設定すると下線が引かれます。 デフォルトでは下線が無効になっています。

overline (parameter)

このパラメータを "true" に設定すると上線が引かれます。 デフォルトでは上線が無効になっています。

strikeout (parameter)

このパラメータを "true" に設定すると打ち消し線が引かれます。 デフォルトでは打ち消し線が無効になっています。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

出力するテキスト。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_continue_text()
- ps_show_xy()
- ps_setfont()

---

ps_string_geometry

(PECL ps:1.2.0-1.3.5)

ps_string_geometry — 文字列のジオメトリを取得する

説明

array **ps_string_geometry** (resource $psdoc , string $text [, int $fontid [, float $size]])

この関数は ps_stringwidth() と似ていますが、 返り値は、テキストの幅、ascender および descender を含む配列形式になります。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

ジオメトリを計算するテキスト。

fontid

使用するフォントの ID。フォントが指定されていない場合は、 現在のフォントが使用されます。

size

フォントのサイズ。指定されていない場合は現在のサイズが使用されます。

返り値

文字の寸法を格納した配列です。'width' には ps_stringwidth() で返される文字列の幅が入ります。 下に突き出している部分の長さの最大値が 'descender' に、 また上に突き出している部分の長さの最大値が 'ascender' に入ります。

参考

- ps_continue_text()
- ps_stringwidth()

---

ps_stringwidth

(PECL ps:1.1.0-1.3.5)

ps_stringwidth — 文字列の幅を取得する

説明

float **ps_stringwidth** (resource $psdoc , string $text [, int $fontid [, float $size]])

指定されたフォントとサイズで出力した場合に、 文字列の幅が何ポイントになるかを計算します。 正確な幅を計算するため、この関数は Adobe のフォントメトリックファイルを必要とします。カーニングが有効になっている場合、 それも考慮して計算します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

text

幅を計算するテキスト。

fontid

使用するフォントの ID。フォントが指定されていない場合は、 現在のフォントが使用されます。

size

フォントのサイズ。指定されていない場合は現在のサイズが使用されます。

返り値

文字列の幅をポイント数で返します。

参考

- ps_string_geometry()

---

ps_stroke

(PECL ps:1.1.0-1.3.5)

ps_stroke — 現在のパスを描画する

説明

bool **ps_stroke** (resource $psdoc)

ps_lineto() のような関数で事前に組み立てられたパスを描画します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_closepath_stroke()
- ps_fill()
- ps_fill_stroke()

---

ps_symbol_name

(PECL ps:1.2.0-1.3.5)

ps_symbol_name — グリフ名を取得する

説明

string **ps_symbol_name** (resource $psdoc , int $ord [, int $fontid])

この関数は、有効なグリフ名を取得するために Adobe のフォントメトリックファイルを必要とします。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

ord

ord は、フォントエンコーディングベクタ の中でのグリフの位置です。

fontid

使用するフォントの ID。フォントが指定されていない場合は、 現在のフォントが使用されます。

返り値

指定されたフォントのグリフ名を返します。

参考

- ps_symbol()
- ps_symbol_width()

---

ps_symbol_width

(PECL ps:1.2.0-1.3.5)

ps_symbol_width — グリフの幅を取得する

説明

float **ps_symbol_width** (resource $psdoc , int $ord [, int $fontid [, float $size]])

指定されたフォントとサイズで出力した場合に、 グリフの幅が何ポイントになるかを計算します。正確な幅を計算するため、 この関数は Adobe のフォントメトリックファイルを必要とします。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

ord

ord は、 フォントエンコーディングベクタの中でのグリフの位置です。

fontid

使用するフォントの ID。フォントが指定されていない場合は、 現在のフォントが使用されます。

size

フォントのサイズ。指定されていない場合は現在のサイズが使用されます。

返り値

グリフの幅をポイント数で返します。

参考

- ps_symbol()
- ps_symbol_name()

---

ps_symbol

(PECL ps:1.2.0-1.3.5)

ps_symbol — グリフを出力する

説明

bool **ps_symbol** (resource $psdoc , int $ord)

現在のフォントのフォントエンコーディングベクタの中で ord 番目の位置にあるグリフを出力します。 フォントエンコーディングは、ps_findfont() でフォントを読み込む際に設定することができます。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

ord

フォントエンコーディングベクタの中でのグリフの位置。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- ps_symbol_name()
- ps_symbol_width()

---

ps_translate

(PECL ps:1.1.0-1.3.5)

ps_translate — 座標変換を設定する

説明

bool **ps_translate** (resource $psdoc , float $x , float $y)

新しい座標系の原点を設定します。

パラメータ

psdoc

ps_new() が返す、postscript ファイルのリソース ID。

x

変換後の座標系の原点の x 座標。

y

変換後の座標系の原点の y 座標。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 座標系の変換

```
<?php
function rectangle($ps) {
    ps_moveto($ps, 0, 0);
    ps_lineto($ps, 0, 50);
    ps_lineto($ps, 50, 50);
    ps_lineto($ps, 50, 0);
    ps_lineto($ps, 0, 0);
    ps_stroke($ps);
}

$ps = ps_new();
if (!ps_open_file($ps, "translate.ps")) {
  print "Cannot open PostScript file¥n";
  exit;
}

ps_set_info($ps, "Creator", "translate.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Translated example");
ps_set_info($ps, "BoundingBox", "0 0 596 842");

$psfont = ps_findfont($ps, "Helvetica", "", 0);

ps_begin_page($ps, 596, 842);
ps_set_text_pos($ps, 100, 100);
ps_translate($ps, 500, 750);
rectangle($ps);
ps_translate($ps, -500, -750);
ps_setfont($ps, $psfont, 8.0);
ps_show($ps, "Text at initial position");
ps_end_page($ps);

ps_begin_page($ps, 596, 842);
ps_set_text_pos($ps, 100, 100);
ps_save($ps);
ps_translate($ps, 500, 750);
rectangle($ps);
ps_restore($ps);
ps_setfont($ps, $psfont, 8.0);
ps_show($ps, "Text at initial position");
ps_end_page($ps);

ps_delete($ps);
?>
```

上の例は、図形（ここでは単なる矩形）を、図形自身の座標系を使用したまま ページ内の任意の位置に配置するための 2 種類の方法を示しています。 矩形を描画する前に、座標系の原点を変更しているのがポイントです。 矩形を描画した後は、座標系をもとに戻さなければなりません。

2 ページ目では、少し異なった、よりエレガントな方法を使用しています。 座標系の変換を元に戻すために **ps_translate()** をもう一度コールするのではなく、 座標系の変更前に描画コンテキストを保存しておいて、 矩形の描画後にそれを復元しています。

参考

- ps_scale()
- ps_rotate()

目次

- ps_add_locallink — 同一ドキュメント内のページへのリンクを追加する
- ps_add_note — 現在のページに注釈を追加する
- ps_add_pdflink — 別の PDF ドキュメントのページへのリンクを追加する
- ps_add_weblink — Web 上の場所へのリンクを追加する
- ps_arc — 反時計回りに円弧を描く
- ps_arcn — 時計回りに円弧を描く
- ps_begin_page — 新しいページを開始する
- ps_begin_pattern — 新しいパターンを開始する
- ps_begin_template — 新しいテンプレートを開始する
- ps_circle — 円を描く
- ps_clip — 現在のパスを描画範囲として指定する
- ps_close_image — 画像を閉じ、メモリを開放する
- ps_close — PostScript ドキュメントを閉じる
- ps_closepath_stroke — パスを閉じ、描画する
- ps_closepath — パスを閉じる
- ps_continue_text — 次の行にテキストを続ける
- ps_curveto — 曲線を描く
- ps_delete — PostScript ドキュメントの全リソースを削除する
- ps_end_page — ページを終了する
- ps_end_pattern — パターンを終了する
- ps_end_template — テンプレートを終了する
- ps_fill_stroke — 現在のパスを塗りつぶし、線を引く
- ps_fill — 現在のパスを塗りつぶす
- ps_findfont — フォントを読み込む
- ps_get_buffer — 生成された PS データを含むバッファの内容を取得する
- ps_get_parameter — パラメータを取得する
- ps_get_value — 値を取得する
- ps_hyphenate — 単語をハイフネーションする
- ps_include_file — 外部ファイルを生の PostScript コードとして読み込む
- ps_lineto — 直線を描く
- ps_makespotcolor — スポット色を作成する
- ps_moveto — 現在位置を設定する
- ps_new — 新しい PostScript ドキュメントオブジェクトを作成する
- ps_open_file — 出力用のファイルを開く
- ps_open_image_file — ファイルから画像を開く
- ps_open_image — 後で配置するために画像を読み込む
- ps_open_memory_image — GD 画像を受け取り、PS ドキュメントにはめ込む画像を返す
- ps_place_image — 画像をページに配置する
- ps_rect — 矩形を描く
- ps_restore — 以前に保存されたコンテキストを復元する
- ps_rotate — 回転の程度を設定する
- ps_save — 現在のコンテキストを保存する
- ps_scale — 縮尺を設定する
- ps_set_border_color — 注記の枠線の色を設定する
- ps_set_border_dash — 注記の枠線の、破線の状態を設定する
- ps_set_border_style — 注記の枠線の形式を設定する
- ps_set_info — ドキュメントの情報を設定する
- ps_set_parameter — パラメータを設定する
- ps_set_text_pos — テキストの出力位置を設定する
- ps_set_value — 値を設定する
- ps_setcolor — 色を設定する
- ps_setdash — 破線の形状を設定する
- ps_setflat — 平面度を設定する
- ps_setfont — 以降の出力で用いるフォントを設定する
- ps_setgray — グレー値を設定する
- ps_setlinecap — 線端の形状を設定する
- ps_setlinejoin — 線の連結方法を設定する
- ps_setlinewidth — 線幅を設定する
- ps_setmiterlimit — miter limit 値を設定する
- ps_setoverprintmode — overprint モードを設定する
- ps_setpolydash — 破線の形状を設定する
- ps_shading_pattern — シェーディング用のパターンを作成する
- ps_shading — 以降の出力で用いるシェーディングを作成する
- ps_shfill — 範囲をシェーディングで塗りつぶす

- ps_show_boxed — テキストをボックス内に出力する
- ps_show_xy2 — テキストを指定した位置に出力する
- ps_show_xy — 指定された位置にテキストを出力する
- ps_show2 — テキストを現在の位置に出力する
- ps_show — テキストを出力する
- ps_string_geometry — 文字列のジオメトリを取得する
- ps_stringwidth — 文字列の幅を取得する
- ps_stroke — 現在のパスを描画する
- ps_symbol_name — グリフ名を取得する
- ps_symbol_width — グリフの幅を取得する
- ps_symbol — グリフを出力する
- ps_translate — 座標変換を設定する

Pspell 関数

導入

これらの関数により、単語のスペルチェックを行い、修正案を提案させる ことが可能となります。

要件

PHP を pspell サポートつきでコンパイルするには、aspellライブラリが 必要です。これは、» http://aspell.sourceforge.net/ で取得可能です。

インストール手順

PHP をコンパイルする際に、オプション --with-pspell[=dir] を追加する 必要があります。

注意: **Win32 ユーザへの注意** この拡張モジュールを動作させるには、 Windows システムの PATH が通った場所に DLL ファイルが存在する必要があります。 FAQ の "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" で、その方法を説明しています。 DLL ファイルを PHP のフォルダから Windows のシステムディレクトリにコピーしても動作します (システムディレクトリは、デフォルトで PATH に含まれるからです) が、これは推奨しません。 この拡張モジュールを使用するには、以下のファイルが PATH の通った場所にある必要があります。 aspell の bin フォルダにある aspell-15.dll
Win32 は、PHP 4.3.3 以降でのみサポートします。また、 aspell バージョン 0.50 以降が必要です。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

PSPELL_FAST (integer)
PSPELL_NORMAL (integer)
PSPELL_BAD_SPELLERS (integer)
PSPELL_RUN_TOGETHER (integer)

---

pspell_add_to_personal

(PHP 4 >= 4.0.2, PHP 5)

pspell_add_to_personal — ユーザの単語リストに単語を追加する

説明

bool **pspell_add_to_personal** (int $dictionary_link , string $word)

pspell_add_to_personal() はユーザの単語リスト に単語を追加します。ディレクトリをオープンするために pspell_new_config() を pspell_config_personal() とともに使用した場合、 pspell_save_wordlist() で単語リストを保存することが可能です。

パラメータ

dictionary_link

word

追加する単語。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_add_to_personal()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config($pspell_config);

pspell_add_to_personal($pspell_link, "Vlad");
pspell_save_wordlist($pspell_link);
?>
```

注意

> **注意**: この関数は、pspell .11.2 と aspell .32.5 以降でない限り動作しないことに注意してください。

---

pspell_add_to_session

(PHP 4 >= 4.0.2, PHP 5)

pspell_add_to_session — 現在のセッションの単語リストに単語を追加する

説明

bool **pspell_add_to_session** (int $dictionary_link , string $word)

pspell_add_to_session() は、 現在のセッションに関連する単語リストに単語を追加します。この関数は、 pspell_add_to_personal() に似ています。

パラメータ

dictionary_link

word

追加する単語。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

pspell_check

(PHP 4 >= 4.0.2, PHP 5)

pspell_check — 単語をチェックする

説明

bool **pspell_check** (int $dictionary_link , string $word)

pspell_check() は単語のスペルをチェックします。

パラメータ

dictionary_link

word

チェックする単語。

返り値

スペルが正しい場合に **TRUE**、そうでない場合に **FALSE** を返します。

例

Example#1 pspell_check() の例

```
<?php
$pspell_link = pspell_new("en");

if (pspell_check($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
?>
```

---

pspell_clear_session

(PHP 4 >= 4.0.2, PHP 5)

pspell_clear_session — 現在のセッションをクリアする

説明

bool **pspell_clear_session** (int $dictionary_link)

pspell_clear_session() は、 現在のセッションをクリアします。現在の単語リストはブランクになり、 例えば pspell_save_wordlist() でこのリストを保存しても何もおきません。

パラメータ

dictionary_link

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_add_to_personal() の例

```
<?php
$pspell_config  = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link  = pspell_new_config($pspell_config);

pspell_add_to_personal($pspell_link, "Vlad");
pspell_clear_session($pspell_link);
pspell_save_wordlist($pspell_link);    //"Vlad"  は保存されません
?>
```

---

pspell_config_create

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_create — 辞書をオープンする際に使用する設定を作成する

説明

int **pspell_config_create** (string $language [, string $spelling [, string $jargon [, string $encoding]]])

辞書をオープンする際に使用する設定を作成します。

pspell_config_create() は、 pspell_new() の構文に非常によく似ています。実際、 pspell_new_config() の直後に **pspell_config_create()** を使用した場合、全く同じ 結果となります。しかし、新しい設定を作成した後、 pspell_new_config() をコールする前に関数 **pspell_config_*()** を使用することで いくつかの進んだ機能が使用できます。

より詳細な情報と例については、pspell Web サイト » http://aspell.net/ のオンラインマニュアルを参照ください。

パラメータ

language

パラメータ language は、2 文字の ISO 639 言語コードと オプションでダッシュまたはアンダースコアの後に 2 文字の ISO 3166 国コードからなる言語コードです。

spelling

パラメータ spelling は、英語のように複数のスペルがある言語に関して スペルの指定を行うものです。指定可能な値は、 'american', 'british', 'canadian'です。

jargon

パラメータ jargon は、同じ language および spelling パラメータを有する 2 つの異なる単語リストを区別するための 追加情報を有しています。

encoding

パラメータ encoding は、単語のエンコーディングとして 予想されるものです。有効な値は、'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32' です。このパラメータはよくテストされていないため、 使用する際には注意してください。

返り値

pspell 設定 ID、あるいはエラー時に **FALSE** を返します。

例

Example#1 pspell_config_create()

```
<?php
$pspell_config  = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link  = pspell_new_personal($pspell_config, "en");
?>
```

---

pspell_config_data_dir

(PHP 5)

pspell_config_data_dir — 言語データファイルの場所

説明

bool **pspell_config_data_dir** (int $conf , string $directory)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

pspell_config_dict_dir

(PHP 5)

pspell_config_dict_dir — メイン単語リストの場所

説明

bool **pspell_config_dict_dir** (int $conf , string $directory)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

pspell_config_ignore

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_ignore — 長さが N 文字未満の単語を無視する

説明

bool **pspell_config_ignore** (int $dictionary_link , int $n)

pspell_config_ignore() は、 pspell_new_config() をコールする前に使用しなければなりません。 この関数は、スペルチェッカに無視させる短い単語の長さを指定します。

パラメータ

dictionary_link

n

n 文字より短い単語をスキップします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_config_ignore()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd");    // エラーとはなりません
?>
```

---

pspell_config_mode

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_mode — 返される提案の数のモードを変更する

説明

bool **pspell_config_mode** (int $dictionary_link , int $mode)

pspell_config_mode() は、 pspell_new_config() のコール前に設定を行う際に使用します。この関数は、 pspell_suggest() により返される修正候補の数を定義します。

パラメータ

dictionary_link

mode

パラメータ mode は、スペルチェッカの動作モードです。 使用可能なモードを以下に示します。

- **PSPELL_FAST** - 高速モード (修正候補の数は最小)
- **PSPELL_NORMAL** - 通常モード (修正候補はより多い)
- **PSPELL_BAD_SPELLERS** - 低速モード (修正候補は多い)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_config_mode() の例

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

---

pspell_config_personal

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_personal — 個人の単語リストを保持するファイルを設定する

説明

bool **pspell_config_personal** (int $dictionary_link , string $file)

個人の単語リストファイルを設定します。個人の単語リストは、 pspell_new_config()をコールした後にロードされ、 標準的な単語リストに追加されて使用されます。このファイルは、 pspell_save_wordlist() により個人的な単語リスト が保存されるファイルでもあります。

pspell_config_personal() は、 pspell_new_config() をコールする前に設定を行うために使用する必要があります。

パラメータ

dictionary_link

file

個人の単語リスト。存在しない場合は作成します。PHP の実行ユーザ (たとえば nobody) が書き込み可能である必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_config_personal()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

注意

注意: この関数は、 pspell .11.2 および aspell .32.5 以降でない限り動作しないことに注意してください。

---

pspell_config_repl

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_repl — 置換候補を保持するファイルを設定する

説明

bool **pspell_config_repl** (int $dictionary_link , string $file)

置換候補を保持するファイルを設定します。

置換の組は、スペルチェッカの品質を改善します。 単語のスペルミスをした場合、そして、適当な修正候補がリストにあった場合、 pspell_store_replacement() を置換候補を保存するために使用し、置換候補を含む単語リストを保存するために pspell_save_wordlist()を使

用することが可能です。

pspell_config_repl() は、 pspell_new_config() をコールする前に設定を行うために使用する必要があります。

パラメータ

dictionary_link

file

このファイルは PHP の実行ユーザ（たとえば nobody）が書き込み可能である必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_config_repl()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

注意

> **注意**: この関数は、pspell .11.2 および aspell .32.5 以降でない限り動作しないことに注意してください。

---

pspell_config_runtogether

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_runtogether — 複合語を有効な単語の組み合わせとして考慮する

説明

bool **pspell_config_runtogether** (int $dictionary_link , bool $flag)

この関数は、複合語を正しい複合語として処理するかどうかを定義します。 つまり、"thecat" には、二つの単語の間に空白はありませんが、 正しい複合語となります。この設定の変更は、 pspell_check() による返り値にのみ影響を与えます。 pspell_suggest() は、この場合でも修正候補を返します。

pspell_config_runtogether() は、 pspell_new_config() をコールする前に設定を行うために使用する必要があります。

パラメータ

dictionary_link

flag

連続した単語を複合語として扱う場合な **TRUE**、 それ以外の場合は **FALSE**。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_config_runtogether()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_runtogether($pspell_config, true);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

---

pspell_config_save_repl

(PHP 4 >= 4.0.2, PHP 5)

pspell_config_save_repl — 単語リストと共に置換リストを保存するかどうかを定義する

説明

bool **pspell_config_save_repl** (int $dictionary_link , bool $flag)

pspell_config_save_repl() は、 pspell_save_wordlist() が単語リストと共に置換リストを保存するかどうかを定義します。 通常はこの関数を使用する必要はありません。なぜなら、 pspell_config_repl() を使用した場合は置換の組は pspell_save_wordlist() により保存され、 そうでない場合には置換の組は保存されないためです。

pspell_config_save_repl() は、 pspell_new_config() のコール前に設定を行う際に使用します。

パラメータ

dictionary_link

flag

置換リストを保存する場合に **TRUE**、それ以外の場合に **FALSE**。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

> **注意**: この関数は、pspell .11.2およびaspell .32.5 以降を有していない場合には 動作しないことに注意してください。

---

pspell_new_config

(PHP 4 >= 4.0.2, PHP 5)

pspell_new_config — 指定した設定に基づき新規辞書をロードする

説明

int **pspell_new_config** (int $config)

pspell_new_config() は、 pspell_config_create() で作成され、 関数 **pspell_config_*()** で修正された設定を使用して 新規辞書をオープンします。この方法は最も柔軟で、 pspell_new() および pspell_new_personal() で提供された全ての機能を有しています。

パラメータ

config

パラメータ config は、config が作成された際に pspell_config_create() により返されたパラメータです。

返り値

成功した場合に辞書リンク ID を返します。

例

Example#1 pspell_new_config()

```
<?php
$pspell_config = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config($pspell_config);
?>
```

---

pspell_new_personal

(PHP 4 >= 4.0.2, PHP 5)

pspell_new_personal — 個人の単語リストを有する新規辞書をロードする

説明

int **pspell_new_personal** (string $personal , string $language [, string $spelling [, string $jargon [, string $encoding [, int $mode]]]])

pspell_new_personal() は、個人の単語リストと共に 新規辞書をオープンし、辞書リンクIDを返します。この ID は他の pspell 関数で使用されます。単語リストは修正可能で、必要に応じて pspell_save_wordlist() で保存することも可能です。 しかし、置換の組は保存されません。置換の組を保存するには、 pspell_config_create() を用いて設定を作成し、 pspell_config_personal() で個人の単語リストを 設定し、 pspell_config_repl() で置換のファイルを 設定し、pspell_new_config() で新規の辞書を オープンします。

詳細な情報および例については、pspell の Web サイト :» http://aspell.net/ にあるインラインマニュアルを参照ください。

パラメータ

personal

個人リストに追加された単語が保存される場所を指定します。 この場所には '/' で始まる絶対パスのファイル名を指定する必要があります。 そうでない場合は $HOME からの相対パスになりますが、 これは多くのシステムでは、"/root" であり、 おそらく望ましい結果とはならないためです。

language

2 文字の ISO 639 言語コードと オプションでダッシュまたはアンダースコアの後に 2 文字の ISO 3166 国コードからなる言語コードです。

spelling

英語のように複数のスペルがある言語に関して スペルの指定を行うものです。指定可能な値は、 'american', 'british', 'canadian'です。

jargon

同じ language および spelling パラメータを有する 2 つの異なる単語リストを区別するための 追加情報を有しています。

encoding

単語のエンコーディングとして予想されるもの。有効な値は、 utf-8, iso8859-, koi8-r, viscii, cp1252, machine unsigned 16, machine unsigned 32 です。*

mode

スペルチェッカの動作モードです。 使用可能なモードを以下に示します。

- ***PSPELL_FAST*** *- 高速モード (修正候補の数は最小)*
- ***PSPELL_NORMAL*** *- 通常モード (修正候補はより多い)*
- ***PSPELL_BAD_SPELLERS*** *- 低速モード (修正候補は多い)*
- ***PSPELL_RUN_TOGETHER*** *- つながった単語を複合語 (legal compound) として考慮します。この場合、"thecat" には二つの 単語の間に空白はありませんが複合語となります。この設定の変更は pspell_check() から返される結果にのみ影響します。 設定変更後も pspell_suggest() は修正候補を返します。*

*mode は、これらのさまざまな定数を用いたビットマスクです。 しかし **PSPELL_FAST**、 **PSPELL_NORMAL** および **PSPELL_BAD_SPELLERS** は相反するため、 この中のひとつを選択する必要があります。*

返り値

その他の pspell 関数で使用する辞書リンク ID を返します。

例

Example#1 pspell_new_personal()

```
<?php
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
        "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER);
?>
```

---

pspell_new

(PHP 4 >= 4.0.2, PHP 5)

pspell_new — 新規辞書をロードする

説明

*int **pspell_new** (string $language [, string $spelling [, string $jargon [, string $encoding [, int $mode]]]])*

***pspell_new()** は、新規の辞書をロードして 辞書リンク ID を返します。このリンク ID は、他の pspell 関数で使用されます。*

詳細な情報および例については、pspell の Web サイト :» http://aspell.net/ にあるインラインマニュアルを参照ください。

パラメータ

language

パラメータ language は、2 文字の ISO 639 言語コードと オプションでダッシュまたはアンダースコアの後に 2 文字の ISO 3166 国コードからなる言語コードです。

spelling

パラメータ spelling は、英語のように複数のスペルがある言語に関して スペルの指定を行うものです。指定可能な値は、 'american', 'british', 'canadian'です。

jargon

パラメータ jargon は、同じ language および spelling パラメータを有する 2 つの異なる単語リストを区別するための 追加情報を有しています。

encoding

パラメータ encoding は、単語のエンコーディングとして 予想されるものです。有効な値は、'utf-8', 'iso8859-', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32' です。このパラメータはよくテストされていないため、 使用する際には注意してください。*

mode

パラメータ mode は、スペルチェッカの動作モードです。 使用可能なモードを以下に示します。

- ***PSPELL_FAST*** *- 高速モード (修正候補の数は最小)*
- ***PSPELL_NORMAL*** *- 通常モード (修正候補はより多い)*
- ***PSPELL_BAD_SPELLERS*** *- 低速モード (修正候補は多い)*
- ***PSPELL_RUN_TOGETHER*** *- つながった単語を複合語 (legal compound) として考慮します。この場合、"thecat" には二つの 単語の間に空白はありませんが複合語となります。この設定の変更は pspell_check() から返される結果にのみ影響します。 設定変更後も pspell_suggest() は修正候補を返します。*

*mode は、これらのさまざまな定数を用いたビットマスクです。 しかし **PSPELL_FAST**、 **PSPELL_NORMAL** および **PSPELL_BAD_SPELLERS** は相反するため、 この中のひとつを選択する必要があります。 Mode is a bitmask constructed from different constants listed above. However, **PSPELL_FAST**, **PSPELL_NORMAL** and **PSPELL_BAD_SPELLERS** are mutually exclusive, so you should select only one of them.*

返り値

*Returns the dictionary link identifier on success, or **FALSE** on failure.*

例

Example#1 pspell_new()

```
<?php
```

```
$pspell_link  = pspell_new("en", "", "", "",
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
?>
```

---

pspell_save_wordlist

(PHP 4 >= 4.0.2, PHP 5)

pspell_save_wordlist — 個人の単語リストをファイルに保存する

説明

bool **pspell_save_wordlist** (int $dictionary_link)

pspell_save_wordlist() は、 現在のセッションから個人の単語リストを保存します。 ファイルの位置は pspell_config_personal() および (オプションで) pspell_config_repl() で指定されている必要があります。

パラメータ

dictionary_link

pspell_new_personal() でオープンした辞書リンク ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_add_to_personal()

```
<?php
$pspell_config  = pspell_config_create("en");
pspell_config_personal($pspell_config, "/tmp/dicts/newdict");
$pspell_link  = pspell_new_config($pspell_config);

pspell_add_to_personal($pspell_link, "Vlad");
pspell_save_wordlist($pspell_link);
?>
```

注意

> **注意**: この関数は、pspell .11.2 および aspell .32.5 以降がない限り動作しないことに注意してください。

---

pspell_store_replacement

(PHP 4 >= 4.0.2, PHP 5)

pspell_store_replacement — 単語を置換する組を保存する

説明

bool **pspell_store_replacement** (int $dictionary_link , string $misspelled , string $correct)

pspell_store_replacement() は、単語の置換候補を 保存します。これにより、この後の pspell_suggest() で置換候補を返すことが可能と なります。この関数の利点を活かすためには、辞書をオープンする際に pspell_new_personal() を使用する必要があります。 置換候補を恒久的に保存するためには pspell_config_personal() を使用する必要があり、 pspell_config_repl() によりカスタム単語リストを 保存するパスを設定し、この後、変更点をディスクへ書き込むために pspell_save_wordlist() を使用する必要があります。

パラメータ

dictionary_link

pspell_new_personal() でオープンした辞書リンク ID。

misspelled

まちがったスペルの単語。

correct

misspelled を修正した単語。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 pspell_store_replacement()

```
<?php
$pspell_config  = pspell_config_create("en");
pspell_config_personal($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link  = pspell_new_config($pspell_config);
```

```
pspell_store_replacement($pspell_link, $misspelled, $correct);
pspell_save_wordlist($pspell_link);
?>
```

注意

注意: この関数は、pspell .11.2 および aspell .32.5 以降でない限り 動作しないことに注意してください。

---

pspell_suggest

(PHP 4 >= 4.0.2, PHP 5)

pspell_suggest — 単語のスペルについて修正候補を示す

説明

array **pspell_suggest** (int $dictionary_link , string $word)

pspell_suggest() は、 指定した単語について可能性のあるスペルの配列を返します。

パラメータ

dictionary_link

word

調べたい単語。

返り値

修正候補の配列を返します。

例

Example#1 pspell_suggest() の例

```
<?php
$pspell_link = pspell_new("en");

if (!pspell_check($pspell_link, "testt")) {
    $suggestions = pspell_suggest($pspell_link, "testt");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br />";
    }
}
?>
```

目次

---

qtdom 関数

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な

PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

注意: この拡張モジュールは Windows 環境では利用できません。

注意: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 5.0.0.

要件

Qt ライブラリ >=2.2.0 が必要です。

インストール手順

以下の関数を使用するには、PHP --with-qtdom を configure に指定します。

---

qdom_error

(PHP 4 >= 4.0.5)

qdom_error — 直近の QDOM 操作からのエラー文字列、 またはエラーが発生しなかった場合に FALSE を返す

説明

string **qdom_error** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

qdom_tree

(PHP 4 >= 4.0.4)

qdom_tree — XML 文字列のツリーを作成する

説明

QDomDocument **qdom_tree** (string $doc)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

Radius

導入

このパッケージは、FreeBSD の libradius をもとにしたものです。 この PECL モジュールは、Radius 認証 (» RFC 2865) および Radius 課金 (» RFC 2866) を完全にサポートします。 このパッケージは、Unix (FreeBSD および Linux でテストしました) および Windows で使用可能です。

注意: libradius に関する正確な説明は» ここ にあります。設定ファイルについての詳細な説明は » ここです。

インストール手順

どうやってインストールするの?

- tar パッケージを (通常は php4/ext に) 展開する
- radius-x.x の名前を radius に変更する
- php4 ディレクトリで ./buildconf を実行する
- ./configure --enable-radius を実行する
- make; make install

あるいは、もし .so 形式にしたければ

- tar パッケージを展開する
- radius-x.x ディレクトリで phpize を実行する
- radius-x.x ディレクトリで ./configure を実行する
- make; make install

Windows の場合は、 » http://snaps.php.net/ から php_radius.dll をダウンロードして使用することを推奨します。 バンドルされていない PECL 拡張モジュールは » http://pecl4win.php.net/ からダウンロードできます。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

RADIUS_ACCESS_REQUEST ()
認証のリクエスト
RADIUS_ACCESS_ACCEPT ()
アクセスが許可されました
RADIUS_ACCESS_REJECT ()
アクセスが拒否されました
RADIUS_ACCOUNTING_REQUEST ()
課金のリクエスト
RADIUS_ACCOUNTING_RESPONSE ()
課金のレスポンス
RADIUS_ACCESS_CHALLENGE ()
アクセスチャレンジ
RADIUS_USER_NAME (string)
ユーザ名
RADIUS_USER_PASSWORD (string)
パスワード
RADIUS_CHAP_PASSWORD (string)
Chap パスワード。 chappass = md5(ident + plaintextpass + challenge)
RADIUS_NAS_IP_ADDRESS (string)
NAS IP アドレス
RADIUS_NAS_PORT (int)
NAS ポート
RADIUS_SERVICE_TYPE (int)

サービスの型、以下のいずれか

- **RADIUS_LOGIN**
- **RADIUS_FRAMED**
- **RADIUS_CALLBACK_LOGIN**
- **RADIUS_CALLBACK_FRAMED**
- **RADIUS_OUTBOUND**
- **RADIUS_ADMINISTRATIVE**
- **RADIUS_NAS_PROMPT**
- **RADIUS_AUTHENTICATE_ONLY**
- **RADIUS_CALLBACK_NAS_PROMPT**

RADIUS_FRAMED_PROTOCOL (int)

フレームのプロトコル、以下のいずれか

- **RADIUS_PPP**
- **RADIUS_SLIP**
- **RADIUS_ARAP**
- **RADIUS_GANDALF**
- **RADIUS_XYLOGICS**

RADIUS_FRAMED_IP_ADDRESS (string)
IP アドレス
RADIUS_FRAMED_IP_NETMASK (string)
ネットマスク
RADIUS_FRAMED_ROUTING (int)
ルーティング
RADIUS_FILTER_ID (string)
フィルタ ID
RADIUS_FRAMED_MTU (int)
MTU
RADIUS_FRAMED_COMPRESSION (int)

圧縮、以下のいずれか

- **RADIUS_COMP_NONE**
- **RADIUS_COMP_VJ**
- **RADIUS_COMP_IPXHDR**

RADIUS_LOGIN_IP_HOST (string)
ログイン IP ホスト
RADIUS_LOGIN_SERVICE (int)
ログインサービス
RADIUS_LOGIN_TCP_PORT (int)
ログイン TCP ポート
RADIUS_REPLY_MESSAGE (string)
応答メッセージ
RADIUS_CALLBACK_NUMBER (string)
コールバック番号
RADIUS_CALLBACK_ID (string)
コールバック ID
RADIUS_FRAMED_ROUTE (string)
フレームのルート
RADIUS_FRAMED_IPX_NETWORK (string)
フレームの IPX ネットワーク
RADIUS_STATE (string)
状態
RADIUS_CLASS (int)
クラス

RADIUS_VENDOR_SPECIFIC (*int*)
ベンダ固有の属性
RADIUS_SESSION_TIMEOUT (*int*)
セッションタイムアウト
RADIUS_IDLE_TIMEOUT (*int*)
アイドルタイムアウト
RADIUS_TERMINATION_ACTION (*int*)
停止アクション
RADIUS_CALLED_STATION_ID (*int*)
呼び出し先ステーション ID
RADIUS_CALLING_STATION_ID (*string*)
呼び出し元ステーション ID
RADIUS_NAS_IDENTIFIER (*int*)
NAS ID
RADIUS_PROXY_STATE (*int*)
Proxy の状態
RADIUS_LOGIN_LAT_SERVICE (*int*)
ログイン LAT サービス
RADIUS_LOGIN_LAT_NODE (*int*)
ログイン LAT ノード
RADIUS_LOGIN_LAT_GROUP (*int*)
ログイン LAT グループ
RADIUS_FRAMED_APPLETALK_LINK (*int*)
フレームの Appletalk リンク
RADIUS_FRAMED_APPLETALK_NETWORK (*int*)
フレームの Appletalk ネットワーク
RADIUS_FRAMED_APPLETALK_ZONE (*int*)
フレームの Appletalk ゾーン
RADIUS_CHAP_CHALLENGE (*string*)
チャレンジ
RADIUS_NAS_PORT_TYPE (*int*)

NAS ポート型、以下のいずれか

- ***RADIUS_ASYNC***
- ***RADIUS_SYNC***
- ***RADIUS_ISDN_SYNC***
- ***RADIUS_ISDN_ASYNC_V120***
- ***RADIUS_ISDN_ASYNC_V110***
- ***RADIUS_VIRTUAL***
- ***RADIUS_PIAFS***
- ***RADIUS_HDLC_CLEAR_CHANNEL***
- ***RADIUS_X_25***
- ***RADIUS_X_75***
- ***RADIUS_G_3_FAX***
- ***RADIUS_SDSL***
- ***RADIUS_ADSL_CAP***
- ***RADIUS_ADSL_DMT***
- ***RADIUS_IDSL***
- ***RADIUS_ETHERNET***
- ***RADIUS_XDSL***
- ***RADIUS_CABLE***
- ***RADIUS_WIRELESS_OTHER***
- ***RADIUS_WIRELESS_IEEE_802_11***

RADIUS_PORT_LIMIT (*int*)
ポートの限界
RADIUS_LOGIN_LAT_PORT (*int*)
ログイン LAT ポート
RADIUS_CONNECT_INFO (*string*)
接続の情報
RADIUS_ACCT_STATUS_TYPE (*int*)

課金状態の型、以下のいずれか

- ***RADIUS_START***
- ***RADIUS_STOP***
- ***RADIUS_ACCOUNTING_ON***
- ***RADIUS_ACCOUNTING_OFF***

RADIUS_ACCT_DELAY_TIME (*int*)
課金の遅延時間
RADIUS_ACCT_INPUT_OCTETS (*int*)
課金の入力バイト数
RADIUS_ACCT_OUTPUT_OCTETS (*int*)
課金の出力バイト数
RADIUS_ACCT_SESSION_ID (*int*)
課金のセッション ID
RADIUS_ACCT_AUTHENTIC (*int*)

課金認証、以下のいずれか

- ***RADIUS_AUTH_RADIUS***
- ***RADIUS_AUTH_LOCAL***
- ***RADIUS_AUTH_REMOTE***

RADIUS_ACCT_SESSION_TIME (*int*)
課金のセッション時間
RADIUS_ACCT_INPUT_PACKETS (*int*)
課金の入力パケット
RADIUS_ACCT_OUTPUT_PACKETS (*int*)
課金の出力パケット
RADIUS_ACCT_TERMINATE_CAUSE (*int*)

課金終了の原因、以下のいずれか

- ***RADIUS_TERM_USER_REQUEST***
- ***RADIUS_TERM_LOST_CARRIER***
- ***RADIUS_TERM_LOST_SERVICE***
- ***RADIUS_TERM_IDLE_TIMEOUT***
- ***RADIUS_TERM_SESSION_TIMEOUT***

- RADIUS_TERM_ADMIN_RESET
- RADIUS_TERM_ADMIN_REBOOT
- RADIUS_TERM_PORT_ERROR
- RADIUS_TERM_NAS_ERROR
- RADIUS_TERM_NAS_REQUEST
- RADIUS_TERM_NAS_REBOOT
- RADIUS_TERM_PORT_UNNEEDED
- RADIUS_TERM_PORT_PREEMPTED
- RADIUS_TERM_PORT_SUSPENDED
- RADIUS_TERM_SERVICE_UNAVAILABLE
- RADIUS_TERM_CALLBACK
- RADIUS_TERM_USER_ERROR
- RADIUS_TERM_HOST_REQUEST

RADIUS_ACCT_MULTI_SESSION_ID (string)
課金のマルチセッション ID
RADIUS_ACCT_LINK_COUNT (int)
課金のリンク数
RADIUS_VENDOR_MICROSOFT (int)

Microsoft 固有のベンダ属性 (» RFC 2548)、 以下のいずれか

- RADIUS_MICROSOFT_MS_CHAP_RESPONSE
- RADIUS_MICROSOFT_MS_CHAP_ERROR
- RADIUS_MICROSOFT_MS_CHAP_PW_1
- RADIUS_MICROSOFT_MS_CHAP_PW_2
- RADIUS_MICROSOFT_MS_CHAP_LM_ENC_PW
- RADIUS_MICROSOFT_MS_CHAP_NT_ENC_PW
- RADIUS_MICROSOFT_MS_MPPE_ENCRYPTION_POLICY
- RADIUS_MICROSOFT_MS_MPPE_ENCRYPTION_TYPES
- RADIUS_MICROSOFT_MS_RAS_VENDOR
- RADIUS_MICROSOFT_MS_CHAP_DOMAIN
- RADIUS_MICROSOFT_MS_CHAP_CHALLENGE
- RADIUS_MICROSOFT_MS_CHAP_MPPE_KEYS
- RADIUS_MICROSOFT_MS_BAP_USAGE
- RADIUS_MICROSOFT_MS_LINK_UTILIZATION_THRESHOLD
- RADIUS_MICROSOFT_MS_LINK_DROP_TIME_LIMIT
- RADIUS_MICROSOFT_MS_MPPE_SEND_KEY
- RADIUS_MICROSOFT_MS_MPPE_RECV_KEY
- RADIUS_MICROSOFT_MS_RAS_VERSION
- RADIUS_MICROSOFT_MS_OLD_ARAP_PASSWORD
- RADIUS_MICROSOFT_MS_NEW_ARAP_PASSWORD
- RADIUS_MICROSOFT_MS_ARAP_PASSWORD_CHANGE_REASON
- RADIUS_MICROSOFT_MS_FILTER
- RADIUS_MICROSOFT_MS_ACCT_AUTH_TYPE
- RADIUS_MICROSOFT_MS_ACCT_EAP_TYPE
- RADIUS_MICROSOFT_MS_CHAP2_RESPONSE
- RADIUS_MICROSOFT_MS_CHAP2_SUCCESS
- RADIUS_MICROSOFT_MS_CHAP2_PW
- RADIUS_MICROSOFT_MS_PRIMARY_DNS_SERVER
- RADIUS_MICROSOFT_MS_SECONDARY_DNS_SERVER
- RADIUS_MICROSOFT_MS_PRIMARY_NBNS_SERVER
- RADIUS_MICROSOFT_MS_SECONDARY_NBNS_SERVER
- RADIUS_MICROSOFT_MS_ARAP_CHALLENGE

クイックスタート

どのように使用するの?

- radius リソースを取得する
- ライブラリを設定する
- リクエストを作成する
- 属性を設定する
- リクエストを送信する
- 属性を受け取る
- radius リソースを閉じる (オプション)

パッケージに含まれている例もごらんください。

このパッケージにはサンプル PHP スクリプトが含まれています。 このスクリプトでは、PAP あるいは CHAP (md5) を使用した radius 認証の 方法を説明しています。Microsoft Radius サーバを使用して認証を行う 場合は、CHAP (md5) を使用することはできません。Microsoft のサーバを使用する場合は、MS-CHAPv1 あるいは MS-CHAPv2 を使用する必要があります。 しかし、正しいデータを生成するために md4、sha1 および des が必要となる こともあり、これは複雑な手順となります。同梱されている例では すべての認証方式を説明しており、そこには MS-CHAPv1 および MS-CHAPv2 も含まれています。MS-CHAP を動作させるためには、拡張モジュール mcrypt および mhash が必要となります。 バージョン 1.2 以降では、mcrypt 拡張モジュールは必要なくなりました。

連絡先の情報

コメント・バグフィックス・機能拡張・あるいは開発を手伝いたいなどの場合は、 メールを » mbretter@php.net に送ってください。Windows 用のバイナリは » ここから ダウンロードできます。

---

radius_acct_open

(PECL radius:1.1-1.2.5)

radius_acct_open — 課金用の Radius ハンドルを作成する

説明

resource **radius_acct_open** (void)

返り値

成功した場合にハンドル、エラー時に **FALSE** を返します。 この関数が失敗するのは、無効なメモリを使用した場合のみです。

例

Example#1 radius_acct_open() の例

```
<?php
$res = radius_acct_open ()
    or die ("ハンドルを作成できませんでした");
print("ハンドルの作成に成功しました");
?>
```

---

radius_add_server

(PECL radius:1.1-1.2.5)

radius_add_server — サーバを追加する

説明

bool **radius_add_server** (resource $radius_handle , string $hostname , int $port , string $secret , int $timeout , int $max_tries)

radius_add_server() は複数回コールされることも ありえます。また、radius_config() とともに 使用されることでしょう。最大で 10 までのサーバを指定できます。 複数のサーバが指定されると、有効なレスポンスを受信するか あるいは各サーバへの接続回数が max_tries をこえるまでラウンドロビン形式でのアクセスを試みます。

パラメータ

radius_handle

hostname

hostname はサーバのホストを FQDN あるいはドット区切りの IP アドレス形式で指定します。

port

port は、サーバとの接続に使用する UDP ポートを指定します。0 を指定すると、このライブラリは ネットワークサービスデータベースから radius/udp あるいは radacct/udp サービスを 検索し、見つかったポートを使用します。見つからなかった場合は 標準の Radius ポート、すなわち認証には 1812、 課金には 1813 を使用します。

secret

secret パラメータには、サーバホストに 対する共有秘密鍵が渡されます。Radius プロトコルは、共有秘密鍵の 最初の 128 バイト以外を無視します。

timeout

timeout パラメータには、サーバから 応答を受信する際のタイムアウトを秒単位で指定します。

max_tries

結果が返ってこなかった場合に最大何回までリクエストを繰り返すかを max_tries に指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 radius_add_server() の例

```
<?php
if (!radius_add_server($res, 'radius.example.com', 1812, 'testing123', 3, 3)) {
    echo 'RadiusError:' . radius_strerror($res). "¥n<br>";
    exit;
}
?>
```

参考

- radius_config()

---

radius_auth_open

(PECL radius:1.1-1.2.5)

radius_auth_open — 認証用の Radius ハンドルを作成する

説明

resource **radius_auth_open** (void)

返り値

成功した場合にハンドル、エラー時に **FALSE** を返します。 この関数が失敗するのは、無効なメモリを使用した場合のみです。

例

Example#1 radius_auth_open() の例

```
<?php
$radh = radius_auth_open()
    or die ("ハンドルを作成できませんでした");
echo "ハンドルの作成に成功しました";
?>
```

---

radius_close

(PECL radius:1.1-1.2.5)

radius_close — すべてのリソースを開放する

説明

bool **radius_close** (resource $radius_handle)

各リクエストの終了時に PHP がすべてのリソースを開放するので、 この関数をコールする必要はありません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

radius_config

(PECL radius:1.1-1.2.5)

radius_config — 指定した設定ファイルをライブラリに読み込ませる

説明

bool **radius_config** (resource $radius_handle , string $file)

Radius リクエストを発行する前には、ライブラリが接続可能なサーバを 知っている必要があります。ライブラリを設定するいちばん簡単な方法は **radius_config()** をコールすることです。 **radius_config()** は、ライブラリに » radius.conf 形式のファイルを読み込ませます。

パラメータ

radius_handle

file

設定ファイルへのパスを、引数として **radius_config()** に渡します。 radius_add_server() をコールすることで、 プログラム上でライブラリの設定をすることも可能です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- radius_add_server()

---

radius_create_request

(PECL radius:1.1-1.2.5)

radius_create_request — 課金あるいは認証のリクエストを作成する

説明

bool **radius_create_request** (resource $radius_handle , int $type)

Radius リクエストには、リクエストの種類を指定するコードおよび 追加情報を指定するゼロ個以上の属性が含まれます。新しいリクエストを 作成するには、**radius_create_request()** をコールします。

> **注意**: 注意: 属性を設定する前にこの関数をコールする必要があります。

パラメータ

radius_handle

RADIUS_ACCESS_REQUEST あるいは **RADIUS_ACCOUNTING_REQUEST** のいずれか。

type

その説明

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 radius_create_request() の例

```
<?php
if (!radius_create_request($res, RADIUS_ACCESS_REQUEST)) {
    echo 'RadiusError:' . radius_strerror($res). "¥n<br />";
    exit;
}
?>
```

参考

- radius_send_request()

---

radius_cvt_addr

(PECL radius:1.1-1.2.5)

radius_cvt_addr — 生データを IP アドレスに変換する

説明

string **radius_cvt_addr** (string $data)

例

Example#1 radius_cvt_addr() の例

```
<?php
while ($resa = radius_get_attr($res)) {

    if (!is_array($resa)) {
        printf ("属性取得時のエラー: %s\n", radius_strerror($res));
        exit;
    }

    $attr = $resa['attr'];
    $data = $resa['data'];

    switch ($attr) {

    case RADIUS_FRAMED_IP_ADDRESS:
        $ip = radius_cvt_addr($data);
        echo "IP: $ip<br>¥n";
        break;

    case RADIUS_FRAMED_IP_NETMASK:
        $mask = radius_cvt_addr($data);
        echo "マスク: $mask<br>\n";
        break;
    }
}
?>
```

参考

- radius_cvt_int()
- radius_cvt_string()

---

radius_cvt_int

(PECL radius:1.1-1.2.5)

radius_cvt_int — 生データを整数に変換する

説明

int **radius_cvt_int** (string $data)

例

Example#1 radius_cvt_int() の例

```
<?php
while ($resa = radius_get_attr($res)) {

    if (!is_array($resa)) {
        printf ("属性取得時のエラー: %s\n", radius_strerror($res));
        exit;
    }
```

```
    $attr = $resa['attr'];
    $data = $resa['data'];

    switch ($attr) {

    case RADIUS_FRAMED_MTU:
        $mtu = radius_cvt_int($data);
        echo "MTU: $mtu<br>¥n";
        break;
    }
}
?>
```

参考

- radius_cvt_addr()
- radius_cvt_string()

---

radius_cvt_string

(PECL radius:1.1-1.2.5)

radius_cvt_string — 生データを文字列に変換する

説明

string **radius_cvt_string** (string $data)

例

Example#1 radius_cvt_string() の例

```
<?php
while ($resa = radius_get_attr($res)) {

    if (!is_array($resa)) {
        printf ("属性取得時のエラー: %s\n", radius_strerror($res));
        exit;
    }

    $attr = $resa['attr'];
    $data = $resa['data'];

    switch ($attr) {

    case RADIUS_FILTER_ID:
        $id = radius_cvt_string($data);
        echo "フィルタ ID: $id<br>\n";
        break;
    }
}
?>
```

参考

- radius_cvt_addr()
- radius_cvt_int()

---

radius_demangle_mppe_key

(PECL radius:1.2-1.2.5)

radius_demangle_mppe_key — 変形されたデータから mppe キーを得る

説明

string **radius_demangle_mppe_key** (resource $radius_handle , string $mangled)

MS-CHAPv2 で MPPE を使用している場合には、送信キーおよび受信キーが 変形されます (» RFC 2548 を参照ください)。しかしこの関数は無意味です。なぜなら PHP では PPTP-MPPE は実装されていないし、今後も実装されるとは思えないからです。

返り値

復元したデータ、あるいはエラー時には **FALSE** を返します。

---

radius_demangle

(PECL radius:1.2-1.2.5)

radius_demangle — データを復元する

説明

string ***radius_demangle*** *(resource $radius_handle , string $mangled)*

セキュリティ上の理由により、(パスワード、MS-CHAPv1 MPPE キーなど) いくつかのデータは変形されます。それを使用する際には事前に復元する必要があります。

返り値

復元した文字列、あるいはエラー時には ***FALSE*** *を返します。*

---

radius_get_attr

(PECL radius:1.1-1.2.5)

radius_get_attr — 属性を取得する

説明

mixed ***radius_get_attr*** *(resource $radius_handle)*

Radius リクエストと同様、各レスポンスもゼロ個以上の属性を含んでいます。radius_send_request() でレスポンスを受け取ったら、 ***radius_get_attr()*** *を使用して各属性を抽出することが できます。* ***radius_get_attr()*** *がコールされるたびに、 現在のレスポンスから次の属性を取得します。*

返り値

属性の型とデータを含む連想配列、あるいは 0 以上の エラー番号を返します。

例

Example#1 radius_get_attr() の例

```
<?php
while ($resa = radius_get_attr($res)) {

    if (!is_array($resa)) {
        printf("属性取得エラー: %s\n", radius_strerror($res));
        exit;
    }

    $attr = $resa['attr'];
    $data = $resa['data'];
    printf("属性を取得:%d %d Bytes %s\n", $attr, strlen($data), bin2hex($data));
}
?>
```

参考

- *radius_put_attr()*
- *radius_get_vendor_attr()*
- *radius_put_vendor_attr()*
- *radius_send_request()*

---

radius_get_vendor_attr

(PECL radius:1.1-1.2.5)

radius_get_vendor_attr — ベンダ固有の属性を取得する

説明

array ***radius_get_vendor_attr*** *(string $data)*

radius_get_attr() が ***RADIUS_VENDOR_SPECIFIC*** *を返す場合に、 ベンダを特定するために* ***radius_get_vendor_attr()*** *をコールします。*

返り値

属性の型、ベンダおよびデータを含む連想配列か、エラー時には ***FALSE*** *を返します。*

例

Example#1 radius_get_vendor_attr() の例

```
<?php
while ($resa = radius_get_attr($res)) {

    if (!is_array($resa)) {
        printf ("属性取得エラー: %s\n", radius_strerror($res));
        exit;
    }

    $attr = $resa['attr'];
    $data = $resa['data'];
    printf("属性を取得:%d %d Bytes %s\n", $attr, strlen($data), bin2hex($data));
    if ($attr == RADIUS_VENDOR_SPECIFIC) {
```

```
        $resv = radius_get_vendor_attr($data);
        if (is_array($resv)) {
            $vendor = $resv['vendor'];
            $attrv = $resv['attr'];
            $datav = $resv['data'];
            printf("ベンダ属性を取得:%d %d Bytes %s\n", $attrv, strlen($datav), bin2hex($datav));
        }

    }
}
?>
```

参考

- radius_get_attr()
- radius_put_vendor_attr()

---

radius_put_addr

(PECL radius:1.1-1.2.5)

radius_put_addr — IP アドレス属性を設定する

説明

bool **radius_put_addr** (resource $radius_handle , int $type , string $addr)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

radius_put_attr

(PECL radius:1.1-1.2.5)

radius_put_attr — バイナリ属性を設定する

説明

bool **radius_put_attr** (resource $radius_handle , int $type , string $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 radius_put_attr() の例

```
<?php
mt_srand(time());
$chall = mt_rand();
$chapval = md5(pack('Ca*',1 , 'sepp' . $chall));
$pass = pack('CH*', 1, $chapval);
if (!radius_put_attr($res, RADIUS_CHAP_PASSWORD, $pass)) {
    echo 'RadiusError:' . radius_strerror($res). "¥n<br />";
    exit;
}
?>
```

参考

- radius_get_attr()
- radius_get_vendor_attr()
- radius_put_vendor_attr()

---

radius_put_int

(PECL radius:1.1-1.2.5)

radius_put_int — 整数属性を設定する

説明

bool **radius_put_int** (resource $radius_handle , int $type , int $value)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** *を、失敗した場合に* **FALSE** *を返します。*

例

Example#1 radius_put_int() の例

```
<?php
if (!radius_put_int($res, RAD_FRAMED_PROTOCOL, RAD_PPP)) {
    echo 'RadiusError:' . radius_strerror($res). "¥n<br />";
    exit;
}
?>
```

参考

- radius_put_string()
- radius_put_vendor_int()
- radius_put_vendor_string()

---

radius_put_string

(PECL radius:1.1-1.2.5)

radius_put_string — 文字列属性を設定する

説明

bool **radius_put_string** (resource $radius_handle , int $type , string $value)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** *を、失敗した場合に* **FALSE** *を返します。*

例

Example#1 radius_put_string() の例

```
<?php
if (!radius_put_string($res, RADIUS_USER_NAME, 'billy')) {
     echo 'RadiusError:' . radius_strerror($res). "¥n<br />";
     exit;
}
?>
```

参考

- radius_put_int()
- radius_put_vendor_int()
- radius_put_vendor_string()

---

radius_put_vendor_addr

(PECL radius:1.1-1.2.5)

radius_put_vendor_addr — ベンダ固有の IP アドレス属性を設定する

説明

bool **radius_put_vendor_addr** (resource $radius_handle , int $vendor , int $type , string $addr)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** *を、失敗した場合に* **FALSE** *を返します。*

---

radius_put_vendor_attr

(PECL radius:1.1-1.2.5)

radius_put_vendor_attr — ベンダ固有のバイナリ属性を設定する

説明

bool **radius_put_vendor_attr** (resource $radius_handle , int $vendor , int $type , string $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 radius_put_vendor_attr() の例

```
<?php
if (!radius_put_vendor_attr($res, RADIUS_VENDOR_MICROSOFT, RAD_MICROSOFT_MS_CHAP_CHALLENGE, $challenge)) {
    echo 'RadiusError:' . radius_strerror($res). "¥n<br />";
    exit;
}
?>
```

---

radius_put_vendor_int

(PECL radius:1.1-1.2.5)

radius_put_vendor_int — ベンダ固有の整数属性を設定する

説明

bool **radius_put_vendor_int** (resource $radius_handle , int $vendor , int $type , int $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

radius_put_vendor_string

(PECL radius:1.1-1.2.5)

radius_put_vendor_string — ベンダ固有の文字列属性を設定する

説明

bool **radius_put_vendor_string** (resource $radius_handle , int $vendor , int $type , string $value)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

radius_request_authenticator

(PECL radius:1.1-1.2.5)

radius_request_authenticator — リクエスト認証子を返す

説明

string **radius_request_authenticator** (resource $radius_handle)

パスワードや暗号化キーのような加工されたデータを復元する際には リクエスト認証子が必要となります。

返り値

リクエスト認証子を文字列で返します。エラー時には **FALSE** を返します。

参考

- radius_demangle()

---

radius_send_request

(PECL radius:1.1-1.2.5)

radius_send_request — リクエストを送信し、応答を待つ

説明

int **radius_send_request** (resource $radius_handle)

Radius リクエストを作成した後は、 **radius_send_request()** でそれを送信します。

radius_send_request() 関数は、リクエストを送信して 応答を待ちます。定義されているサーバ間で、必要に応じてラウンドロビン 形式で再試行します。

返り値

有効な応答を受信したら、**radius_send_request()** は応答の型を示す Radius コードを返します。一般的なコードは **RADIUS_ACCESS_ACCEPT**、**RADIUS_ACCESS_REJECT** あるいは **RADIUS_ACCESS_CHALLENGE** です。 有効な応答を受信できなかった場合は、 **radius_send_request()** は **FALSE** を返します。

参考

- radius_create_request()

---

radius_server_secret

(PECL radius:1.1-1.2.5)

radius_server_secret — 共有秘密鍵を返す

説明

string **radius_server_secret** (resource $radius_handle)

パスワードや暗号化キーのような加工されたデータを復元する際の SALT として、共有秘密鍵が必要となります。

返り値

サーバの共有秘密鍵を文字列で返します。エラー時には **FALSE** を返します。

---

radius_strerror

(PECL radius:1.1-1.2.5)

radius_strerror — エラーメッセージを返す

説明

string **radius_strerror** (resource $radius_handle)

Radius 関数が失敗した場合にはエラーメッセージが記録されます。 この関数により、エラーメッセージを取得することが可能となります。

返り値

失敗した radius 関数からのエラーメッセージを文字列で返します。

目次

- radius_put_int — 整数属性を設定する
- radius_put_string — 文字列属性を設定する
- radius_put_vendor_addr — ベンダ固有の IP アドレス属性を設定する
- radius_put_vendor_attr — ベンダ固有のバイナリ属性を設定する
- radius_put_vendor_int — ベンダ固有の整数属性を設定する
- radius_put_vendor_string — ベンダ固有の文字列属性を設定する
- radius_request_authenticator — リクエスト認証子を返す
- radius_send_request — リクエストを送信し、応答を待つ
- radius_server_secret — 共有秘密鍵を返す
- radius_strerror — エラーメッセージを返す

Rar 関数

導入

RAR は、Eugene Roshal が作成した強力で効率的なアーカイバです。 この拡張モジュールは unRAR ライブラリを使用していますが、 このライブラリは Rar アーカイブの読み込みしかサポートしていません。 unRAR のソースを使用して RAR/WinRAR アーカイバを作成することは、 unRAR のライセンス上で禁じられています。

RAR および unRAR についてのより詳細な情報は » http://www.rarlabs.com/ で見つけられます。

要件

外部ライブラリを必要としません。

実行時設定

設定ディレクティブは定義されていません。

インストール手順

Rar 拡張モジュールは、現在 PECL » http://pecl.php.net/package/rar で取得可能です。

また、以下のコマンドを使用して PECL インストーラで Rar 拡張モジュールを インストールすることができます。**pecl -v install rar**

tar.gz パッケージをダウンロードし、Rar を手動でインストールすることも 可能です。

Example#1 Rar のインストール

```
gunzip  rar-xxx.tgz
tar  -xvf  rar-xxx.tar
cd  rar-xxx
phpize
./configure  &&  make  &&  make  install
```

Windows ユーザは、拡張 DLL php_rar.dll をここから ダウンロードします。» http://snaps.php.net/win32/PECL_STABLE/

リソース型

Rar 拡張モジュールが使用するリソースはひとつ、すなわち rar_open() が返すファイル記述子です。

定義済み定数

RAR_HOST_MSDOS (integer)
RAR_HOST_OS2 (integer)
RAR_HOST_WIN32 (integer)
RAR_HOST_UNIX (integer)
RAR_HOST_BEOS (integer)

例

Example#2 Rar 拡張モジュールの概要

```
<?php

$rar_file = rar_open('example.rar') or die("Rar アーカイブをオープンできません。");

$entries = rar_list($rar_file);

foreach ($entries as $entry) {
    echo 'ファイル名: ' . $entry->getName() . "¥n";
    echo '圧縮時のファイルサイズ: ' . $entry->getPackedSize() . "¥n";
    echo '展開後のファイルサイズ: ' . $entry->getUnpackedSize() . "¥n";

    $entry->extract('/dir/extract/to/');
}

rar_close($rar_file);

?>
```

この例は Rar アーカイブファイルをオープンし、 指定したディレクトリに個々のファイルを展開します。

---

rar_close

(No version information available, might be only in CVS)

rar_close — Rar アーカイブをクローズし、全リソースを開放する

説明

bool **rar_close** (resource $rar_file)

Rar アーカイブをクローズし、割り当てられた全リソースを開放します。

パラメータ

rar_file

rar_open() でオープンした Rar ファイルリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

rar_entry_get

(No version information available, might be only in CVS)

rar_entry_get — Rar アーカイブからエントリオブジェクトを取得する

説明

RarEntry **rar_entry_get** (resource $rar_file , string $entry_name)

Rar アーカイブから、エントリオブジェクトを取得します。

パラメータ

rar_file

rar_open() でオープンした Rar ファイルリソース。

entry_name

Rar アーカイブ内のエントリへのパス。

返り値

rar_get_entry() は、成功した場合にエントリオブジェクト、 エラー時には **FALSE** を返します。

例

Example#1 rar_entry_get() の例

```
<?php
$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりません");
print_r($entry);
?>
```

Rar::extract

(No version information available, might be only in CVS)

Rar::extract — アーカイブのエントリを展開する

説明

Rar
bool **extract** (string $dir [, string $filepath])

Rar::extract() は、エントリのデータを dir に展開します。指定した dir に、エントリの名前と同名の新しいファイルを作成します。

パラメータ

dir

ファイルを展開するディレクトリへのパス。

filepath

dir の代わりに filepath が指定されている場合は、 **Rar::extract()** は指定したファイルに エントリのデータを展開します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 Rar::extract() の例

```
<?php

$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりません");

$entry->extract('/dir/to'); // /dir/to/Dir/file.txt を作成します
$entry->extract(false, '/dir/to/new_name.txt'); // /dir/to/new_name.txt を作成します

?>
```

Example#2 アーカイブ内のすべてのファイルを展開する方法

```
<?php

/* erix こと Erik Jenssen によるサンプルです */

$filename = "foobar.rar";
$filepath = "/home/foo/bar/";

$rar_file = rar_open($filepath.$filename);
$list = rar_list($rar_file);
foreach($list as $file) {
    $entry = rar_entry_get($rar_file, $file);
    $entry->extract("."); // カレントディレクトリに展開します
}
rar_close($rar_file);

?>
```

Rar::getAttr

(No version information available, might be only in CVS)

Rar::getAttr — エントリの属性を取得する

説明

Rar
int **getAttr** (void)

Rar::getAttr() は、アーカイブエントリの属性を返します。

返り値

属性、あるいはエラー時に **FALSE** を返します。

例

Example#1 Rar::getAttr() の例

```
<?php

$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");

$entry = rar_entry_get($rar_file, 'dir/in/the/archive') or die("そのようなエントリは見つかりません");

$host_os = $entry->getHostOs();
$attr = $entry->getAttr();

switch($host_os) {
     case RAR_HOST_MSDOS:
     case RAR_HOST_OS2:
     case RAR_HOST_WIN32:
     case RAR_HOST_MACOS:
         printf("%c%c%c%c%c%c¥n",
                 ($attr & 0x08) ? 'V' : '.',
                 ($attr & 0x10) ? 'D' : '.',
                 ($attr & 0x01) ? 'R' : '.',
                 ($attr & 0x02) ? 'H' : '.',
                 ($attr & 0x04) ? 'S' : '.',
                 ($attr & 0x20) ? 'A' : '.');
          break;
     case RAR_HOST_UNIX:
     case RAR_HOST_BEOS:
          switch ($attr & 0xF000)
          {
               case 0x4000:
                   printf("d");
                    break;
               case 0xA000:
                   printf("l");
                    break;
               default:
                   printf("-");
```

```
                break;
        }
        printf("%c%c%c%c%c%c%c%c¥n",
                ($attr & 0x0100) ? 'r' : '-',
                ($attr & 0x0080) ? 'w' : '-',
                ($attr & 0x0040) ? (($attr & 0x0800) ? 's':'x'):(($attr & 0x0800) ? 'S':'-'),
                ($attr & 0x0020) ? 'r' : '-',
                ($attr & 0x0010) ? 'w' : '-',
                ($attr & 0x0008) ? (($attr & 0x0400) ? 's':'x'):(($attr & 0x0400) ? 'S':'-'),
                ($attr & 0x0004) ? 'r' : '-',
                ($attr & 0x0002) ? 'w' : '-',
                ($attr & 0x0001) ? 'x' : '-');
        break;
}

rar_close($rar_file);

?>
```

参考

- Rar::getHostOs

Rar::getCrc

(No version information available, might be only in CVS)

Rar::getCrc — エントリの CRC を取得する

説明

Rar
int **getCrc** *(void)*

Rar::getCrc() は、アーカイブエントリの CRC を返します。

返り値

アーカイブエントリの CRC、あるいはエラー時に **FALSE** を返します。

Rar::getFileTime

(No version information available, might be only in CVS)

Rar::getFileTime — エントリの最終更新時刻を取得する

説明

Rar
string **getFileTime** *(void)*

エントリの最終更新時刻を取得します。

返り値

エントリの最終更新時刻を YYYY-MM-DD HH:II:SS 形式で表した文字列、あるいはエラー時に **FALSE** を返します。

Rar::getHostOs

(No version information available, might be only in CVS)

Rar::getHostOs — エントリのホスト OS を取得する

説明

Rar
int **getHostOs** *(void)*

Rar::getHostOs() は、アーカイブエントリの ホスト OS のコードを返します。

返り値

ホスト OS のコード、あるいはエラー時に **FALSE** を返します。

例

Example#1 Rar::getHostOs() の例

```
<?php

$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりません");
```

```
switch ($entry->getHostOs()) {
     case RAR_HOST_MSDOS:
          echo "MS-DOS¥n";
          break;
     case RAR_HOST_OS2:
          echo "OS2¥n";
          break;
     case RAR_HOST_WIN32:
          echo "Win32¥n";
          break;
     case RAR_HOST_MACOS:
          echo "MacOS¥n";
          break;
     case RAR_HOST_UNIX:
          echo "Unix/Linux¥n";
          break;
     case RAR_HOST_BEOS:
          echo "BeOS¥n";
          break;
}

?>
```

Rar::getMethod

(No version information available, might be only in CVS)

Rar::getMethod — エントリの圧縮方法を取得する

説明

Rar
int **getMethod** (void)

Rar::getMethod() は、現在のアーカイブエントリを 追加する際に使用したメソッドの番号を返します。

返り値

メソッド番号、あるいはエラー時に **FALSE** を返します。

例

Example#1 Rar::getMethod() の例

```
<?php

$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりません");

echo "メソッド番号: " . $entry->getMethod();

?>
```

Rar::getName

(No version information available, might be only in CVS)

Rar::getName — エントリの名前を取得する

説明

Rar
string **getName** (void)

Rar::getName() は、アーカイブエントリの名前を返します。

返り値

エントリ名を表す文字列、あるいはエラー時に **FALSE** を返します。

例

Example#1 Rar::getName() の例

```
<?php

$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");

$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりません");

echo "エントリ名: " . $entry->getName();

?>
```

Rar::getPackedSize

(No version information available, might be only in CVS)

Rar::getPackedSize — 圧縮後のエントリのサイズを取得する

説明

Rar
int **getPackedSize** (void)

圧縮後のアーカイブエントリのサイズを取得します。

返り値

圧縮後のサイズ、あるいはエラー時に **FALSE** を返します。

例

Example#1 Rar::getPackedSize() の例

```
<?php
$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりません");
echo "圧縮後の " . $entry->getName() . " のサイズ = " . $entry->getPackedSize() . " バイト";
?>
```

Rar::getUnpackedSize

(No version information available, might be only in CVS)

Rar::getUnpackedSize — 展開後のエントリのサイズを取得する

説明

Rar
int **getUnpackedSize** (void)

展開後のアーカイブエントリのサイズを取得します。

返り値

展開後のサイズ、あるいはエラー時に **FALSE** を返します。

返り値

Example#1 Rar::getUnpackedSize() の例

```
<?php
$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりません");
echo "展開後の " . $entry->getName() . " のサイズ = " . $entry->getUnpackedSize() . " バイト";
?>
```

Rar::getVersion

(No version information available, might be only in CVS)

Rar::getVersion — エントリを追加するのに用いたアーカイバのバージョンを取得する

説明

Rar
int **getVersion** (void)

アーカイブエントリを追加するのに用いたアーカイバのバージョンを取得します。

返り値

バージョン、あるいはエラー時に **FALSE** を返します。

例

Example#1 Rar::getVersion() の例

```
<?php
```

```
$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");
$entry = rar_entry_get($rar_file, 'Dir/file.txt') or die("そのようなエントリは見つかりませんでした");
echo "使用した Rar (WinRAR) のバージョン: " . $entry->getVersion();
?>
```

---

rar_list

(No version information available, might be only in CVS)

rar_list — Rar アーカイブのエントリー一覧を取得する

説明

array **rar_list** (resource $rar_file)

Rar アーカイブから、エントリの一覧を取得します。

パラメータ

rar_file

rar_open() でオープンした Rar ファイルリソース。

返り値

rar_list() は成功した場合にエントリの配列、 エラー時には **FALSE** を返します。

例

Example#1 rar_list() の例

```
<?php
$rar_file = rar_open('example.rar') or die("Rar アーカイブのオープンに失敗しました");
$entries_list = rar_list($rar_file);
print_r($entries_list);
?>
```

---

rar_open

(No version information available, might be only in CVS)

rar_open — Rar アーカイブをオープンする

説明

resource **rar_open** (string $filename [, string $password])

指定した Rar アーカイブをオープンし、Rar ファイルリソースを返します。

パラメータ

filename

Rar アーカイブへのパス。

password

必要に応じて、プレーンテキストのパスワード。

返り値

rar_open() は成功した場合に Rar ファイルリソース、 エラー時には **FALSE** を返します。

目次

- Rar::getName — エントリの名前を取得する
- Rar::getPackedSize — 圧縮後のエントリのサイズを取得する
- Rar::getUnpackedSize — 展開後のエントリのサイズを取得する
- Rar::getVersion — エントリを追加するのに用いたアーカイバのバージョンを取得する
- rar_list — Rar アーカイブのエントリー覧を取得する
- rar_open — Rar アーカイブをオープンする

GNU Readline

導入

readline 関数群は、GNU Readline ライブラリへの インターフェースを実装したものです。これらの関数は、コマンドラインの 編集機能を提供します。一例をあげると、Bash において文字を挿入したり コマンド履歴を走査したりする際に、矢印キーを使用することを 可能にしているものです。このライブラリは対話的なものであるため、Web アプリケーションで 使用されることはほとんどないでしょう。しかし、 コマンドライン から PHP を使用するスクリプトを書く際には有用です。

注意: この拡張モジュールは Windows 環境では利用できません。

要件

readline 関数を使用するには、libreadline をインストールすることが 必要です。libreadline は、» http://cnswww.cns.cwru.edu/~chet/readline/rltop.html にある GNU Readline プロジェクトのホームページから入手可能です。このライブラリは、Bash の 作者でもある Chet Ramey により管理されています。

この関数を libeditライブラリで使用することも可能です。これは readline ライブラリの代替品で、非 GPL です。 libeditライブラリは BSD ライセンスで配布され、 » http://www.thrysoee.dk/editline/ から入手可能です。

インストール手順

この関数を使用するには、readline サポートを有効にして CGI 版または CLI 版の PHP をコンパイルする必要があります。PHP の configure に --with-readline[=DIR] を指定する必要が あります。readline の代替品である libedit を使用したい場合、PHP の configure に --with-libedit[=DIR] を指定してください。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

readline_add_history

(PHP 4, PHP 5)

readline_add_history — ヒストリに 1 行追加する

説明

bool **readline_add_history** (string $line)

この関数は、コマンドラインヒストリに 1 行追加します。

パラメータ

line

ヒストリに追加する行。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

readline_callback_handler_install

(PHP 5 >= 5.1.0)

readline_callback_handler_install — readline コールバックインターフェースと端末を初期化し、 プロンプトを表示して結果をすぐに返す

説明

bool **readline_callback_handler_install** (string $prompt , callback $callback)

readline コールバックインターフェースを設定し、プロンプト prompt を表示して入力を受け取ります。 コールバック関数 callback はひとつのパラメータを とり、そこにはユーザの入力内容が格納されます。一度登録したコールバック インターフェースを削除せずもういちどこの関数をコールした場合、もとの インターフェースは自動的に上書きされます。

コールバック機能は stream_select() と組み合わせると 有用です。これは readline() とは異なり、 IO とユーザの入力を交互に取り扱います。

パラメータ

prompt

確認メッセージ。

callback

callback 関数が受け取るパラメータはひとつで、 ユーザから返された入力です。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 Readline コールバックインターフェースの例

```
<?php
function rl_callback($ret)
{
    global $c, $prompting;

    echo "You entered: $ret¥n";
    $c++;

    if ($c > 10) {
        $prompting = false;
        readline_callback_handler_remove();
    } else {
        readline_callback_handler_install("[$c] Enter something: ", 'rl_callback');
    }
}

$c = 1;
$prompting = true;

readline_callback_handler_install("[$c] Enter something: ", 'rl_callback');

while ($prompting) {
    $w = NULL;
    $e = NULL;
    $n = stream_select($r = array(STDIN), $w, $e, null);
    if ($n && in_array(STDIN, $r)) {
        // read a character, will call the callback when a newline is entered
        readline_callback_read_char();
    }
}

echo "Prompting disabled. All done.¥n";
?>
```

参考

- readline_callback_handler_remove()
- readline_callback_read_char()
- stream_select()

---

readline_callback_handler_remove

(PHP 5 >= 5.1.0)

readline_callback_handler_remove — インストールされたハンドラを削除し、端末の設定をもとに戻す

説明

bool **readline_callback_handler_remove** (void)

インストールされたハンドラを削除し、端末の設定をもとに戻します。

返り値

インストールされたコールバックが削除できた場合に TRUE、 削除するハンドラが見つからなかった場合に FALSE を返します。

例

readline コールバックインターフェースの使用方法についての例は readline_callback_handler_install() を参照ください。

参考

- readline_callback_handler_install()

- readline_callback_read_char()

---

readline_callback_read_char

(PHP 5 >= 5.1.0)

readline_callback_read_char — 文字を読み込み、改行を受け取ると readline コールバックインターフェースに通知する

説明

void **readline_callback_read_char** (void)

ユーザが入力した文字を読み込みます。改行を受け取ると、この関数は readline_callback_handler_install() で インストールされた readline コールバックインターフェースに対して 入力待ちを通知します。

返り値

値を返しません。

例

readline コールバックインターフェースの使用方法についての例は readline_callback_handler_install() を参照ください。

参考

- readline_callback_handler_install()
- readline_callback_handler_remove()

---

readline_clear_history

(PHP 4, PHP 5)

readline_clear_history — ヒストリをクリアする

説明

bool **readline_clear_history** (void)

この関数はコマンドラインヒストリ全体をクリアします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

readline_completion_function

(PHP 4, PHP 5)

readline_completion_function — 補完関数を登録する

説明

bool **readline_completion_function** (callback $function)

この関数は補完用の関数を登録します。 これは、Bash を使用している際に、タブキーを 押して得られるのと同様の機能です。

パラメータ

function

コマンドラインの一部を入力とし、 マッチする可能性がある文字列の配列を返す 既存の関数の名前を指定する必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

readline_info

(PHP 4, PHP 5)

readline_info — 種々の readline の内部変数を取得/設定する

説明

mixed **readline_info** ([string $varname [, string $newvalue]])

さまざまな readline の内部変数の取得あるいは設定を行います。

パラメータ

varname

変数の名前。

newvalue

指定した場合は、これがその設定の新しい値となります。

返り値

パラメータを指定しないでコールした場合、この関数は readline が使用する すべての設定の値を配列で返します。要素の添字は次のようになります。 done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, terminal_name

ひとつあるいはふたつのパラメータを指定してコールした場合は、 元の値が返されます。

---

readline_list_history

(PHP 4, PHP 5)

readline_list_history — ヒストリを一覧表示する

説明

array **readline_list_history** (void)

コマンドラインヒストリ全体を取得します。

返り値

コマンドラインヒストリ全体の配列を返します。 各要素にはゼロから始まる整数の添字が付加されています。

---

readline_on_new_line

(PHP 5 >= 5.1.0)

readline_on_new_line — カーソルが新しい行に移動したことを readline に通知する

説明

void **readline_on_new_line** (void)

readline に、カーソルが新しい行に移動したことを通知します。

返り値

値を返しません。

---

readline_read_history

(PHP 4, PHP 5)

readline_read_history — ヒストリを読み込む

説明

bool **readline_read_history** ([string $filename])

この関数は、コマンドヒストリをファイルから読み込みます。

パラメータ

filename

コマンドヒストリを含むファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

readline_redisplay

(PHP 5 >= 5.1.0)

readline_redisplay — 画面を再描画する

説明

void **readline_redisplay** (void)

画面の再描画を readline に依頼します。

返り値

値を返しません。

---

readline_write_history

(PHP 4, PHP 5)

readline_write_history — ヒストリを書きこむ

説明

bool **readline_write_history** ([string $filename])

この関数はコマンドヒストリをファイルに書き込みます。

パラメータ

filename

保存するファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

readline

(PHP 4, PHP 5)

readline — 一行読み込む

説明

string **readline** ([string $prompt])

ユーザからの入力を一行読み込みます。 この行を readline_add_history(). を用いてヒストリに追加する必要があります。

パラメータ

prompt

ユーザに示す確認文字列を指定します。

返り値

ユーザから取得した文字列を一つだけ返します。 返り値の最後の改行は取り除かれます。

例

Example#1 readline() の例

```
<?php
// ユーザから 3 回コマンドを取得
for ($i=0; $i < 3; $i++) {
        $line = readline("Command: ");
        readline_add_history($line);
}

// ヒストリをダンプ
print_r(readline_list_history());

// 変数をダンプ
print_r(readline_info());
?>
```

目次

- readline_clear_history — ヒストリをクリアする
- readline_completion_function — 補完関数を登録する
- readline_info — 種々の readline の内部変数を取得/設定する
- readline_list_history — ヒストリを一覧表示する
- readline_on_new_line — カーソルが新しい行に移動したことを readline に通知する
- readline_read_history — ヒストリを読み込む
- readline_redisplay — 画面を再描画する
- readline_write_history — ヒストリを書きこむ
- readline — 一行読み込む

GNU Recode 関数

導入

このモジュールは、GNU recode ライブラリへのインターフェースを 有しています。GNU Recode ライブラリは、様々にコード化された 文字や表面的なエンコード法を相互に変換します。 正確に変換できない場合、問題のある文字を削除するか近似を行います。 このライブラリは、約 150 種類の文字セットを認識、生成することが可能で、 ほとんど全ての組み合わせで相互の変換を行うことができます。 » RFC 1345 文字セットの ほとんどがサポートされています。

注意: この拡張モジュールは Windows 環境では利用できません。

要件

GNU Recode 3.5 以降をシステムにインストールしておく必要があります。 このパッケージは、» http://directory.fsf.org/All_GNU_Packages/recode.html から ダウンロードすることが可能です。

警告

Recode ライブラリのバージョン 3.6 は、ある状況下での文字列の変換時に 変な文字を付加してしまうことがあります。そのため、Recode v3.5 か、あるいは別の選択肢として iconv あるいは mbstring 拡張モジュールを使用するほうが安全です。

インストール手順

本モジュールで定義された関数を使用するには、--with-recode[=DIR] オプションを指定して PHP インタプリタをコンパイルする必要があります。

警告

拡張モジュール mysql または imap をロードした 後に recode を拡張モジュールとしてロードした場合に、 PHP のクラッシュおよび起動に関する問題が 発生する可能性があります。これらの拡張モジュールの前に recode を ロードすることにより、問題を解決することができます。 これは、imap で使用されている c-client ライブラリおよび recode の両方が 固有の hash_lookup() 関数を有しており、 mysql と recode が固有の hash_insert 関数を有している ことによる技術的な問題です。

警告

IMAP、recode、 YAZ および Cyrus 拡張モジュールは、組み合わせて使用することはできません。これは、これらすべてが同一の内部シンボルを使用しているためです。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

recode_file

(PHP 4, PHP 5)

recode_file — コード変換指令に基づきファイルからファイルにコード変換する

説明

bool **recode_file** (string $request , resource $input , resource $output)

コード変換指令 request に基づきファイルハンドル input が指すファイルをファイルハンドル output が指すファイルにコード変換します。

パラメータ

request

変換指令の型。

input

入力として使用する ローカルファイルハンドルリソース。

output

出力として使用する ローカルファイルハンドルリソース。

返り値

応じることができない場合に **FALSE**、それ以外の場合に **TRUE** を返します。

例

Example#1 基本的な recode_file() の例

```
<?php
$input = fopen('input.txt', 'r');
$output = fopen('output.txt', 'w');
recode_file("us..flat", $input, $output);
?>
```

注意

この関数は現在リモートファイル(URL)を指すファイルハンドルを処理 できません。ファイルハンドルは共にローカルなファイルを指している 必要があります。

参考

- fopen()

---

recode_string

(PHP 4, PHP 5)

recode_string — コード変換指令に基づき文字列のコードを変換する

説明

string **recode_string** (string $request , string $string)

コード変換指令 request に基づき文字列 string のコードを変換します。

パラメータ

request

変換指令の型。

string

変換する文字列。

返り値

変換後の文字列、または変換指令を実行できない場合に **FALSE** を返します。

例

Example#1 基本的な recode_string() の例

```
<?php
echo recode_string("us..flat", "The following character has a diacritical mark: &aacute;");
?>
```

注意

簡単なコード変換指令は、"lat1..iso646-de" のようになります。

参考

- コード変換指令に関する詳細な手順に関しては、インストールされている GNU Recode のドキュメントも参照ください。

---

recode

(PHP 4, PHP 5)

recode — recode_string() のエイリアス

説明

この関数は次の関数のエイリアスです。 recode_string().

目次

- recode_file — コード変換指令に基づきファイルからファイルにコード変換する
- recode_string — コード変換指令に基づき文字列のコードを変換する
- recode — recode_string のエイリアス

RPM ヘッダ読み込み関数

導入

このモジュールは、» RedHat Package Manager (» RPM) ファイルのヘッダに保存されているメタ情報を読み込む機能を提供します。

要件

RPMReader 拡張モジュールは、PHP 5 以降で動作します。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/rpmreader.

リソース型

RPMReader モジュールは 1 つのリソース型を使用します。 それは、作業対象の RPM ファイrうを指すファイルポインタです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

以下の定数の一覧は、rpm_get_tag() を使用して 情報を取得する際に使用されます。これらの定数は RPM ファイルの ヘッダセクションから取得できるタグ番号を表しています。個々の タグ番号が指すデータの内容について、以下で説明します。

RPMREADER_MINIMUM (integer)
使用可能な RPM タグ番号の最小値。
RPMREADER_NAME (integer)
RPM パッケージの名前。
RPMREADER_VERSION (integer)
RPM パッケージのバージョン。
RPMREADER_RELEASE (integer)
RPM パッケージのリリース番号。
RPMREADER_EPOCH (integer)
RPMREADER_SERIAL (integer)
RPMREADER_SUMMARY (integer)
RPM パッケージの概要テキスト。
RPMREADER_DESCRIPTION (integer)
RPM パッケージについての詳細な説明テキスト。
RPMREADER_BUILDTIME (integer)
RPM パッケージがビルドされた日付と時刻。
RPMREADER_BUILDHOST (integer)
RPM パッケージがビルドされたホストの名前。
RPMREADER_INSTALLTIME (integer)
RPMREADER_SIZE (integer)
RPM パッケージのサイズ。
RPMREADER_DISTRIBUTION (integer)
RPMREADER_VENDOR (integer)
RPMREADER_GIF (integer)
RPMREADER_XPM (integer)
RPMREADER_LICENSE (integer)
RPMREADER_COPYRIGHT (integer)
RPMREADER_PACKAGER (integer)
RPMREADER_GROUP (integer)
RPMREADER_SOURCE (integer)
RPMREADER_PATCH (integer)
RPMREADER_URL (integer)
RPMREADER_OS (integer)
RPMREADER_ARCH (integer)
RPMREADER_PREIN (integer)
RPMREADER_POSTIN (integer)
RPMREADER_PREUN (integer)
RPMREADER_POSTUN (integer)
RPMREADER_OLDFILENAMES (integer)
RPM パッケージ内のファイル一覧（古い形式）。現在では、 RPM が "CompressedFileNames" と呼んでいる方式のもとで 以下の 3 つの タグ (RPMREADER_BASENAMES, RPMREADER_DIRINDEXES, RPMREADER_DIRNAMES) を組み合わせて使用するのが正しい方法です。古い形式の RPM ファイルで "CompressedFileNames" を使用していないものではこのタグがまだ使用 されており、過去との互換性のためにこの定数が残されています。
RPMREADER_FILESIZES (integer)
RPMREADER_FILESTATES (integer)
RPMREADER_FILEMODES (integer)
RPMREADER_FILERDEVS (integer)
RPMREADER_FILEMTIMES (integer)
RPMREADER_FILEMD5S (integer)
RPMREADER_FILELINKTOS (integer)
RPMREADER_FILEFLAGS (integer)
RPMREADER_FILEUSERNAME (integer)
RPMREADER_FILEGROUPNAME (integer)
RPMREADER_ICON (integer)
RPMREADER_SOURCERPM (integer)
RPMREADER_FILEVERIFYFLAGS (integer)

RPMREADER_ARCHIVESIZE (integer)
RPMREADER_PROVIDENAME (integer)
RPMREADER_PROVIDES (integer)
RPMREADER_REQUIREFLAGS (integer)
RPMREADER_REQUIRENAME (integer)
RPMREADER_REQUIREVERSION (integer)
RPMREADER_CONFLICTFLAGS (integer)
RPMREADER_CONFLICTNAME (integer)
RPMREADER_CONFLICTVERSION (integer)
RPMREADER_EXCLUDEARCH (integer)
RPMREADER_EXCLUDEOS (integer)
RPMREADER_EXCLUSIVEARCH (integer)
RPMREADER_EXCLUSIVEOS (integer)
RPMREADER_RPMVERSION (integer)
RPMREADER_TRIGGERSCRIPTS (integer)
RPMREADER_TRIGGERNAME (integer)
RPMREADER_TRIGGERVERSION (integer)
RPMREADER_TRIGGERFLAGS (integer)
RPMREADER_TRIGGERINDEX (integer)
RPMREADER_VERIFYSCRIPT (integer)
RPMREADER_CHANGELOGTIME (integer)
Changelog エントリの更新日の一覧。
RPMREADER_CHANGELOGNAME (integer)
Changelog エントリの名前。
RPMREADER_CHANGELOGTEXT (integer)
Changelog エントリのテキストの一覧。
RPMREADER_PREINPROG (integer)
RPMREADER_POSTINPROG (integer)
RPMREADER_PREUNPROG (integer)
RPMREADER_POSTUNPROG (integer)
RPMREADER_BUILDARCHS (integer)
RPMREADER_OBSOLETENAME (integer)
RPMREADER_OBSOLETES (integer)
RPMREADER_VERIFYSCRIPTPROG (integer)
RPMREADER_TRIGGERSCRIPTPROG (integer)
RPMREADER_COOKIE (integer)
RPMREADER_FILEDEVICES (integer)
RPMREADER_FILEINODES (integer)
RPMREADER_FILELANGS (integer)
RPMREADER_PREFIXES (integer)
RPMREADER_INSTPREFIXES (integer)
RPMREADER_PROVIDEFLAGS (integer)
RPMREADER_PROVIDEVERSION (integer)
RPMREADER_OBSOLETEFLAGS (integer)
RPMREADER_OBSOLETEVERSION (integer)
RPMREADER_DIRINDEXES (integer)
RPM パッケージ内でのディレクトリ名とファイルとの 関連インデックスの一覧。 このタグは、RPM の新しい "CompressedFileNames" 方式で 格納されたファイル名を再構築するために RPMREADER_BASENAMES および RPMREADER_DIRNAMES とともに使用されます。
RPMREADER_BASENAMES (integer)
パス情報を含まない、RPM パッケージ内のファイル名の一覧。 このタグは、RPM の新しい "CompressedFileNames" 方式で 格納されたファイル名を再構築するために RPMREADER_DIRINDEXES および RPMREADER_DIRNAMES とともに使用されます。
RPMREADER_DIRNAMES (integer)
RPM パッケージ内のファイルが使用しているディレクトリ名の一覧。 このタグは、RPM の新しい "CompressedFileNames" 方式で 格納されたファイル名を再構築するために RPMREADER_BASENAMES および RPMREADER_DIRINDEXES とともに使用されます。
RPMREADER_OPTFLAGS (integer)
RPMREADER_DISTURL (integer)
RPMREADER_PAYLOADFORMAT (integer)
RPMREADER_PAYLOADCOMPRESSOR (integer)
RPMREADER_PAYLOADFLAGS (integer)
RPMREADER_INSTALLCOLOR (integer)
RPMREADER_INSTALLTID (integer)
RPMREADER_REMOVETID (integer)
RPMREADER_RHNPLATFORM (integer)
RPMREADER_PLATFORM (integer)
RPMREADER_PATCHESNAME (integer)
RPMREADER_PATCHESFLAGS (integer)
RPMREADER_PATCHESVERSION (integer)
RPMREADER_CACHECTIME (integer)
RPMREADER_CACHEPKGPATH (integer)
RPMREADER_CACHEPKGSIZE (integer)
RPMREADER_CACHEPKGMTIME (integer)
RPMREADER_FILECOLORS (integer)
RPMREADER_FILECLASS (integer)
RPMREADER_CLASSDICT (integer)
RPMREADER_FILEDEPENDSX (integer)
RPMREADER_FILEDEPENDSN (integer)
RPMREADER_DEPENDSDICT (integer)
RPMREADER_SOURCEPKGID (integer)
RPMREADER_FILECONTEXTS (integer)
RPMREADER_FSCONTEXTS (integer)
RPMREADER_RECONTEXTS (integer)
RPMREADER_POLICIES (integer)
RPMREADER_MAXIMUM (integer)
使用可能な RPM タグ番号の最大値。

例

この例では RPM ファイルをオープンし、ファイルから名前・バージョン・ リリース番号を読み込み、結果を表示してファイルを閉じます。

Example#1 基本的な RPMReader の例

```php
<?php

$filename = "/path/to/file.rpm";

// ファイルをオープンします
$rpmr = rpm_open($filename);

// "Name" タグを取得します
$name = rpm_get_tag($rpmr, RPMREADER_NAME);
```

```
// "Version" タグを取得します
$ver = rpm_get_tag($rpmr, RPMREADER_VERSION);

// "Release" タグを取得します
$rel = rpm_get_tag($rpmr, RPMREADER_RELEASE);

echo "$name-$ver-$rel<br>¥n";

// ファイルを閉じます
rpm_close($rpmr);

?>
```

---

rpm_close

(PECL rpmreader:0.1-0.3)

rpm_close — RPM ファイルを閉じる

説明

bool **rpm_close** (resource $rpmr)

rpm_close() は RPM ファイルポインタを閉じます。

パラメータ

rpmr

rpm_open() でオープンしたファイルポインタリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 rpm_close() の例

```
<?php

$file = "/path/to/file.rpm";
$rpmr = rpm_open($file);

rpm_close($rpmr);

?>
```

参考

- rpm_open()

---

rpm_get_tag

(PECL rpmreader:0.1-0.3)

rpm_get_tag — RPM ファイルからヘッダタグを取得する

説明

mixed **rpm_get_tag** (resource $rpmr , int $tagnum)

rpm_get_tag() は、RPM ファイルのヘッダから 指定したタグを取得してそれを返します。

パラメータ

rpmr

rpm_open() でオープンした ファイルポインタリソース。

tagnum

RPM ヘッダから取得するタグの番号。この値は、 このモジュールで定義されている定数の一覧を使用して指定可能です。

返り値

関数に渡した tagnum の値により、 返り値の型はいろいろな形式になりえます。

例

Example#1 rpm_get_tag() の例

```
<?php

$file = "/path/to/file.rpm";
$rpmr = rpm_open($file);
```

```
$name = rpm_get_tag($rpmr, RPMREADER_NAME);
echo "$name<br>¥n";

rpm_close($rpmr);

?>
```

参考

- rpm_open()
- rpm_close()

---

rpm_is_valid

(PECL rpmreader:0.1-0.3)

rpm_is_valid — filename が RPM ファイルであるかどうかを確かめる

説明

bool **rpm_is_valid** (string $filename)

rpm_is_valid() は、ファイルが RPM ファイルとして有効な形式であるかどうかを調べます。これは rpm_open() とは違って単にファイルを調べるだけであり、 他の関数で使用するためのファイルポインタは返しません。

パラメータ

filename

調べたい RPM ファイル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 rpm_is_valid() の例

```
<?php

$file = "/path/to/file.rpm";

if (rpm_is_valid($file)) {
    echo "ファイルは RPM 形式です。<br>\n";
}
else {
    echo "ファイルは RPM 形式ではありません。<br>\n";
}

?>
```

---

rpm_open

(PECL rpmreader:0.1-0.3)

rpm_open — RPM ファイルをオープンする

説明

resource **rpm_open** (string $filename)

rpm_open() は、RPM ファイルをオープンして それが正しい RPM 形式であるかどうかを調べます。

パラメータ

filename

オープンしたい RPM ファイルの名前。

返り値

オープンに成功すると、**rpm_open()** は オープンしたファイルのファイルポインタリソースを返します。 エラー時にはこの関数は **FALSE** を返します。

例

Example#1 rpm_open() の例

```
<?php

$file = "/path/to/file.rpm";
$rpmr = rpm_open($file);
```

```
rpm_close($rpmr);

?>
```

参考

- rpm_close()

---

rpm_version

(PECL rpmreader:0.3)

rpm_version — rpmreader 拡張モジュールの現在のバージョンを表す文字列を返す

説明

string **rpm_version** (void)

rpm_version() は、rpmreader 拡張モジュールの 現在のバージョンを返します。

返り値

rpm_version() は、現在 PHP に読み込まれている rpmreader のバージョンを表す文字列を返します。

例

Example#1 rpm_version() の例

```
<?php

$rpmr_ver = rpm_version();

echo "$rpmr_ver<br />¥n";

?>
```

目次

---

runkit 関数

導入

runkit 拡張モジュールは、定数・ユーザ定義関数 およびユーザ定義クラスを変更する機能を提供します。 また、独自のスーパーグローバル変数を定義したり、 サンドボックスを利用した組み込みインタプリタの作成も可能です。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/runkit.

このパッケージには、 » classkit パッケージに 取って代わるための機能も含まれています。 ./configure に --enable-runkit=classkit オプションを つけてコンパイルすると、classkit と互換性のある関数や定数を提供します。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/runkit.

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

要件

定数・関数・クラス・メソッドの変更機能 は、PHP 4 と PHP 5 の全てのリリースで動作します。特別な動作条件はありません。

独自のスーパーグローバル変数 は PHP 4.2.0 以降で使用可能です。

サンドボックスのサポート は、PHP 5.1.0 以降か あるいは TSRM パッチを適用した PHP 5.0.0 が必要です。 どのバージョンの PHP を使用しているかにかかわらず、コンパイル時に --enable-maintainer-zts オプションが必要です。 詳しい情報は、runkit パッケージに含まれる README ファイルを参照ください。

実行時設定

php.ini の設定により動作が変化します。

Runkit 設定オプション

名前	デフォルト	変更の可否	変更履歴
runkit.superglobal	""	PHP_INI_PERDIR	
runkit.internal_override	"0"	PHP_INI_SYSTEM	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

runkit.superglobal string
スーパーグローバルとして扱いたい変数名の、カンマ区切りのリストです。 これはシステム全体で利用する php.ini ファイルで指定すべきですが、SAPI の設定によってはディレクトリローカルで指定しても動作するかもしれません。

Example#1 php.ini で runkit.superglobal=_FOO,_BAR と指定した場合のスーパーグローバル

```
<?php
function show_values() {
   echo "Foo is $_FOO¥n";
   echo "Bar is $_BAR¥n";
   echo "Baz is $_BAZ¥n";
}

$_FOO = 'foo';
$_BAR = 'bar';
$_BAZ = 'baz';

/* foo と bar が表示されるが、baz は表示されない */
show_values();
?>
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

RUNKIT_IMPORT_FUNCTIONS (integer)
runkit_import() のフラグで、 指定したファイルから通常の関数が読み込まれることを示します。
RUNKIT_IMPORT_CLASS_METHODS (integer)
runkit_import() のフラグで、 指定したファイルからクラスメソッドが読み込まれることを示します。
RUNKIT_IMPORT_CLASS_CONSTS (integer)
runkit_import() のフラグで、 指定したファイルからクラス定数が読み込まれることを示します。 このフラグは PHP のバージョン 5.1.0 以降でしか意味を持たないことに 注意しましょう。
RUNKIT_IMPORT_CLASS_PROPS (integer)
runkit_import() のフラグで、 指定したファイルからクラスのプロパティが読み込まれることを示します。
RUNKIT_IMPORT_CLASSES (integer)
runkit_import() のフラグで、 **RUNKIT_IMPORT_CLASS_*** の論理和（ビット OR）を表します。
RUNKIT_IMPORT_OVERRIDE (integer)
runkit_import() のフラグで、 すでに存在する関数・メソッド・定数 あるいはプロパティが指定された場合に新しい定義で置き換えることを 示します。このフラグが設定されていない場合、すでに定義されている 内容については新しい定義は無視されます。
RUNKIT_ACC_PUBLIC (integer)
runkit_method_add() の PHP5 限定のフラグです。
RUNKIT_ACC_PROTECTED (integer)
runkit_method_add() の PHP5 限定のフラグです。
RUNKIT_ACC_PRIVATE (integer)
runkit_method_add() の PHP5 限定のフラグです。
CLASSKIT_ACC_PUBLIC (integer)
runkit_method_add() の PHP5 限定のフラグです。 classkit 互換モードが有効になっている場合にのみ定義されます。
CLASSKIT_ACC_PROTECTED (integer)
runkit_method_add() の PHP5 限定のフラグです。 classkit 互換モードが有効になっている場合にのみ定義されます。
CLASSKIT_ACC_PRIVATE (integer)
runkit_method_add() の PHP5 限定のフラグです。 classkit 互換モードが有効になっている場合にのみ定義されます。
CLASSKIT_AGGREGATE_OVERRIDE (integer)
classkit_import() の PHP5 限定のフラグです。 classkit 互換モードが有効になっている場合にのみ定義されます。
RUNKIT_VERSION (string)
runkit パッケージのバージョンを定義します。
CLASSKIT_VERSION (string)
runkit パッケージのバージョンを定義します。 classkit 互換モードが有効になっている場合にのみ定義されます。

Runkit_Sandbox

(No version information available, might be only in CVS)

Runkit_Sandbox — Runkit Sandbox クラス -- PHP バーチャルマシン

説明

Runkit_Sandbox クラスをインスタンス化すると、 独自のスコープとプログラムスタックを持つ新しいスレッドが生成されます。 コンストラクタに渡すオプションを設定することで、この環境では インタプリタの機能を制限することが可能です。そのため、ユーザから渡された コードを実行する場合などに、より安全な環境を提供可能です。

> **注意**: サンドボックスのサポート (runkit_lint(), runkit_lint_file(), および Runkit_Sandbox クラスで使用)は、PHP 5.1 以降または特別なパッチを適用した PHP 5.0 でのみ利用可能であり、スレッドセーフを有効にしておく必要があります。詳細については、runkit パッケージに含まれる README ファイルを参照してください。

コンストラクタ

void **Runkit_Sandbox::__construct** ([array $options])

options は連想配列で、その中に 以下の ini オプションの組み合わせを格納します。

safe_mode

Runkit_Sandbox クラスをインスタンス化した 外部スクリプトが safe_mode = off と設定されている 場合、サンドボックス内の safe_mode を on にすることが可能です。 外部スクリプトで safe_mode が有効になっている 場合に、サンドボックス内でそれを無効にすることはできません。

safe_mode_gid

Runkit_Sandbox クラスをインスタンス化した 外部スクリプトが safe_mode_gid = on と設定されている 場合、サンドボックス内の safe_mode_gid を off にすることが可能です。 外部スクリプトで safe_mode_gid が無効になっている 場合に、サンドボックス内でそれを有効にすることはできません。

safe_mode_include_dir

Runkit_Sandbox クラスをインスタンス化した 外部スクリプトが safe_mode_include_dir を 組み込んで configure されていた場合、サンドボックス内の safe_mode_include_dir を定義されているディレクトリ以下に指定することが 可能です。この機能を無効にすることを示すため、safe_mode_include_dir をクリアすることも可能です。 外部スクリプトで safe_mode_include_dir が空白になっているが safe_mode は有効でない場合、任意の safe_mode_include_dir を 指定して safe_mode を on にすることが可能です。

open_basedir

open_basedir は、カレントの open_basedir 以下の任意のパスを指定可能です。 グローバルスコープで open_basedir が指定されていない場合、それはルートディレクトリと判断され、 どの場所でも指定可能となります。

allow_url_fopen

safe_mode と同様、より厳しくする方向にのみ 指定可能です。この場合は **TRUE** と指定されている場合に **FALSE** を 指定することが可能となります。

disable_functions

サンドボックス内のインタプリタで無効とする関数を、カンマ区切りの リストで指定します。現在すでに無効になっている関数名を含める必要は ありません。それらはリストに載っているか否かにかかわらず無効となります。

disable_classes

サンドボックス内のインタプリタで無効とするクラスを、カンマ区切りの リストで指定します。現在すでに無効になっているクラス名を含める必要はありません。それらはリストに載っているか否かにかかわらず無効となります。

runkit.superglobal

サンドボックス内のインタプリタでスーパーグローバルとして扱う変数を、 カンマ区切りのリストで指定します。これらの変数は、既に内部で 定義されている変数やグローバルの runkit.superglobal 設定に 追加して使用されます。

runkit.internal_override

Ini オプション runkit.internal_override は、 サンドボックス内では無効になる（そして、再度有効にはならない） かもしれません。

Example#1 機能を制限したサンドボックスのインスタンス化

```php
<?php
$options = array(
  'safe_mode'=>true,
  'open_basedir'=>'/var/www/users/jdoe/',
  'allow_url_fopen'=>'false',
  'disable_functions'=>'exec,shell_exec,passthru,system',
  'disable_classes'=>'myAppClass');
$sandbox = new Runkit_Sandbox($options);
/* 制限されていない ini 項目は、普通に設定できる */
$sandbox->ini_set('html_errors',true);
?>
```

変数へのアクセス

サンドボックス環境内のすべてのグローバル変数は、サンドボックス オブジェクトのプロパティとしてアクセス可能です。しかし、 オブジェクト変数やリソース変数はインタプリタ越しに 利用することができないことに注意しましょう。これは、これらの 2 つのスレッドのメモリ管理が仮想マシン上で行われていることが原因です。 さらに、配列はすべてディープコピーされ、参照渡しのデータは 失われます。つまり、参照渡しのデータをインタプリタ越しに使用することは できないということです。

Example#2 サンドボックス内部の変数の扱い

```php
<?php
$sandbox = new Runkit_Sandbox();

$sandbox->foo = 'bar';
$sandbox->eval('echo "$foo¥n"; $bar = $foo . "baz";');
echo "{$sandbox->bar}¥n";
if (isset($sandbox->foo)) unset($sandbox->foo);
$sandbox->eval('var_dump(isset($foo));');
?>
```

上の例の出力は以下となります。

```
bar
barbaz
bool(false)
```

PHP 関数のコール

サンドボックス内で定義されている関数は、すべてサンドボックス オブジェクトのメソッドとしてコールできます。これには、擬似関数として 扱われる以下のような言語構造も含みます。eval()、 include()、include_once()、 require()、require_once()、 echo()、print()、 die() および exit()。

Example#3 サンドボックス内の関数のコール

```php
<?php
```

```
$sandbox = new Runkit_Sandbox();

echo $sandbox->str_replace('a','f','abc');
?>
```

上の例の出力は以下となります。

```
fbc
```

サンドボックス内の関数への引数は、外部の PHP インスタンスから 渡されます。もしサンドボックスのスコープから引数を渡したい場合は、 上で示したようにサンドボックスのプロパティとしてそれにアクセスする ようにしてください。

Example#4 サンドボックス内の関数に引数を渡す

```
<?php
$sandbox = new Runkit_Sandbox();

$foo = 'bar';
$sandbox->foo = 'baz';
echo $sandbox->str_replace('a',$foo,'a');
echo $sandbox->str_replace('a',$sandbox->foo,'a');
?>
```

上の例の出力は以下となります。

```
bar
baz
```

サンドボックス設定の変更

runkit バージョン 0.5 以降では、配列へのアクセスと同じ構文で、 実行時にサンドボックスの一部の設定を変更することが可能です。 active は読み込み専用で、現在の状態に ついての情報を提供します。output_handler は通常の配列オフセットと同様に読み書きが可能です。 将来的には書き込み専用の設定項目も現れるかもしれませんが、 今のところはそのような項目はありません。

サンドボックス設定 / 状態のインジケータ

設定	型	目的	デフォルト
active	Boolean (Read Only)	サンドボックスが使用可能な状態である場合に **TRUE** 、 die() や exit() のコールもしくは致命的なエラーなどで リクエストから抜けた状態である場合に **FALSE** 。	**TRUE** (Initial)
output_handler	Callback	有効なコールバックを指定すると、サンドボックス インスタンスの生成するすべての出力に対してその 関数が適用されます。 サンドボックス出力ハンドラは、システム全体の 出力ハンドラと同じ呼び出し規約に従います。	なし
parent_access	Boolean	サンドボックスが Runkit_Sandbox_Parent クラスのインスタンスを使用するかどうか。使用するには、その他の Runkit_Sandbox_Parent 関連の設定を有効にする必要があります。	**FALSE**
parent_read	Boolean	サンドボックスが親コンテキストの変数を読み込むかどうか。	**FALSE**
parent_write	Boolean	サンドボックスが親コンテキストの変数を変更するかどうか。	**FALSE**
parent_eval	Boolean	サンドボックスが親コンテキストの任意のコードを評価する（evaluate） かどうか。*危険です。*	**FALSE**
parent_include	Boolean	サンドボックスが親コンテキストの php ファイルを include するかどうか。 *危険です。*	**FALSE**
parent_echo	Boolean	サンドボックスが親コンテキストのデータを表示する際に、それ自身の 出力ハンドラを回避するかどうか。	**FALSE**
parent_call	Boolean	サンドボックスが親コンテキストの関数をコールするかどうか。	**FALSE**
parent_die	Boolean	サンドボックスが親コンテキスト（そして自分自身）を終了させるかどうか。	**FALSE**
parent_scope	Integer	親のプロパティに対してどのようなスコープでアクセスするか。 0 == Global scope, 1 == Calling scope, 2 == Scope preceeding calling scope, 3 == The scope before that, etc..., etc...	*0* (Global)
parent_scope	String	*parent_scope* に文字列値が設定されている場合は、 グローバルスコープの配列変数名を表します。アクセス時にその名前の 変数が存在しない場合は、空の配列が作成されます。変数が存在するが 配列ではない場合は、その変数への参照を含むダミー配列が作成されます。	

Runkit_Sandbox_Parent

(No version information available, might be only in CVS)

Runkit_Sandbox_Parent — Runkit 反サンドボックス（Anti-Sandbox）クラス

説明

void **Runkit_Sandbox_Parent::__construct** (void)

Runkit_Sandbox クラスが作成した サンドボックス環境から Runkit_Sandbox_Parent クラスのインスタンスを作成し、サンドボックスからその親

インスタンスに アクセスするための（管理された）手段を提供します。

> **注意**: サンドボックスのサポート (runkit_lint(), runkit_lint_file(), および Runkit_Sandbox クラスで使用)は、PHP 5.1 以降または特別なパッチを適用した PHP 5.0 でのみ利用可能であり、スレッドセーフを有効にしておく必要があります。詳細については、runkit パッケージに含まれる README ファイルを参照してください。

Runkit_Sandbox_Parent の機能を有効にするには、 親のコンテキストから各サンドボックス単位で parent_access フラグを有効にする必要があります。

Example#1 サンドボックスから変数を使用する

```
<?php
$sandbox = new Runkit_Sandbox();
$sandbox['parent_access'] = true;
?>
```

親の変数にアクセスする

サンドボックス内の変数へのアクセスと同様、親の変数の読み込みや 親の変数への書き込みが可能です。これは Runkit_Sandbox_Parent クラスのプロパティとして 扱われます。親の変数の読み込みは、parent_read 設定を有効にすることで可能となります（それに加えて、基本となる parent_access の設定も必要です）。 一方、書き込みについては parent_write の設定により 可能となります。

サンドボックス内の変数へのアクセスとは異なり、変数のスコープが グローバルに限定されることはありません。parent_scope を適切な整数値に設定することで、アクティブなコールスタック内で 他のスコープが使用可能です。0（デフォルト）は、グローバルスコープで 変数に直接アクセスします。1 は、サンドボックスのコードが実行された ブロック内のスコープを設定します。それより大きい値を指定した場合、 サンドボックスのコードを実行した関数の呼び出し元を順にたどり、 そのスコープの変数にアクセスしようと試みます。

Example#2 親の変数にアクセスする

```
<?php
$php = new Runkit_Sandbox();
$php['parent_access'] = true;
$php['parent_read'] = true;

$test = "Global";

$php->eval('$PARENT = new Runkit_Sandbox_Parent;');

$php['parent_scope'] = 0;
one();

$php['parent_scope'] = 1;
one();

$php['parent_scope'] = 2;
one();

$php['parent_scope'] = 3;
one();

$php['parent_scope'] = 4;
one();

$php['parent_scope'] = 5;
one();

function one() {
    $test = "one()";
    two();
}

function two() {
    $test = "two()";
    three();
}

function three() {
    $test = "three()";
    $GLOBALS['php']->eval('var_dump($PARENT->test);');
}
?>
```

上の例の出力は以下となります。

```
string(6) "Global"
string(7) "three()"
string(5) "two()"
string(5) "one()"
string(6) "Global"
string(6) "Global"
```

親の関数をコールする

適切な設定を有効にすることで、サンドボックスはその親の関数に アクセスすることが可能となります。 parent_call を有効にすると、サンドボックスから 親スコープの全ての関数をコールすることが可能となります。 言語構造については、以下のように個々の設定項目で管理されます。
print() および echo() は parent_echo を有効にすると使用可能です。 die() および exit() は parent_die を有効にすると使用可能です。
eval() は parent_eval を 有効にすると使用可能です。また include()、 include_once()、 require() および require_once() は parent_include を有効にすると使用可能です。

---

runkit_class_adopt

(PECL runkit:0.7-0.9)

runkit_class_adopt — ある基底クラスを、他のクラスを継承させたクラスに変換する。親クラスの適切なメソッドを追加する

説明

bool **runkit_class_adopt** (string $classname , string $parentname)

パラメータ

classname

変換の対象となるクラス。

parentname

継承させる親クラス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 runkit_class_adopt() の例

```
<?php
class myParent {
   function parentFunc() {
     echo "Parent Function Output¥n";
   }
}

class myChild {
}

runkit_class_adopt('myChild','myParent');
myChild::parentFunc();
?>
```

上の例の出力は以下となります。

```
Parent Function Output
```

参考

- runkit_class_emancipate()

---

runkit_class_emancipate

(PECL runkit:0.7-0.9)

runkit_class_emancipate — 他のクラスを継承しているクラスから継承関係を解消し、 親クラスから継承しているメソッドを取り除く

説明

bool **runkit_class_emancipate** (string $classname)

パラメータ

classname

継承関係を解消するクラス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 runkit_class_emancipate() の例

```
<?php
class myParent {
   function parentFunc () {
     echo "Parent Function Output¥n";
   }
}
class myChild extends myParent {
}

myChild::parentFunc();
runkit_class_emancipate('myChild');
myChild::parentFunc();
?>
```

上の例の出力は以下となります。

```
Parent Function Output
Fatal error: Call to undefined function:  parentFunc() in example.php on line 12
```

参考

- runkit_class_adopt()

---

runkit_constant_add

(PECL runkit:0.7-0.9)

runkit_constant_add — define() と同じだが、クラス定数も指定可能

説明

bool **runkit_constant_add** (string $constname , mixed $value)

パラメータ

constname

定義する定数名。グローバル定数を表す文字列、あるいは classname::constname 形式でクラス定数を示す。

value

新しい定数に定義する値。 NULL, Bool, Long, Double, String, あるいは Resource のいずれかの型の値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- define()
- runkit_constant_redefine()
- runkit_constant_remove()

---

runkit_constant_redefine

(PECL runkit:0.7-0.9)

runkit_constant_redefine — 定義済みの定数を再定義する

説明

bool **runkit_constant_redefine** (string $constname , mixed $newvalue)

パラメータ

constname

再定義する定数名。グローバル定数を表す文字列、あるいは classname::constname 形式でクラス定数を示す。

newvalue

定数に新しく設定する値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- runkit_constant_add()
- runkit_constant_remove()

---

runkit_constant_remove

(PECL runkit:0.7-0.9)

runkit_constant_remove — 定義済みの定数を削除する

説明

bool **runkit_constant_remove** (string $constname)

パラメータ

constname

削除する定数名。グローバル定数を表す文字列、あるいは *classname::constname* 形式でクラス定数を示す。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- define()
- runkit_constant_add()
- runkit_constant_redefine()

---

runkit_function_add

(PECL runkit:0.7-0.9)

runkit_function_add — 新しい関数を追加する。create_function() と同じ

説明

bool **runkit_function_add** (string $funcname , string $arglist , string $code)

パラメータ

funcname

作成する関数の名前。

arglist

カンマ区切りの引数のリスト。

code

関数のコード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 runkit_function_add() の例

```
<?php
runkit_function_add('testme','$a,$b','echo "The value of a is $a¥n"; echo "The value of b is $b¥n";');
testme(1,2);
?>
```

上の例の出力は以下となります。

```
The value of a is 1
The value of b is 2
```

参考

- create_function()
- runkit_function_redefine()
- runkit_function_copy()
- runkit_function_rename()
- runkit_function_remove()
- runkit_method_add()

---

runkit_function_copy

(PECL runkit:0.7-0.9)

runkit_function_copy — 関数を別の名前でコピーする

説明

bool **runkit_function_copy** (string $funcname , string $targetname)

パラメータ

funcname

既存の関数の名前。

targetname

コピー後の新しい関数の名前。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

例

Example#1 runkit_function_copy() の例

```
<?php
function original() {
  echo "In a function¥n";
}
runkit_function_copy('original','duplicate');
original();
duplicate();
?>
```

上の例の出力は以下となります。

```
In a function
In a function
```

参考

- *runkit_function_add()*
- *runkit_function_redefine()*
- *runkit_function_rename()*
- *runkit_function_remove()*

---

runkit_function_redefine

(PECL runkit:0.7-0.9)

runkit_function_redefine — 関数の定義を新しい実装で置き換える

説明

bool ***runkit_function_redefine*** *(string $funcname , string $arglist , string $code)*

> ***注意****: デフォルトでは、削除・リネーム・変更が可能なのはユーザ定義関数だけです。組み込み関数をオーバーライドするには、システムの php.ini ファイルで runkit.internal_override を有効にする必要があります。*

パラメータ

funcname

再定義する関数の名前。

arglist

再定義後の関数が受け取る引数のリスト。

code

新しい実装コード。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

例

Example#1 runkit_function_redefine() の例

```
<?php
function testme() {
  echo "Original Testme Implementation¥n";
}
testme();
runkit_function_redefine('testme','','echo "New Testme Implementation¥n";');
testme();
?>
```

上の例の出力は以下となります。

```
Original Testme Implementation
New Testme Implementation
```

参考

- runkit_function_add()
- runkit_function_copy()
- runkit_function_rename()
- runkit_function_remove()

---

runkit_function_remove

(PECL runkit:0.7-0.9)

runkit_function_remove — 関数の定義を削除する

説明

bool **runkit_function_remove** (string $funcname)

> **注意**: デフォルトでは、削除・リネーム・変更が可能なのはユーザ定義関数だけです。組み込み関数をオーバーライドするには、システムの php.ini ファイルで runkit.internal_override を有効にする必要があります。

パラメータ

funcname

削除する関数の名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- runkit_function_add()
- runkit_function_copy()
- runkit_function_redefine()
- runkit_function_rename()

---

runkit_function_rename

(PECL runkit:0.7-0.9)

runkit_function_rename — 関数名を変更する

説明

bool **runkit_function_rename** (string $funcname , string $newname)

> **注意**: デフォルトでは、削除・リネーム・変更が可能なのはユーザ定義関数だけです。組み込み関数をオーバーライドするには、システムの php.ini ファイルで runkit.internal_override を有効にする必要があります。

パラメータ

funcname

現在の関数名。

newname

新しい関数名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- runkit_function_add()
- runkit_function_copy()
- runkit_function_redefine()
- runkit_function_remove()

---

runkit_import

(PECL runkit:0.7-0.9)

runkit_import — ファイルから関数やクラスの定義を読み込み、必要に応じて書き換える

説明

bool **runkit_import** (string $filename [, int $flags])

include() と似ていますが、関数やクラスの外部に あるコードは無視されます。 また、flags の設定により、 現在実行中の環境内の既存の関数やクラスを自動的に上書きします。

パラメータ

filename

関数やクラスの定義を読み込むファイル名。

flags

RUNKIT_IMPORT_* 系の定数の論理和。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

runkit_lint_file

(PECL runkit:0.7-0.9)

runkit_lint_file — 指定したファイルの PHP 文法をチェックする

説明

bool **runkit_lint_file** (string $filename)

runkit_lint_file() 関数は、指定したファイルの 文法チェック (lint) を行い、スクリプトのエラーをチェックします。 これは、コマンドラインから php -l を実行するのと同じです。

> **注意**: サンドボックスのサポート (runkit_lint(), **runkit_lint_file()**, および Runkit_Sandbox クラスで使用)は、PHP 5.1 以降または特別なパッチを適用した PHP 5.0 でのみ利用可能であり、スレッドセーフを有効にしておく必要があります。詳細については、runkit パッケージに含まれる README ファイルを参照してください。

パラメータ

filename

PHP コードを含む、チェック対象のファイル。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- runkit_lint()

---

runkit_lint

(PECL runkit:0.7-0.9)

runkit_lint — 指定した PHP コードの文法をチェックする

説明

bool **runkit_lint** (string $code)

runkit_lint() 関数は、指定した PHP コードの 文法チェック (lint) を行い、スクリプトのエラーをチェックします。 これは、コマンドラインから php -l を実行するのと同じですが、 **runkit_lint()** はファイル名ではなくコードそのものを 受け付けます。

> **注意**: サンドボックスのサポート (**runkit_lint()**, runkit_lint_file(), および Runkit_Sandbox クラスで使用)は、PHP 5.1 以降または特別なパッチを適用した PHP 5.0 でのみ利用可能であり、スレッドセーフを有効にしておく必要があります。詳細については、runkit パッケージに含まれる README ファイルを参照してください。

パラメータ

code

チェック対象の PHP コード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- runkit_lint_file()

---

runkit_method_add

(PECL runkit:0.7-0.9)

runkit_method_add — 指定したクラスに、新しいメソッドを動的に追加する

説明

bool **runkit_method_add** (string $classname , string $methodname , string $args , string $code [, int $flags])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

classname

メソッドを追加するクラスの名前。

methodname

追加するメソッドの名前。

args

カンマで区切った、新しいメソッドの引数。

code

methodname がコールされた際に 評価されるコード。

flags

作成するメソッドの型。 **RUNKIT_ACC_PUBLIC**、 **RUNKIT_ACC_PROTECTED** あるいは **RUNKIT_ACC_PRIVATE** のいずれか。

> **注意**: このパラメータは PHP 5 以降でのみ使用されます。なぜなら、それ以前の バージョンでは全てのメソッドが public だからです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 runkit_method_add() の例

```
<?php
class Example {
     function foo() {
         echo "foo!¥n";
     }
}

// Example のオブジェクトを作成
$e = new Example();

// 新しい public メソッドの追加
runkit_method_add(
     'Example',
     'add',
     '$num1, $num2',
     'return $num1 + $num2;',
     RUNKIT_ACC_PUBLIC
);

// 12 + 4 を計算する
echo $e->add(12, 4);
?>
```

上の例の出力は以下となります。

```
16
```

参考

- runkit_method_copy()
- runkit_method_redefine()
- runkit_method_remove()
- runkit_method_rename()
- runkit_function_add()

---

runkit_method_copy

(PECL runkit:0.7-0.9)

runkit_method_copy — あるクラスのメソッドを別のクラスにコピーする

説明

bool **runkit_method_copy** (string $dClass , string $dMethod , string $sClass [, string $sMethod])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

dClass

メソッドのコピー先のクラス。

dMethod

コピー先のメソッド名。

sClass

メソッドのコピー元のクラス。

sMethod

元のクラスからコピーするメソッドの名前。指定されなかった場合は dMethod と同じであるとみなされます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 runkit_method_copy() の例

```
<?php
class Foo {
     function example() {
         return "foo!¥n";
     }
}

class Bar {
    // 最初は、何もメソッドがない
}

// Foo クラスの example() メソッドを Bar クラスに baz() という名前でコピーする
runkit_method_copy('Bar', 'baz', 'Foo', 'example');

// コピーされた関数の出力
echo Bar::baz();
?>
```

上の例の出力は以下となります。

```
foo!
```

参考

- runkit_method_add()
- runkit_method_redefine()
- runkit_method_remove()
- runkit_method_rename()
- runkit_function_copy()

---

runkit_method_redefine

(PECL runkit:0.7-0.9)

runkit_method_redefine — 指定されたメソッドのコードを動的に変更する

説明

bool **runkit_method_redefine** (string $classname , string $methodname , string $args , string $code [, int $flags])

注意: この関数は、現在実行中(もしくはチェーンド)のメソッドを操作することはできません。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて

変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

classname

メソッドを再定義するクラス。

methodname

再定義するメソッドの名前。

args

カンマで区切られた、再定義後のメソッドの引数。

code

methodname がコールされた際に 評価される、新しいコード。

flags

再定義するメソッドの型。 **RUNKIT_ACC_PUBLIC**、 **RUNKIT_ACC_PROTECTED** あるいは **RUNKIT_ACC_PRIVATE** のいずれか。

> **注意**: このパラメータは PHP 5 以降でのみ使用されます。なぜなら、それ以前の バージョンでは全てのメソッドが public だからです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 runkit_method_redefine() の例

```
<?php
class Example {
     function foo() {
         return "foo!¥n";
     }
}

// Example オブジェクトの作成
$e = new Example();

// Example::foo() の出力（再定義前）
echo "Before: " . $e->foo();

// 'foo' メソッドの再定義
runkit_method_redefine(
     'Example',
     'foo',
     '',
     'return "bar!¥n";',
     RUNKIT_ACC_PUBLIC
);

// Example::foo() の出力（再定義後）
echo "After: " . $e->foo();
?>
```

上の例の出力は以下となります。

```
Before: foo!
After: bar!
```

参考

- runkit_method_add()
- runkit_method_copy()
- runkit_method_remove()
- runkit_method_rename()
- runkit_function_redefine()

---

runkit_method_remove

(PECL runkit:0.7-0.9)

runkit_method_remove — 指定したメソッドを動的に削除する

説明

bool **runkit_method_remove** (string $classname , string $methodname)

> **注意**: この関数は、現在実行中(もしくはチェーンド)のメソッドを操作することはできません。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

classname

メソッドを削除するクラス。

methodname

削除するメソッドの名前。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 runkit_method_remove() の例

```
<?php
class Example {
     function foo() {
         return "foo!¥n";
     }

     function bar() {
         return "bar!¥n";
     }
}

// 'foo' メソッドを削除する
runkit_method_remove(
     'Example',
     'foo'
);

echo implode(' ', get_class_methods('Example'));

?>
```

上の例の出力は以下となります。

```
bar
```

参考

- runkit_method_add()
- runkit_method_copy()
- runkit_method_redefine()
- runkit_method_rename()
- runkit_function_remove()

---

runkit_method_rename

(PECL runkit:0.7-0.9)

runkit_method_rename — 指定したメソッドの名前を動的に変更する

説明

bool **runkit_method_rename** (string $classname , string $methodname , string $newname)

注意: この関数は、現在実行中(もしくはチェーンド)のメソッドを操作することはできません。

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

classname

メソッド名を変更するクラス。

methodname

変更するメソッドの名前。

newname

変更後のメソッドの名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 runkit_method_rename() の例

```
<?php
class Example {
     function foo() {
         return "foo!¥n";
     }
}

// 'foo' メソッドの名前を 'bar' に変更する
runkit_method_rename(
     'Example',
     'foo',
     'bar'
);

// 変更後の関数の出力
echo Example::bar();
?>
```

上の例の出力は以下となります。

```
foo!
```

参考

- runkit_method_add()
- runkit_method_copy()
- runkit_method_redefine()
- runkit_method_remove()
- runkit_function_rename()

---

runkit_return_value_used

(PECL runkit:0.8-0.9)

runkit_return_value_used — 現在の関数の返り値が使用されているかどうかを調べる

説明

bool **runkit_return_value_used** (void)

返り値

呼び出し元のスコープで関数の返り値が使用されている場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

例

Example#1 runkit_return_value_used() の例

```
<?php
function foo() {
  var_dump(runkit_return_value_used());
}

foo();
$f = foo();
?>
```

上の例の出力は以下となります。

```
bool(false)
bool(true)
```

---

runkit_sandbox_output_handler

(PECL runkit:0.7-0.9)

runkit_sandbox_output_handler — サンドボックス内での出力を取得・処理するための関数を指定する

説明

mixed **runkit_sandbox_output_handler** (object $sandbox [, mixed $callback])

通常、 (echo() や print() などの) あらゆる出力は、親のスコープから出力しているかのように出力されます。 しかし、 **runkit_sandbox_output_handler()** を使用すると サンドボックス内の出力（エラーを含む）をサンドボックス外で受け取ることが 可能です。

注意: サンドボックスのサポート (runkit_lint(), runkit_lint_file(), および Runkit_Sandbox クラスで使用)は、PHP 5.1 以降または特別なパッチを適用した PHP 5.0 でのみ利用可能であり、スレッドセーフを有効にしておく必要があります。詳細については、runkit パッケージに含まれる README ファイルを参照してください。

注意: **非推奨** runkit バージョン 0.5 以降この関数は非推奨となっており、1.0 が リリースされるまでに削除される予定です。指定した Runkit_Sandbox インスタンスの出力ハンドラは、配列オフセット構文を使用して 取得/設定が可能です。この方法について、Runkit_Sandbox のクラス定義で説明しています。

パラメータ

sandbox

出力の処理を設定する Runkit_Sandbox クラスのインスタンス。

callback

ひとつのパラメータを受け取る関数の名前。 sandbox による出力がこのコールバックに 渡されます。コールバックが返す値は通常通り表示されます。 このパラメータが渡されなかった場合、出力の操作方法は変わりません。 真でない値が渡された場合、出力の操作は無効となり直接表示されます。

返り値

前に定義されていた出力ハンドラコールバックの名前を返します。 ハンドラが定義されていなかった場合は **FALSE** を返します。

例

Example#1 出力を変数に送る

```
<?php
function capture_output($str) {
  $GLOBALS['sandbox_output'] .= $str;

  return '';
}

$sandbox_output = '';

$php = new Runkit_Sandbox();
runkit_sandbox_output_handler($php, 'capture_output');
$php->echo("Hello¥n");
$php->eval('var_dump("Excuse me");');
$php->die("I lost myself.");
unset($php);

echo "Sandbox Complete¥n¥n";
echo $sandbox_output;
?>
```

上の例の出力は以下となります。

```
Sandbox Complete

Hello
string(9) "Excuse me"
I lost myself.
```

---

runkit_superglobals

(PECL runkit:0.7-0.9)

runkit_superglobals — 登録されているスーパーグローバルを、数値添字の配列で返す

説明

array **runkit_superglobals** (void)

返り値

登録されているスーパーグローバルを、数値添字の配列で返します。 スーパーグローバルとは、例えば _GET、_POST、_REQUEST、_COOKIE、_SESSION、_SERVER、_ENV、_FILES などです。

参考

- 変数のスコープ

目次

- runkit_constant_remove — 定義済みの定数を削除する
- runkit_function_add — 新しい関数を追加する。create_function と同じ
- runkit_function_copy — 関数を別の名前でコピーする
- runkit_function_redefine — 関数の定義を新しい実装で置き換える
- runkit_function_remove — 関数の定義を削除する
- runkit_function_rename — 関数名を変更する
- runkit_import — ファイルから関数やクラスの定義を読み込み、必要に応じて書き換える
- runkit_lint_file — 指定したファイルの PHP 文法をチェックする
- runkit_lint — 指定した PHP コードの文法をチェックする
- runkit_method_add — 指定したクラスに、新しいメソッドを動的に追加する
- runkit_method_copy — あるクラスのメソッドを別のクラスにコピーする
- runkit_method_redefine — 指定されたメソッドのコードを動的に変更する
- runkit_method_remove — 指定したメソッドを動的に削除する
- runkit_method_rename — 指定したメソッドの名前を動的に変更する
- runkit_return_value_used — 現在の関数の返り値が使用されているかどうかを調べる
- runkit_sandbox_output_handler — サンドボックス内での出力を取得・処理するための関数を指定する
- runkit_superglobals — 登録されているスーパーグローバルを、数値添字の配列で返す

SAM - Simple Asynchronous Messaging: 単純な非同期メッセージング

導入

この拡張モジュールを使用すると、IBM WebSphere MQSeries 製品群のようなメッセージングシステム・キューイングシステムに PHP スクリプトからアクセスできるようになります。このインターフェイスは、 一般的な操作（単純なテキストメッセージをキューに配送するなど）については限りなくシンプルに実行できるように設計されています。 また、熟練者向けには、より複雑な操作もできるようになっています。 大半のユーザが使用する場合には、複雑で膨大な設定オプションは無視することができます。

SAM 拡張モジュールは、さまざまなメッセージングミドルウェアシステムにアクセスするための手段として 非常にシンプルな API を提供するフレームワークです。 現在このパッケージに標準で組み込まれているのは、 MQTT (MQ Telemetry Transport) メッセージングプロトコルのサポート および IBM Messaging and Queuing ミドルウェア製品群のサポートです。 SAM は、その他のメッセージングシステムをサポートするために拡張できるよう設計されています。 この拡張モジュールは、C あるいは PHP で書くことができます。

インストール手順

SAM フレームワークおよび MQTT サポートをビルドして使用するためには、 特にその他の前提条件はありません。 MQTT 以外のプロトコルのサポートはライブラリとして提供されており、 クライアント側のコードの中には XMS を使用するものもあります。

組み込みの MQTT サポートを使用したいだけである場合は、 SAM を拡張モジュールとしてビルドして設定するか、 あるいは単に "php_sam.php" を PHP スクリプトから "requires" あるいは "requires_once" で読み込みます。この場合は、 コードをインストールするだけでよく、拡張モジュールをビルドする必要はありません。 これを行うには、pear インストーラで次のように指定します。

```
pecl install -B SAM
```

前提条件

SAM 拡張モジュールは、IBM Messaging and Queuing ミドルウェア製品とのインターフェイスとして XMS のライブラリおよびクライアント側のコードを使用します。 このパッケージは、IBM support pack IA94 としてフリーでダウンロードできます。 このパッケージについての説明、そしてダウンロード用のリンクは » Introducing XMS - The IBM Message Service API にあります。

SAM を使用して WebSphere MQ のメッセージングおよびキューイング機能にアクセスしたい場合は、 ローカル MQ キューマネージャあるいは WebSphere MQ クライアントパッケージをインストールする必要があります。 クライアントパッケージは、サポートパック (» MQC6) としてフリーで公開されています。

単に、WebSphere Application Server のキューに対して WebSphere Platform Messaging protocol (WPM) を使用したメッセージの送受信を行いたいだけなら、MQC6 パッケージをインストールする必要はありません。

これらのパッケージをインストールしたら、次に XMS のバイナリや（もし使用するなら）MQ クライアントの bin ディレクトリが PATH 環境変数に含まれていることを確認しましょう。これにより、 Apache や PHP から関連する DLL/ライブラリ を見つけられるようになります。

Linux でのインストール手順

SAM 拡張モジュールは PECL モジュールとして提供され、 次の手順でダウンロード、インストールすることができます。

```
pear install sam
```

(php の環境にもよりますが、おそらく root になる必要があるでしょう)

このモジュールを PHP に読み込ませるために、 php.ini に以下を追加します。

```
extension=sam.so
```

XMS サポートを使用して IBM Messaging and Queuing family にアクセスしたい場合は、 SAM XMS 拡張モジュールも有効にしなければなりません。

```
extension=sam_xms.so
```

PEAR インストーラを使用できない場合は、 拡張モジュールをダウンロードして手動でビルドします。

```
pear download sam            # sam-<version>.tgz をダウンロードします
tar  -xzf  sam-<version>.tgz
cd  sam-<version>
phpize
./configure
make
make install                # おそらく root になる必要があるでしょう
```

最新のソースを使用したい場合は、cvs からソースを取得して、 上の手順で手動ビルドします。

Windows でのインストール手順

SAM のウェブサイトにあるビルド済みバイナリは、非常に限られた範囲のものになっています。 そのため、おそらく自分でビルドすることになるでしょう。この拡張モジュールのビルド手順は、 Windows 上での標準的な拡張モジュールビルド手順と同じです。

SAM 拡張モジュールを使用する予定の PHP と同じバージョンの、 PHP 自身のソースツリーが必要となります。これは php.net から取得します。取得したソースを、どこかの作業ディレクトリに展開します。

また、PHP 拡張モジュールが使用するライブラリやヘッダを http://www.php.net/extra/win32build.zip から取得し、 作業ディレクトリの下に展開します。

するとこのような状態になります。

```
c:¥php-build¥-
                |
                  |---php-5.0.5--|---build
                |                |---ext
                |                |--- ...
                |
                  |---win32build--|---bin
                                  |---include
                                  |---lib
```

コンパイラが必要です。たとえば、フリーの Visual Studio C++ Express が Microsoft のウェブサイトから取得可能です。 また、Microsoft Windows Platform SDK も必要となります。 これも Microsoft のウェブサイトからダウンロードできます。

SAM 拡張モジュールのソースを、pear を使用して (pear download sam) あるいは CVS から取得し、それを PHP ソースツリーの "ext" ディレクトリ内に作成した "sam" ディレクトリに配置します。

この拡張モジュールをビルドするには、ビルド環境を次の場所から開きます。 start menu->all programs->microsoft platform SDK for windows-> open build environment window->windows 200 build environment-> set windows 2000 build environment (retail)

これは、コマンドプロンプトを開き、プラットフォーム SDK などを使用するためのすべての環境変数を設定します。 次に、Visual Studio 用の環境変数を設定するために、このウィンドウでコマンド "vcvars32.bat" を実行します。

cd c:\php-build などとして、作業ディレクトリに移動します。 そして win32build ツール群にアクセスできるよう、そのパスを環境変数 PATH に追加します。

```
set  PATH=..¥win32build¥bin;%PATH%
```

buildconf.bat コマンドを実行します。これは configure.js ファイルを作成します。

cscript コマンドに適切なオプションをつけて実行します。 SAM 拡張モジュールフレームワークと MQTT サポートだけをビルドするのなら

```
cscript  /nologo  configure.js  --with-sam
```

SAM フレームワークおよび XMS サポートをビルドするのなら

```
cscript /nologo configure.js --with-sam --with-sam_xms="c:¥program files¥ibm¥xms"
```

とします。sam_xms に渡している追加のパラメータは、XMS ライブラリやランタイムへのパスです。 これらは、このファイルの最初に書いてある前提条件によってインストールされているものです。

cscript のパラメータとして、php のビルドオプションのうちお好みのものを追加したり削除することもできます。

すべてがうまくいけば、あとは SAM フレームワーク用の make を実行するだけです!

```
nmake php_sam.dll
```

また、もし XML サポートを使用するのなら、sam_xms 拡張モジュールも make しなければなりません。

```
nmake php_sam_xms.dll
```

Visual Studio 2005 を使用して DLL をビルドする場合は、これ以降に進む前に次の追加手順を実行しなければなりません。

作成された DLL (php_sam.dll およびオプションで php_sam_xms.dll) を、PHP をセットアップしたディレクトリ以下の適切な場所にコピーします。このモジュールを PHP に読み込ませるために、次の行を php.ini に追加しておきましょう。

```
extension=php_sam.dll
```

XMS サポートを使用して IBM Messgaing and Queuing family にアクセスしたい場合は SAM XMS 拡張モジュールも有効にする必要があります。

```
extension=php_sam_xms.dll
```

Visual Studio 2005 用の追加手順

SAM 拡張モジュールを Microsoft Visual Studio 2005 のコンパイラでビルドしたい場合は、 さらに追加の手順が必要です。これにより、php_sam.dll が実行時に C ランタイムライブラリとリンクできるようにします。 この追加手順では、依存性マニフェストを DLL に組み込みます。php_sam.dll が作成されたディレクトリ (通常は、php ソースディレクトリ配下の Release_TS あるいは Debug_TS) に移動し、次のように謎の呪文をとなえます。

```
mt.exe -manifest php_sam.dll.manifest -outputresource:php_sam.dll;2
```

XMS 機能を使用する場合は、同様に SAM XMS DLL も指定しなければなりません。

```
mt.exe -manifest php_sam_xms.dll.manifest -outputresource:php_sam_xms.dll;2
```

SAM 拡張モジュールを Microsoft Visual Studio 2005 のコンパイラやライブラリでビルドした場合は、 ランタイムコンポーネントが SAM を使用する予定のシステムにインストールされている必要があります。 そのためには、そこに Visual Studio 2005 をインストールするか、あるいはフリーで配布されている » ランタイムパッケージ を使用します。

プロトコルサポートおよびマッピング

SAM フレームワークを拡張して、他のメッセージングプロトコルや通信機構をサポートさせることができます。新しいプロトコルや接続ライブラリのサポートを追加するには、それをサポートするクラスを定義する必要があります。これは C 拡張モジュールあるいは PHP スクリプトとして作成します。また、"ファクトリ" スクリプトも作成しなければなりません。サポートクラスは、SAMConnection クラスの全メソッドを実装する必要があります。 SAMConnection を継承してはいけません。ファクトリスクリプトはフレームワークからコールされ、実装クラスのインスタンスを作成します。SAM がどのファクトリをコールするのかは、 "connect" の最初のパラメータで指定したプロトコルによって決まります。

デフォルトでは、connect でプロトコルに SAM_MQTT ("mqtt") を指定した場合に組み込みの MQTT サポートを使用します。その他のプロトコルを指定した場合は XMS 拡張モジュールを使用します。新たなプロトコルのサポートを追加したりデフォルトの挙動を変更したりするには、 php.ini の [sam] セクションにエントリを追加します。デフォルトのマッピングは、以下のエントリと同等です。

```
[sam]
sam.factory.mqtt=mqtt
sam.factory.wmq=xms
sam.factory.wmq:client=xms
sam.factory.wmq:bindings=xms
```

```
sam.factory.wpm=xms
sam.factory.rtt=xms
```

これらの例でわかるように、エントリは "sam.factory.pppp=xxx" という形式になります。pppp の部分が connect コール時に指定するプロトコル文字列で、 xxx はファクトリ名のサフィックスです。注意: SAM では、これらのプロトコルに対応する定数を定義しています。たとえば SAM_WMQ=wmq、SAM_WPM=wpm、SAM_RTT=rtt、SAM_MQTT=mqtt などとなります。

connect コールの際にこれを指定すると、 SAM はそのプロトコル名を php.ini のエントリから探し、対応するファクトリスクリプト sam_factory_xxx.php をコールします。エントリが見つからなかった場合は、デフォルトは XMS となります。

API の使用法

接続

メッセージングやキューイングの関数を実行するには、 メッセージングサーバとの接続が確立されていなければなりません。そのためには、SAMConnection オブジェクトを作成してその "connect" メソッドを実行します。 その際に、接続用のプロパティを指定します。SAMConnection オブジェクトが破棄されるまでの間は、接続が使用可能となります。 SAMConnection オブジェクトは、PHP スクリプトが終了する際に破棄されます。

メッセージングサーバに接続する際にはデフォルトのプロパティを使用することもできますが、 最低限、使用するプロトコルだけは PHP スクリプトで指定しなければなりません。

Example#1 接続の作成、およびリモートの WebSphere MQSeries Messaging Server への接続

```php
<?php
    $conn = new SAMConnection();
   $conn->connect(SAM_WMQ, array(SAM_HOST => myhost.mycompany.com,
                                 SAM_PORT => 1506,
                                 SAM_BROKER => mybroker));
?>
```

Example#2 接続の作成、およびリモートの WebSphere Application Server への接続

```php
<?php
    $conn = new SAMConnection();
   $conn->connect(SAM_WMQ, array(SAM_ENDPOINTS => 'localhost:7278:BootstrapBasicMessaging',
                                 SAM_BUS => 'Bus1',
                                 SAM_TARGETCHAIN => 'InboundBasicMessaging'));
?>
```

Example#3 接続の作成、および MQTT サーバへの接続

```php
<?php
    $conn = new SAMConnection();
   $conn->connect(SAM_MQTT, array(SAM_HOST => 'myhost.mycompany.com',
                                 SAM_PORT => 1883));
?>
```

メッセージ

キューとのメッセージの送受信を行うのが SAMMessage オブジェクトです。 SAMMessage オブジェクトは、メッセージ本文（もしあれば）とそのメッセージに関連するヘッダプロパティをカプセル化します。 SAMMessage オブジェクトは、メッセージング操作の際のパラメータ、 そしてその返り値の両方に用いられます。

Example#4 単純なテキストのメッセージの作成

```php
<?php
    $msg = new SAMMessage('This is a simple text message');
?>
```

メッセージにはヘッダプロパティを関連付けることができます。 これは、メッセージの配送方法を制御したり、 受け取り側のアプリケーションに詳しい情報を提供したりするものです。 デフォルトでは、メッセージのプロパティは文字列として配送されます。 この場合は、プロパティを設定する際に次のような簡単な構文を使用します。

Example#5 デフォルト構文による、テキスト形式のプロパティの設定

```php
<?php
    $msg->header->myPropertyName = 'textData';
?>
```

デフォルト以外の構文で使用する型情報を渡す場合は、値と型情報を連想配列で渡します。

Example#6 型ヒントを使用したプロパティの設定

```php
<?php
    $msg->header->myPropertyName = array(3.14159, SAM_FLOAT);
?>
```

プロパティは、メッセージのヘッダから取り出すことができます。

Example#7 メッセージのヘッダからのプロパティの取得

```php
<?php
    $myProperty = $msg->header->myPropertyName;
?>
```

メッセージング操作

すべてのメッセージング操作は、接続オブジェクトのメソッドをコールすることで行います。 キューにメッセージを追加するには "send" メソッドを使用し、 キューからメッセージを取得するには "receive" メソッドを使用します。 その他のメソッドには、配信や購読機能、 そしてトランザクションの境界の制御などがあります。

Example#8 キューへのメッセージの追加および応答の取得

```php
<?php
    $msg = new SAMMessage('This is a simple text message');
    $msg->header->SAM_REPLY_TO = 'queue:///receive/test';
    $correlid = $conn->send('queue:///send/test', $msg);

    if (!$correlid) {
      // 送信に失敗しました!
      echo "Send failed ($conn->errno) $conn->error";
    } else {
      $resp = $conn->receive('queue:///receive/test', array(SAM_CORRELID => $correlid));
    }
?>
```

配信/購読およびトピックの購読

SAM では、メッセージをキューに送信するか、あるいは WebSphere MQ および WPM ではトピックに配信/購読することができます。 トピックを SAM に指定するには、場所を指定する際に 'queue://AQUEUE' ではなく 'topic://fred' 形式を使用します。 配信/購読 機能を使用するには、正しいブローカ名を CAMConnect の "connect" コール時に指定し、対象のトピックを SAMConnect の "send" および "receive" コール時に指定する必要があります。 それ以外については、PHP のインターフェイスは point to point モデルと同じです。

デフォルトでは、SAM は永続的でない購読を作成して配信/購読を行います。 つまり、メッセージをトピックに配信しているときにクライアントアプリケーションがアクティブでなくなると、 その後アプリケーションが再開しても受信処理は行われないということです。 WPM あるいは WebSphere MQ の配信/購読 を使用している場合は、 SAM でトピックへの永続的な購読を作成することができます。 こうすると、データの配信時にクライアントがアクティブでなかったとしても、 アプリケーションでデータを受信することができるようになります。

永続的な購読を指定するには SAMConnect の "subscribe" コールを使用します。 このメソッドは、対象となるトピックを入力パラメータとして受け取り、 購読 ID を返します。この ID を使用して、"receive" コールを行います。 購読が不要になった場合は、SAMConnection の "unsubscribe" メソッドを使用して購読を削除します。

Example#9 トピックに対する永続的な購読の作成

```php
<?php

       $subName = $conn->subscribe('topic://A');

       if (!$subName) {
          echo "購読に失敗しました";
       } else {
         # 購読に成功しました
          ...
       }
?>
```

Example#10 WebSphere Platform Messaging (WPM) サーバを使用したトピックの購読

```php
<?php
    $conn = new SAMConnection();
   // 注意: WPM での配信/購読では、永続的な購読を保持するメッセージングエンジン
    //       (SAM_WPM_DUR_SUB_HOME) を接続の際に指定する必要があります
    $conn->connect(SAM_WMQ, array(SAM_ENDPOINTS => 'localhost:7278:BootstrapBasicMessaging',
                                  SAM_BUS => 'Bus1',
                                  SAM_TARGETCHAIN => 'InboundBasicMessaging'
                                  SAM_WPM_DUR_SUB_HOME => 'MyMachineNode01.server1-Bus1'));

    $subName = $conn->subscribe('topic://A');

    if (!$subName) {
       echo "購読に失敗しました";
    } else {
      # 購読に成功しました
       ...
    }
?>
```

Example#11 永続的な購読による、配信されたデータの受信

```php
<?php

       $msg = $conn->receive($subName);
       if ($msg) {
          echo "メッセージの受信に成功しました";
       } else {
          echo "受信に失敗しました";
       }

?>
```

Example#12 トピックへの永続的な購読の削除

```php
<?php

       if (!$conn->unsubscribe($subName)) {
          echo "購読解除に失敗しました";
       }

?>
```

エラー処理

SAMConnection のメソッドでメッセージング操作を行うものはすべて、 リクエスト処理中にエラーが発生すると **FALSE** を返します。 さらに SAMConnection オブジェクトはふたつのプロパティ "errno" および "error" を保持しています。これらはそれぞれ、 接続上でおこった最後のエラーのエラー番号およびエラー内容のテキストを表します。

Example#13 結果を返さないメソッドからのエラーの処理

```php
<?php
    if (!$conn->commit()) {
      // コミットに失敗しました!
```

```
    echo "Commit failed ($conn->errno) $conn->error";
  }
?>
```

Example#14 結果を返すメソッドからのエラーの処理

```
<?php
  $correlid = $conn->send('queue://send/test', $msg);

  if (!$correlid) {
    // 送信に失敗しました!
    echo "Send failed ($conn->errno) $conn->error";
  } else {
    ...
  }
?>
```

定義済みクラス

SAMConnection

メッセージングサーバとの接続を表すオブジェクトです。

コンストラクタ

- new SAMConnection - 新しい接続オブジェクトを作成し、メッセージング環境への接続を可能にします。

メソッド

- commit - 現在作業中の内容をコミット（正常に完了）するメソッドです。
- connect - PHP スクリプトをメッセージングサーバに接続させるメソッドです。
- disconnect - PHP スクリプトとメッセージングサーバとの接続を解除するメソッドです。
- isConnected - PHP スクリプトがメッセージングサーバと接続しているかどうかを調べるメソッドです。
- peek - メッセージをキューから取得し、それをキューに残したままにしておくメソッドです。
- peekAll - ひとつあるいは複数のメッセージをキューから取得し、それをキューに残したままにしておくメソッドです。
- receive - メッセージをキューあるいは購読から取得するメソッドです。
- remove - メッセージをキューから削除するメソッドです。
- rollback - 現在作業中の内容をキャンセル（ロールバック）するメソッドです。
- send - キューにメッセージを送信したり、トピックの投稿したりするメソッドです。
- setDebug - 追加のデバッグ出力をオンあるいはオフにするメソッドです。
- subscribe - ひとつあるいは複数のトピックを購読するためのメソッドです。
- unsubscribe - ひとつあるいは複数のトピックの購読を解除するメソッドです。

プロパティ

- errno - この接続で最後に発生したエラーのエラーコードを表す数値です。 直近の処理が正常に終了した場合は、このプロパティは 0 となります。
- error - この接続で最後に発生したエラーの説明テキストです。

SAMMessage

送受信するメッセージを表すオブジェクトです。

コンストラクタ

- new SAMMessage - 新しいメッセージを作成します。

プロパティ

- body - メッセージの本文です。
- header - メッセージのヘッダプロパティです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

SAM_AUTO (string)
自動的に処理します。
SAM_BOOLEAN (string)
型指定子。SAM_Message オブジェクトにプロパティを設定する際に使用します。
SAM_BUS (string)
接続の属性で、接続するエンタープライズサービスバスの名前を設定します。
SAM_BYTE (string)
型指定子。SAM_Message オブジェクトにプロパティを設定する際に使用します。
SAM_BYTES (string)

メッセージ本文の型指定子。
SAM_CORRELID (string)
リクエストを受信、送信および削除する際に使用する属性で、 特定のメッセージを指定します。
SAM_DELIVERYMODE (string)
メッセージヘッダのプロパティです。
SAM_DOUBLE (string)
型指定子。SAM_Message オブジェクトにプロパティを設定する際に使用します。
SAM_ENDPOINTS (string)
接続の属性で、接続先エンドポイントを指定します。
SAM_FLOAT (string)
型指定子。SAM_Message オブジェクトにプロパティを設定する際に使用します。
SAM_HOST (string)
接続の属性で、メッセージングサーバのホスト名を設定します。
SAM_INT (string)
型指定子。SAM_Message オブジェクトにプロパティを設定する際に使用します。
SAM_LONG (string)
型指定子。SAM_Message オブジェクトにプロパティを設定する際に使用します。
SAM_MANUAL (string)
手動で（スクリプトで制御して）処理します。
SAM_MESSAGEID (string)
リクエストを受信および削除する際に使用する属性で、 特定のメッセージを指定します。
SAM_MQTT (string)
MQTT (MQ Telemetry Transport) プロトコルを使用するための 接続プロトコル定義です。
SAM_MQTT_CLEANSTART (bool)
任意で指定する接続オプションで、MQTT サーバに対して このクライアントに関する事前のすべての接続データの削除を指示します。 また、クライアントとの接続が明示的に解除されたり 予期せず解除されたりしたら、購読を削除するよう指示します。
SAM_NON_PERSISTENT (string)
接続の属性で、リクエストメッセージをメッセージングサーバ上で 持続させないことを指定します。
SAM_PASSWORD (string)
接続の属性で、接続時に認証が必要なメッセージングサーバに対する 接続パスワードを指定します。
SAM_PERSISTENT (string)
接続の属性で、リクエストメッセージをメッセージングサーバ上で 持続させることを指定します。これにより、 処理に失敗した際にメッセージを失うことを防ぎます。
SAM_PORT (string)
接続の属性で、メッセージングサーバと接続する際のポート番号を設定します。
SAM_PRIORITY (string)
リクエストを配送する際の優先度を設定するオプション名です。
SAM_REPLY_TO (string)
メッセージのプロパティで、 そのスクリプトが応答あるいは返信の配送先として想定しているキューを指定します。
SAM_RTT (string)
接続プロトコルを指定します。IBM Realtime Transport プロトコルを使用してビジネスインテグレーションメッセージングサーバと通信します。
SAM_STRING (string)
型指定子。SAM_Message オブジェクトにプロパティを設定する際に使用します。
SAM_TARGETCHAIN (string)
接続の属性で、ターゲットチェイン識別子を指定します。
SAM_TEXT (string)
メッセージ本文の型指定子。
SAM_TIMETOLIVE (string)
メッセージ送信時のオプション名。 メッセージを保持し続ける時間をミリ秒で指定します。
SAM_TRANSACTIONS (string)
接続の属性で、トランザクションの振る舞いを指定します。SAM_AUTO（デフォルト）あるいは SAM_MANUAL のいずれかです。
SAM_USERID (string)
接続の属性で、接続時に認証が必要なメッセージングサーバに対する 接続アカウントを指定します。
SAM_WAIT (string)
受信時のプロパティで、キューや購読からメッセージを取得する際のタイムアウトを指定します。
SAM_WMQ (string)
接続プロトコルを指定します。IBM WebSphere MQSeries プロトコルを使用してメッセージングサーバと通信します。
SAM_WMQ_BINDINGS (string)
接続プロトコルを指定します。IBM WebSphere MQSeries プロトコルを使用してローカルのメッセージングサーバと通信します。
SAM_WMQ_CLIENT (string)
接続プロトコルを指定します。IBM WebSphere MQSeries プロトコルを使用してリモートのメッセージングサーバと通信します。
SAM_WMQ_TARGET_CLIENT (string)
リクエストを送信する際に使用するオプション名です。 ターゲットクライアントモードを指定します。デフォルトの 'jms' あるいは 'mq' のいずれかを指定します。 デフォルトの 'jms' は、RFH2 ヘッダがメッセージとともに送信されることを意味します。 一方、'mq' の場合は RFH2 は含まれません。
SAM_WPM (string)
接続プロトコルを指定します。IBM WebSphere Platform Messaging プロトコルを使用して WebSphere Application Server メッセージングサーバと通信します。

SAMConnection->commit()

(No version information available, might be only in CVS)

SAMConnection->commit() — 現在作業中の内容をコミット（正常に完了）する

説明

SAMConnection
bool **commit** (void)

接続オブジェクトに対して "commit" メソッドをコールすると、 現在処理中のトランザクションのすべての内容をコミットします。

返り値

エラーが発生した場合に **FALSE** を返します。

例

Example#1 現在処理中の内容のコミット

```
<?php
  if (!$conn->commit()) {
    // コミットに失敗しました!
    echo "Commit failed ($conn->errno) $conn->error";
  }
?>
```

参考

- SAMConnection->rollback()

SAMConnection->connect()

(No version information available, might be only in CVS)

SAMConnection->connect() — メッセージングサーバとの接続を確立する

説明

SAMConnection
bool **connect** (string $protocol [, array $properties])

SAMConnection オブジェクトに対して "connect" メソッドをコールすると、 PHP スクリプトをメッセージングサーバと接続させます。 接続が作成されるまでは、メッセージの送受信は行われません。

パラメータ

protocol

構造化文字列で、メッセージングサーバに接続する際に使用するプロトコル指定します。 サポートされるプロトコルについては、それぞれ対応する定数が存在します。 通常はこの定数を使用しますが、 お望みに応じてスクリプト中で動的に文字列を作成することもできます。 この文字列はふたつの部分からなります。最初の部分がプロトコル識別子で、 その次がプロトコルのオプションです。このふたつをコロン (:) で連結します。使用可能な値は次の表のとおりです。

プロトコル文字列	定数	使用法
mqtt	SAM_MQTT	MQTT (MQ Telemetry Transport) プロトコルを使用して クライアントからサーバに接続します。
wmq	SAM_WMQ	IBM MQSeries プロトコルを使用してリモートのメッセージングサーバに接続します。 これは "wmq:client" (SAM_WMQ_CLIENT) と同じです。 オプションの配列に、ブローカ名 (SAM_BROKER) を指定する必要があります。
wmq:client	SAM_WMQ_CLIENT	IBM MQSeries プロトコルを使用してリモートのメッセージングサーバに接続します。 オプションの配列に、ブローカ名 (SAM_BROKER) を指定する必要があります。
wmq:bindings	SAM_WMQ_BINDINGS	IBM MQSeries プロトコルを使用してローカルのメッセージングサーバに接続し、 共有メモリを使用して通信します。 オプションの配列に、ブローカ名 (SAM_BROKER) を指定する必要があります。
wpm	SAM_WPM	IBM WebSphere Platform Messaging プロトコルを使用して、 WebSphere Application サーバあるいはクラスタのメッセージングシステムに接続します。 オプションの配列に、バス名 (SAM_BUS) を指定する必要があります。 また、エンドポイントとターゲットチェインも指定します。
rtt	SAM_RTT	IBM Realtime トランスポートプロトコルを使用して、 メッセージングシステムに接続します。

properties

オプションの連想配列で、接続の際に必要な詳細情報を表すプロパティを指定します。 以下の表に、使用できるプロパティ名とその値をまとめます。

プロパティ名	デフォルト値	使用法
SAM_BROKER	none	メッセージングサーバ上で稼動するブローカあるいはキューマネージャの名前。 このプロパティは、WebSphere MQSeries プロトコル (SAM_WMQ, SAM_WMQ_CLIENT, SAM_WMQ_BINDINGS) を使用する場合に必須となります。
SAM_HOST	localhost	メッセージングサーバが稼動しているマシンのホスト名。
SAM_PORT	SAM_WMQ の場合は 1414、 SAM_WPM の場合は 1506、 SAM_MQTT の場合は 1883	メッセージングサーバに接続する際のポート番号。
SAM_EXPIRE_AFTER	0	メッセージが有効期限切れとみなされ、キューから削除されるまでの時間を ミリ秒で指定します。デフォルト値は 0 で、これは決して期限切れにならないことを意味します。 警告: まだ実装されていません!
SAM_MESSAGE_PERSISTENCE	none	メッセージを、配送中に持続させるかどうかを指定します。 SAM_PERSISTENT あるいは SAM_NON_PERSISTENT のいずれかです。 デフォルトは、接続の形式と接続先メッセージングサーバの機能に依存します。 警告: まだ実装されていません!
SAM_MQTT_CLEANSTART	**FALSE**	任意で指定する接続オプションで、 MQTT サーバに対してこのクライアントのこれまでの接続データを削除するよう指示します。 また、クライアントが明示的あるいは不意に接続を切断した際には購読を削除するようにします。 この値を **TRUE** にすると、クライアントが接続した際に SAM は既存の購読を破棄します。また、クライアントが接続を切断する際にも購読を破棄します。このオプションを **FALSE** に設定するかあるいはデフォルトを使用すると、 購読は永続的なものとなり、クライアントの接続が切断されても生き残ります。
SAM_PASSWORD	none	接続先メッセージングサーバが認証を要求した際に使用するパスワードを指定します。
SAM_TRANSACTIONS	SAM_AUTO	この接続でのトランザクションの処理方法を指示します。 SAM_AUTO (デフォルト) の場合は、操作単位で自動的にトランザクションを処理します。 SAM_MANUAL の場合は、PHP スクリプトでトランザクションの範囲を制御できるようになります。 SAM_MANUAL を使用している場合は、コミットを行わずにスクリプトが終了したり コミットを行わずに接続が閉じられた場合に、処理中のトランザクションがロールバックされます。
SAM_USERID	none	接続先メッセージングサーバが認証を要求した際に使用するユーザ ID を指定します。
SAM_WPM_DUR_SUB_HOME	none	永続的な購読を管理するメッセージングエンジンの名前 (WPM のみ)。

返り値

このメソッドは、エラーが発生した場合に **FALSE** を返します。

例

Example#1 IBM MQSeries プロトコル (WMQ) を使用したメッセージングサーバへの接続の作成

```
<?php

$conn->connect(SAM_WMQ, array(SAM_HOST => 'Myhost.myco.com', SAM_PORT => 1506, SAM_BROKER => 'MyBroker'));

?>
```

Example#2 トランザクション制御を指定し、 デフォルトのホストとポートを使用した接続の作成

```
<?php

$conn->connect(SAM_WMQ, array(SAM_BROKER => 'MyBroker', SAM_TRANSACTIONS => SAM_MANUAL));

?>
```

Example#3 IBM WebSphere Platform Messaging プロトコル (WPM) を使用したメッセージングサーバへの接続の作成

```
<?php

$conn->connect(SAM_WPM, array(SAM_ENDPOINTS => 'localhost:7278:BootstrapBasicMessaging',
                              SAM_BUS => 'Bus1', SAM_TARGETCHAIN => 'InboundBasicMessaging'));

?>
```

参考

- SAMConnection->isConnected()
- SAMConnection::disconnect()

SAMConnection->__construct()

(No version information available, might be only in CVS)

SAMConnection->__construct() — メッセージングサーバへの新しい接続を作成する

説明

SAMConnection
__construct *($)*

新しい SAMConnection オブジェクトを作成します。

例

Example#1 接続オブジェクトの作成およびメッセージングサーバへの接続

```
<?php

$conn = new SAMConnection();
$conn->connect(SAM_WMQ, array(SAM_HOST => localhost, SAM_PORT => 1414, SAM_BROKER => 'bull'));

?>
```

SAMConnection::disconnect()

(No version information available, might be only in CVS)

SAMConnection::disconnect() — メッセージングサーバからの接続を解除する

説明

SAMConnection
bool ***disconnect*** *(void)*

SAMConnection オブジェクトに対して "disconnect" メソッドをコールすると、 PHP スクリプトとメッセージングサーバとの接続を解除します。接続が解除された後は、メッセージの送受信はできません。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

例

Example#1 メッセージングサーバからの接続の解除

```
<?php

$conn->disconnect();

?>
```

参考

- SAMConnection->isConnected()
- SAMConnection->connect()

SAMConnection->errno

(No version information available, might be only in CVS)

SAMConnection->errno — 最後に実行した SAM 操作のエラーコードを表す数値を含む

説明

SAMConnection
int$errno;

errno() には、この接続で最後に実行された SAM 操作のエラーコードを表す数値が含まれます。 直近の操作が正常に終了した場合は、このプロパティは 0 となります。

返り値

ゼロより大きい整数値は、この接続の直近のエラーの型を表します。 ゼロの場合は、この接続の直近の操作が正常に終了したことを表します。

例

Example#1 Using the error number and description properties

```
<?php
$conn = new SAMConnection();
$conn->connect(SAM_WMQ, array(SAM_HOST => 'localhost', SAM_PORT => 1506));
$msg = new SAMMessage('This is a simple text item');
if (!$conn->send('topic://test', $msg)) {
    // The Send failed!
    echo "Send failed ($conn->errno) $conn->error";
}
?>
```

参考

- SAMConnection->error

SAMConnection->error

(No version information available, might be only in CVS)

SAMConnection->error — 最後に実行した SAM 操作のエラーの説明を含む

説明

SAMConnection
string$error;

この接続で最後に実行された SAM 操作のテキストによる説明が含まれます。 直近の操作が正常に終了した場合は、このプロパティには空の文字列が含まれます。

返り値

この接続で発生した最後のエラーの内容を表すテキストを返します。 空の文字列は、この接続の直近の操作が正常に終了したことを表します。

例

Example#1 エラー番号および説明の取得法

```
<?php
$conn = new SAMConnection();
$conn->connect(SAM_WMQ, array(SAM_HOST => 'localhost', SAM_PORT => 1506));
$msg = new SAMMessage('This is a simple text item');
if (!$conn->send('topic://test', $msg)) {
    // 送信に失敗しました!
    echo "Send failed ($conn->errno) $conn->error";
}
?>
```

参考

- SAMConnection->errno

SAMConnection->isConnected()

(No version information available, might be only in CVS)

SAMConnection->isConnected() — メッセージングサーバとの接続が確立されているかどうかを調べる

説明

SAMConnection
bool **isConnected** (void)

接続オブジェクトに対して "isConnected" メソッドをコールすると、 PHP スクリプトがメッセージングサーバと接続されているかどうかを調べま

す。 このメソッドは、SAMConnection オブジェクトが正常にメッセージングサーバに接続できた場合に TRUE、それ以外の場合に FALSE を返します。 メッセージングサーバとの接続が確立されない限り、 メッセージの送受信はできません。

返り値

このメソッドは、SAMConnection オブジェクトが メッセージングサーバに正しく接続できた場合に TRUE、 それ以外の場合に FALSE を返します。

例

Example#1 メッセージングサーバとの接続が存在するかどうかのチェック

```
<?php

if ($conn->isConnected()) {
  echo '接続はアクティブです。'
} else {
  echo 'アクティブな接続がありません!'
}

?>
```

参考

- SAMConnection::disconnect()
- SAMConnection->connect()

SAMConnection->peek()

(No version information available, might be only in CVS)

SAMConnection->peek() — メッセージをキューから読み込み、それをキューに残したままにする

説明

SAMConnection
SAMMessage **peek** (string $target [, array $properties])

パラメータ

target

メッセージを取得するキューの識別子。

properties

オプションの連想配列で、読み込み操作を制御するパラメータを指定します。

プロパティ名	とりうる値
SAM_CORRELID	これは、取得するメッセージの相関 ID 文字列です。 典型的な例は、"send" リクエストが返す相関 ID 文字列となります。
SAM_MESSAGEID	これは、取得するメッセージのメッセージ ID です。

返り値

このメソッドは SAMMessage オブジェクトを返します。 エラーが発生した場合は FALSE を返します。

例

Example#1 キューから次のメッセージを取得し、そのままメッセージをキューに残す

```
<?php
$msg = $conn->peek('queue://receive/test');

if (!$msg) {
  // 取得に失敗しました!
  echo "Peek failed ($conn->errno) $conn->error";
}
?>
```

Example#2 キューから指定したメッセージを取得し、そのままメッセージをキューに残す

```
<?php

$msg = $conn->peek('queue://receive/test', array(SAM_MESSAGEID => $messageId));

?>
```

参考

- SAMConnection->peekAll()

SAMConnection->peekAll()

(No version information available, might be only in CVS)

SAMConnection->peekAll() — ひとつあるいは複数のメッセージをキューから読み込み、それをキューに残したままにする

説明

```
SAMConnection
array peekAll ( string $target [, array $properties ] )
```

パラメータ

target

メッセージを取得するキューの識別子。

properties

オプションの連想配列で、読み込み操作を制御するパラメータを指定します。

プロパティ名	とりうる値
SAM_CORRELID	これは、取得するメッセージの相関 ID 文字列です。 典型的な例は、"send" リクエストが返す相関 ID 文字列となります。
SAM_MESSAGEID	これは、取得するメッセージのメッセージ ID 文字列です。

返り値

このメソッドは SAMMessage オブジェクトを返します。 エラーが発生した場合は **FALSE** を返します。

例

Example#1 キューからすべてのメッセージを取得し、そのままメッセージをキューに残す

```
<?php
$msgArray = $conn->peekAll('queue://receive/test');
if ($msgArray) {
    foreach ( $msgArray as $key => $msg) {
        echo "Message $key: body = $msg->body¥n";
    }
} else {
    echo "PeekAll failed ($conn->errno) $conn->error";
}
?>
```

Example#2 相関 ID にマッチするメッセージをキューから取得する

```
<?php

    $msgArray = $conn->peekAll('queue://receive/test', array(SAM_CORRELID => $correlId ));
    if ($msgArray) {

        foreach ( $msgArray as $key => $msg) {
              echo "Message $key: body = $msg->body¥n";
          }
       }
    else
       echo "PeekAll failed ($conn->errno) $conn->error";
    }

?>
```

参考

- SAMConnection->peek()

SAMConnection->receive()

(No version information available, might be only in CVS)

SAMConnection->receive() — メッセージをキューあるいは購読から取得する

説明

```
SAMConnection
SAMMessage receive ( string $target [, array $properties ] )
```

パラメータ

target

メッセージを取得するキュー、トピックあるいは購読の識別子。

properties

オプションの連想配列で、受信操作を制御するパラメータを指定します。

プロパティ名	とりうる値
SAM_CORRELID	メッセージの相関 ID 文字列にもとづいて、 受信するメッセージを選択するために使用します。
SAM_MESSAGEID	メッセージのメッセージ ID 文字列にもとづいて、 受信するメッセージを選択するために使用します。
SAM_WAIT	タイムアウトをミリ秒で指定します。この時間までメッセージの受信を待ち続け、 キューやトピックにメッセージがなかった場合に失敗とします。デフォルト値は 0 で、永遠に待ち続けます。この使用には注意が必要です。 メッセージが存在しない場合、PHP スクリプト自体が期限切れになるまで実行し続けます。

返り値

このメソッドは SAMMessage オブジェクトを返します。 エラーが発生した場合は **FALSE** を返します。

例

Example#1 キューからのメッセージの取得

```
<?php
$msg = $conn->receive('queue://receive/test');

if (!$msg) {
  // 受信に失敗しました!
  echo "Receive failed ($conn->errno) $conn->error";
}
?>
```

Example#2 オプションを指定して、キューからのメッセージの取得

この例では SAM_CORRELID オプションを使用して、 受信するメッセージを表すための相関 ID を指定します。 また、タイムアウトを 10 秒に指定します。

```
<?php

$msg = $conn->receive('queue://receive/test', array(SAM_CORRELID => $token, SAM_WAIT => 10000));

?>
```

Example#3 購読からのメッセージの取得

この例では、購読 ID からメッセージを受信する方法を示します。

```
<?php
$msg = $conn->receive($subscriptionName);

if (!$msg) {
  // 受信に失敗しました!
  echo "Receive failed ($conn->errno) $conn->error";
}
?>
```

$subscriptionName は、事前のコールで取得した購読 ID であることに注意しましょう。

参考

- SAMConnection->send()

SAMConnection->remove()

(No version information available, might be only in CVS)

SAMConnection->remove() — メッセージをキューから削除する

説明

SAMConnection
SAMMessage **remove** (string $target [, array $properties])

メッセージをキューから削除します。

パラメータ

target

メッセージを削除するキューの識別子。

properties

オプションの連想配列で、削除操作を制御するパラメータを指定します。

プロパティ名	とりうる値
SAM_CORRELID	メッセージの、対象相関 ID 文字列。 典型的な例は、"send" リクエストが返す相関 ID 文字列となります。
SAM_MESSAGEID	削除されるメッセージのメッセージ ID 文字列。

返り値

このメソッドは、エラーが発生した場合に **FALSE** を返します。

例

Example#1 メッセージ ID を指定して、メッセージをキューから削除する

```
<?php
if (!$conn->remove('queue://receive/test', array(SAM_MESSAGEID => $messageId))) {
  // 削除に失敗しました!
  echo "Remove failed ($conn->errno) $conn->error";
}
?>
```

SAMConnection->rollback()

(No version information available, might be only in CVS)

SAMConnection->rollback() — 現在作業中の内容をキャンセル (ロールバック) する

説明

SAMConnection
bool **rollback** (void)

現在作業中の内容をロールバックします。

返り値

このメソッドは、エラーが発生した場合に **FALSE** を返します。

例

Example#1 作業中の処理のキャンセル

```
<?php
if (!$conn->rollback()) {
  // ロールバックに失敗しました!
  echo "Rollback failed ($conn->errno) $conn->error";
}
?>
```

参考

- SAMConnection->commit()

SAMConnection->send()

(No version information available, might be only in CVS)

SAMConnection->send() — メッセージをキューに送信、あるいは項目をトピックに投稿する

説明

SAMConnection
string **send** (string $target , SAMMessage $msg [, array $properties])

"send" メソッドを使用して、メッセージを指定したキューに送信したり 指定したトピックに投稿したりします。このメソッドは、 メッセージに返信したり応答したりする際に使用する ID を返します。

パラメータ

target

メッセージを送信する場合はキューの ID (queue://queuename)、 トピックを投稿する場合はトピックの ID (topic://topicname) で、メッセージの配送先を指定します。

msg

送信あるいは投稿するメッセージ。

properties

オプションの連想配列で、受信時の動作を制御するプロパティを指定します。

プロパティ名	とりうる値
SAM_DELIVERYMODE	メッセージングサーバが配送を保障するかどうか、 システム障害時にメッセージが失われることを許可するかどうかを指定します。 SAM_PERSISTENT の場合はメッセージを失うことを許可しません。 SAM_NON_PERSISTENT の場合はメッセージを失うことを許可します。 送信結果の挙動は、PHP スクリプトが接続しているメッセージングサーバの機能によって変わります。 メッセージの永続化をサポートしていないサーバで SAM_PERSISTENT を指定した場合は、その機能が存在しないというエラーを発生させて処理が失敗します。
SAM_PRIORITY	0 から 9 までの数値で、メッセージの配送の優先度を指定します。 0 は優先度が最低であること、9 は優先度が最高であることを表します。 優先度を指定しなかった場合はデフォルト値が割り当てられます。 デフォルト値は、使用するメッセージングサーバによって異なります。
SAM_CORRELID	このメッセージの相関 ID として割り当てる文字列。 指定しなかった場合は、メッセージングサーバが値を自動的に割り当てます。
SAM_TIMETOLIVE	メッセージングサーバがキューにメッセージを残し続ける期間をミリ秒で指定します。 デフォルト値は 0 で、これはメッセージをずっと残し続けることを意味します。
SAM_WMQ_TARGET_CLIENT	このプロパティは WebSphere MQ を使用する場合にのみ有効で、 RFH2 ヘッダをメッセージに含めるかどうかを指定します。 設定できる値は 'jms' あるいは 'mq' のいずれかです。 デフォルトは 'jms' で、これは RFH2 ヘッダを含めることを意味します。 'mq' を指定すると、メッセージに RFH2 が含まれないようになります。

返り値

ID を文字列で返します。返信や応答を取得する際にこの文字列を使用します。 エラーが発生した場合には **FALSE** を返します。

> **注意**: ID が返されるのは、メッセージが送信先キュー (queue://xxxx) に正常に送信できた場合のみで、キュー上のメッセージの ID となります。 トピックにメッセージを投稿するために send を使用した場合の返り値は **TRUE** となり、ID は返されません。

例

Example#1 キューへのメッセージの送信

```
<?php
$msg = new SAMMessage('This is a simple text message');
$correlId = $conn->send('queue://send/test', $msg);
if (!$correlId) {
```

```
    // 送信に失敗しました!
    echo "Send failed ($conn->errno) $conn->error";
}

?>
```

Example#2 トピックへのメッセージの投稿

```
<?php
$msg = new SAMMessage('This is a simple text item');
if (!$conn->send('topic://test', $msg)) {
    // 送信に失敗しました!
    echo "Send failed ($conn->errno) $conn->error";
}
?>
```

Example#3 リクエストの送信と応答の受信

```
<?php
$msg = new SAMMessage('This is a simple text message');
$msg->header->SAM_REPLY_TO = 'queue://receive/test';
$correlid = $conn->send('queue://send/test', $msg);

if (!$correlid) {
    // 送信に失敗しました!
    echo "Send failed ($conn->errno) $conn->error";
} else {
    $resp = $conn->receive('queue://receive/test', array(SAM_CORRELID => $correlid));
}
?>
```

参考

- SAMConnection->receive()

SAMConnection::setDebug()

(No version information available, might be only in CVS)

SAMConnection::setDebug() — 追加のデバッグ出力を有効あるいは無効にする

説明

"setdebug" メソッドを使用して、追加のデバッグ出力を有効あるいは無効にします。 SAM フレームワークは、メソッド/関数 エントリや終了トレースデータなどの追加情報を提供します。 プロトコル固有の実装についても追加出力として提供します。

SAMConnection
void **send** (bool $switch)

パラメータ

switch

このパラメータを **TRUE** にすると、追加のデバッグ出力が行われます。 値を **FALSE** にすると、追加情報の出力を停止します。

例

Example#1 デバッグ出力を有効にする

```
<?php
$conn->setdebug(TRUE);
?>
```

Example#2 デバッグ出力を無効にする

```
<?php
$conn->setdebug(FALSE);
?>
```

SAMConnection->subscribe()

(No version information available, might be only in CVS)

SAMConnection->subscribe() — 指定したトピックの購読を作成する

説明

SAMConnection
string **subscribe** (string $targetTopic)

"subscribe" メソッドを使用して、指定したトピックを購読します。

パラメータ

targetTopic

購読するトピックの識別子 (topic://topicname)。

返り値

購読の識別子を返します。これを、それ以降にデータを取得する際に トピックのデータを取得するためのセレクタとして使用します。 エラーが発生した場合には **FALSE** を返します。 この識別子を使用して、特定のトピック名の受信コールを行います。

例

Example#1 トピックの購読

```
<?php
$subid = $conn->subscribe('topic://A');
if (!$subid) {
  // 購読に失敗しました!
  echo "Subscribe failed ($conn->errno) $conn->error";
}
?>
```

参考

- SAMConnection->unsubscribe()

SAMConnection->unsubscribe()

(No version information available, might be only in CVS)

SAMConnection->unsubscribe() — 指定したトピックの購読を解除する

説明

SAMConnection
bool **unsubscribe** (string $subscriptionId [, string $targetTopic])

"unsubscribe" メソッドを使用して、指定したトピックの購読を解除します。

パラメータ

subscriptionId

subscribe メソッドによって返された、既存の購読の識別子。

返り値

このメソッドは、エラーが発生した場合に **FALSE** を返します。

例

Example#1 購読の削除

```
<?php
if (!$conn->unsubscribe($subid)) {
    // 購読解除に失敗しました!
    echo "Unsubscribe failed ($conn->errno) $conn->error";
}
?>
```

参考

- SAMConnection->subscribe()

SAMMessage->body

(No version information available, might be only in CVS)

SAMMessage->body — メッセージの本文

説明

SAMMessage
string$nody;

"body" プロパティには、メッセージの本文が含まれます。 常に設定されているとは限りません。

例

Example#1 メッセージ本文へのテキスト文字列の設定

```
<?php
$msg = new SAMMessage();
$msg->body = 'This is a simple message';

?>
```

参考

- SAMMessage->header

SAMMessage->__construct()

(No version information available, might be only in CVS)

SAMMessage->__construct() — 新しいメッセージオブジェクトを作成する

説明

SAMMessage
__construct ([mixed $body])

新しい SAMMessage オブジェクトを作成します。 オプションでメッセージ本文を指定します。

パラメータ

body

オプションで指定するメッセージ本文。

例

Example#1 メッセージの作成

```
<?php

$msg = new SAMMessage();

?>
```

Example#2 単純なテキストを含むメッセージの作成

```
<?php

$msg = new SAMMessage('This is a simple text message');

?>
```

SAMMessage->header

(No version information available, might be only in CVS)

SAMMessage->header — メッセージのヘッダプロパティ

説明

SAMMessage
object$header;

header プロパティは、メッセージと関連付けられるシステムプロパティや ユーザプロパティのコンテナとなります。

プロパティは、 メッセージの送信者によってメッセージングシステムでの動作を制御するために指定されたり、 あるいはメッセージングシステム自身によって受信者に追加情報・ メッセージの処理方法を知らせるために指定されたりします。

SAM が理解できるいくつかのプロパティについては、 対応する定数が定義されています。 しかし、SAM の実装は大半のプロパティを無視し、そのままメッセージングシステムに渡します。 これにより、アプリケーションがメッセージング固有のプロパティを使用したり、 独自の "ユーザ" プロパティを定義することができるようになります。

SAM で定義済みのプロパティは次のようになります。

プロパティ名	とりうる値
SAM_MESSAGEID	メッセージを受信する場合は、このフィールドにはメッセージを識別する一意な ID が含まれます。これは、メッセージングシステムによって自動的に割り当てられるものです。 メッセージを送信する場合は、このフィールドは無視されます。
SAM_REPLY_TO	このメッセージに対する返信を投稿するキューの識別子を指定します。
SAM_TYPE	送信するメッセージの型を指定します。 SAM_TEXT はメッセージ本文の内容がテキスト文字列であることを示し、SAM_BYTES はメッセージ本文の内容が 何らかのアプリケーションで定義されているフォーマットであることを意味します。 このプロパティの使用法は、メッセージングサーバに依存します。 たとえば、JMS (Java Message Service) の仕様をサポートしているメッセージングサーバは、 このプロパティを解釈して "jms_text" 型および "jms_bytes" 型のメッセージを送信します。さらに、SAM_TYPE プロパティが SAM_TEXT に設定された場合は、 メッセージ本文のデータが UTF8 エンコードされた文字列であるものと期待されます。

プロパティの値を設定する際には、 その内容をメッセージングシステムにどのように配送するかのヒントを指定すると便利なことがあります。 デフォルトではプロパティの値はテキストとして扱われます。この場合は、 以下のシンプルな構文で値を設定します。

Example#1 デフォルトの構文による、テキスト形式のプロパティの設定

```
<?php
$msg = new SAMMessage();

$msg->header->myPropertyName = 'textData';
?>
```

型情報を渡したい場合は、別の構文を使用します。 この場合は、値と型ヒントを連想配列で渡します。

Example#2 型ヒントを使用することによる、テキスト形式のプロパティの設定

```
<?php
$msg = new SAMMessage();

$msg->header->myPropertyName = array('textData', SAM_STRING);
?>
```

型ヒントを渡す際には、次の表にある SAM の定義済み定数のいずれかを使用します。

定数	型の説明
SAM_BOOLEAN	渡された値は、true あるいは false の論理値として解釈されます。 その値が PHP の boolean 値として解釈できない場合は、 メッセージングシステムに渡される値は未定義となります。
SAM_BYTE	符号つき 8 ビット整数値です。SAM は、このプロパティの値を 1 バイトの値に変換してメッセージングシステムに渡します。 文字列が渡されると、それを数値として解釈しようとします。 数値として解釈した結果が符号つき 8 ビットの範囲に収まらない場合は、 変換の際にデータの損失が発生します。
SAM_DOUBLE	長浮動小数点数値です。SAM は、このプロパティの値を 15 桁の浮動小数点値に変換します。 文字列が渡されると、それを数値として解釈しようとします。 数値として解釈した結果が 15 桁の浮動小数点値の範囲に収まらない場合は、 変換の際にデータの損失が発生します。
SAM_FLOAT	浮動小数点数値です。SAM は、このプロパティの値を 7 桁の浮動小数点値に変換します。 文字列が渡されると、それを数値として解釈しようとします。 数値として解釈した結果が 7 桁の浮動小数点値の範囲に収まらない場合は、 変換の際にデータの損失が発生します。
SAM_INT	符号つき 32 ビット整数値です。SAM は、このプロパティの値を 32 ビットの値に変換してメッセージングシステムに渡します。 文字列が渡されると、それを数値として解釈しようとします。 数値として解釈した結果が符号つき 32 ビットの範囲に収まらない場合は、 変換の際にデータの損失が発生します。
SAM_LONG	符号つき 64 ビット整数値です。SAM は、このプロパティの値を 64 ビットの値に変換してメッセージングシステムに渡します。 文字列が渡されると、それを数値として解釈しようとします。 数値として解釈した結果が符号つき 64 ビットの範囲に収まらない場合は、 変換の際にデータの損失が発生します。
SAM_STRING	SAM は、このプロパティの値を文字列に変換してメッセージングシステムに渡します。

例

Example#3 プロパティに、メッセージの送信者を設定する

```
<?php
$msg = new SAMMessage('This is a test message');

// SAM 固有のプロパティを設定します...
$msg->header->SAM_REPLY_TO = 'queue://test/replyQueue';

// 任意のプロパティを定義します...
//
// デフォルトの文字列形式のプロパティ
$msg->header->myStringProp1 = 'a string property';
// 型ヒントつきの文字列プロパティ
$msg->header->myStringProp2 = array('another string property', SAM_STRING);

// boolean のプロパティ
$msg->header->myBoolProp = array(FALSE, SAM_BOOL);

// 数値形式のプロパティ
$msg->header->myIntProp = array(32768, SAM_INT);
$msg->header->myLongProp = array(9876543, SAM_LONG);
$msg->header->myByteProp1 = array(123, SAM_BYTE);
$msg->header->myByteProp2 = array('12', SAM_BYTE);
$msg->header->myFloatProp = array(3.141592, SAM_FLOAT);
$msg->header->myDoubleProp = array(3.14159265358979, SAM_DOUBLE);
?>
```

Example#4 メッセージからのプロパティの値の取得

```
<?php

// アプリケーション固有のプロパティにアクセスします
$intProp = $msg->header->'MyIntProp';

// メッセージングシステム固有のプロパティにアクセスします
$encoding = $msg->header->'JMS_IBM_Msgtype';

?>
```

参考

- *SAMMessage->body*

目次

- SAMConnection::disconnect() — メッセージングサーバからの接続を解除する
- SAMConnection->errno — 最後に実行した SAM 操作のエラーコードを表す数値を含む
- SAMConnection->error — 最後に実行した SAM 操作のエラーの説明を含む
- SAMConnection->isConnected() — メッセージングサーバとの接続が確立されているかどうかを調べる
- SAMConnection->peek() — メッセージをキューから読み込み、それをキューに残したままにする
- SAMConnection->peekAll() — ひとつあるいは複数のメッセージをキューから読み込み、それをキューに残したままにする
- SAMConnection->receive() — メッセージをキューあるいは購読から取得する
- SAMConnection->remove() — メッセージをキューから削除する
- SAMConnection->rollback() — 現在作業中の内容をキャンセル（ロールバック）する
- SAMConnection->send() — メッセージをキューに送信、あるいは項目をトピックに投稿する
- SAMConnection::setDebug() — 追加のデバッグ出力を有効あるいは無効にする
- SAMConnection->subscribe() — 指定したトピックの購読を作成する
- SAMConnection->unsubscribe() — 指定したトピックの購読を解除する
- SAMMessage->body — メッセージの本文
- SAMMessage->__construct() — 新しいメッセージオブジェクトを作成する
- SAMMessage->header — メッセージのヘッダプロパティ

Satellite CORBA クライアント拡張 [推奨されません]

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

注意: この拡張モジュールは Windows 環境では利用できません。

Satellite は非推奨です。Universe (» http://universe-phpext.sourceforge.net/) を使用し、Satellite は使用しないことを推奨します。

Satellite 拡張は、CORBA オブジェクトにアクセスするために使用されます。 php.ini の idl_directory= エントリに、使用する全ての IDL ファイルを保存する場所のパスを設定する必要があります。

Satellite のインストールについての詳細は、Satellite の README ファイルを参照ください。

OrbitEnum

(No version information available, might be only in CVS)

OrbitEnum — CORBA enums を使用する

説明

new **OrbitEnum** (string $id)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このクラスは、パラメータ id で指定した連番を 表します。

パラメータ

id

連番の名前（例: "MyEnum"）あるいは 完全なレポジトリID（例:"IDL:MyEnum:1.0"）のどちらかです。

例

Example#1 IDL ファイルの例

```
enum MyEnum {
     a,b,c,d,e
};
```

Example#2 MyEnum にアクセスするための PHP コード

```
<?php
$enum = new OrbitEnum ("MyEnum");

echo $enum->a;    /* 0 を出力 */
echo $enum->c;    /* 2 を出力 */
echo $enum->e;    /* 4 を出力 */
?>
```

OrbitObject

(No version information available, might be only in CVS)

OrbitObject — CORBA オブジェクトにアクセスする

説明

new **OrbitObject** (string $ior)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このクラスは CORBA オブジェクトへのアクセス機能を提供します。

パラメータ

ior

リモートオブジェクトを指す IOR (Interoperable Object Reference) を文字列で指定します。

例

Example#1 IDL ファイルの例

```
interface MyInterface {
      void SetInfo (string info);
      string GetInfo();

      attribute int value;
}
```

Example#2 MyInterface にアクセスするための PHP コード

```
<?php
$obj = new OrbitObject ($ior);

$obj->SetInfo ("A 2GooD object");

echo $obj->GetInfo();

$obj->value = 42;

echo $obj->value;
?>
```

OrbitStruct

(No version information available, might be only in CVS)

OrbitStruct — CORBA 構造体を使用する

説明

new **OrbitStruct** (string $id)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このクラスは、パラメータ id で指定した 構造体を表します。

パラメータ

id

構造体の名前（例: "MyStruct"）または完全なレポジトリ ID （例: "IDL:MyStruct:1.0"）のどちらかです。

例

Example#1 IDL ファイルの例

```
struct MyStruct {
      short shortvalue;
      string stringvalue;
};

interface SomeInterface {
   void SetValues (MyStruct values);
   MyStruct GetValues();
}
```

Example#2 MyStruct にアクセスする PHP コード

```
<?php
$obj = new OrbitObject ($ior);

$initial_values = new OrbitStruct ("IDL:MyStruct:1.0");
$initial_values->shortvalue = 42;
$initial_values->stringvalue = "HGTTG";

$obj->SetValues ($initial_values);
```

```
$values = $obj->GetValues();

echo $values->shortvalue;
echo $values->stringvalue;
?>
```

---

satellite_caught_exception

(PHP 4 >= 4.0.3)

satellite_caught_exception — 例外が前の関数からキャッチされたかどうかを調べる

説明

bool **satellite_caught_exception** (void)

この関数は、例外がキャッチされた場合に**TRUE**を返します。

Example#1 IDLファイルのサンプル

```
/* ++?????++ Out of Cheese Error. Redo From Start. */
exception OutOfCheeseError {
     int parameter;
}

interface AnotherInterface {
      void AskWhy() raises (OutOfCheeseError);
}
```

Example#2 CORBA例外を処理するPHPコード

```
<?php
$obj = new OrbitObject ($ior);

$obj->AskWhy();

if (satellite_caught_exception()) {
    if ("IDL:OutOfCheeseError:1.0" == satellite_exception_id()) {
        $exception = satellite_exception_value();
        echo $exception->parameter;
    }
}
?>
```

---

satellite_exception_id

(PHP 4 >= 4.0.3)

satellite_exception_id — 直近の例外に関するレポジトリIDを取得する

説明

string **satellite_exception_id** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

レポジトリID文字列を返します。(例:"IDL:MyException:1.0")例として は、satellite_caught_exception()の使用例を参照 ください。

---

satellite_exception_value

(PHP 4 >= 4.0.3)

satellite_exception_value — 直近の例外に関する例外構造体を得る

説明

OrbitStruct **satellite_exception_value** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例外構造体を返します。例については、 satellite_caught_exception()の使用例を参照下さ い。

---

satellite_get_repository_id

(PHP 4 >= 4.0.3)

satellite_get_repository_id — 未実装

説明

int ***satellite_get_repository_id*** *(object $obj)*

実装されていません。

---

satellite_load_idl

(PHP 4 >= 4.0.3)

satellite_load_idl — タイプマネージャに IDL ファイルのロードを指示する

説明

bool ***satellite_load_idl*** *(string $file)*

IDL ファイルを Satellite にロードします。

---

satellite_object_to_string

(PHP 4 >= 4.0.7)

satellite_object_to_string — オブジェクトを文字列表現に変換する

説明

string ***satellite_object_to_string*** *(object $obj)*

オブジェクトを、その文字列表現（IOR）に変換します。

目次

SCA 関数

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

SCA を使用すると、再利用可能なコンポーネントを PHP プログラマが書けるようになります。 このコンポーネントはさまざまな方法でコール可能ですし、 固有のインターフェイスを持っており、まったく手がかかりません。 現時点ではローカルあるいはウェブサービス経由でしか このコンポーネントをコールできませんが、将来はそれ以外の方法も可能になるでしょう。 これらの作業を、PHP プログラマが違和感なく行えるするようにする手段を提供しています。

SCA のコンポーネントは、phpDocumentor 形式 (http://www.phpdoc.org/ を参照ください) のアノテーション（注記）を用いて 他の SCA コンポーネントやウェブサービスとの依存性を宣言します。 SCA for PHP のランタイムは、これらの依存性をコンポーネントに代わって 実行時に解決します。したがって、プログラマは 依存性を解決するために参照を取得して配置したりすることよりも ビジネスロジックの作成に集中することができます。

SCA for PHP のプログラミングモデルは、さまざまな形式のサービス、 たとえば REST や Atompub などをサポートするように拡張することもできます。 しかし、現在指定できるのはウェブサービス（正確には WSDL で定義された SOAP/HTTP サービス）のみです。

これらのコンポーネントでは、 サービスが公開するインターフェイスを定義するアノテーションも使用します。 SCA for PHP のランタイムは、これらのアノテーションをもとにして WSDL を自動的に生成します。これを用いることで、SCA コンポーネントで簡単にウェブサービスを公開できるようになります。 これらのアノテーションは、phpDocumentor の記法を、 違和感なく拡張したものです。 ウェブサービスを公開する方法は簡単で、単に PHP コンポーネントをウェブサーバのドキュメントルートに置くだけです。

これらのコンポーネントではまた、 データ構造（XML スキーマの複合型を使用して表します）を指定するアノテーションも使用します。 このデータ構造を、Service Data Object (SDO) を用いて処理します。

SCA コンポーネントではなくアノテーションを含まない PHP スクリプトから、SCA コンポーネントのサービスを使用することができます。 また、PHP スクリプトやコンポーネントは、SCA コンポーネントでない ウェブサービスをコールすることもできます。しかし、 参照を取得するには、SCA コンポーネントと同じ形式のコール方法やアノテーションを使用します。

*まず手始めに、ConvertedStockQuote という SCA コンポーネントをご覧に入れましょう。 これを用いて、SCA for PHP の多くの機能について説明します。 このコンポーネントはひとつのメソッド **getQuote()** を持っています。このメソッドは株式の "ティッカー" を指定して その株の株価を取得し、指定した通貨に変換します。 このドキュメントの残りの部分では、この例をもとにして SCA for PHP の機能を説明していきます。*

Example#1 SCA コンポーネントのサンプル

```php
<?php

include "SCA/SCA.php";

/**
 * 指定したティッカーシンボルと通貨についての株価を計算する
 *
 * @service
 * @binding.soap
 */
class ConvertedStockQuote {

    /**
     * 使用する為替レートサービス
     *
     * @reference
     * @binding.php ../ExchangeRate/ExchangeRate.php
     */
    public $exchange_rate;

    /**
     * 使用する株価サービス
     *
     * @reference
     * @binding.soap ../StockQuote/StockQuote.wsdl
     */
    public $stock_quote;

    /**
     * ティッカーシンボルと通貨を指定して、株価を取得する
     *
     * @param string $ticker ティッカーシンボル。
     * @param string $currency 値を変換する通貨単位。
     * @return float 指定した通貨単位で表した株価。
     */
    function getQuote($ticker, $currency)
    {
        $quote  = $this->stock_quote->getQuote($ticker);
        $rate   = $this->exchange_rate->getRate($currency);
        return $rate * $quote;
    }
}
?>
```

この例では、PHP のクラスを含むスクリプトで SCA コンポーネントを実装しており、SCA.php をインクルードしています。このクラスは、ビジネスロジックと 他のコンポーネントやサービスへの参照を両方含んでいます。 **getQuote()** メソッドにはビジネスロジックしかありませんが、 このメソッドではインスタンス変数 $stock_quote および $exchange_rate を使用しています。 これらはそれぞれ別のコンポーネントを参照しており、SCA のランタイムがこれら二つのサービスのプロキシを用いて初期化します。 これは、このコンポーネントが実行されるたびに行われます。 これらの二つのサービスのアノテーションからわかるように、 一方はローカルのコンポーネントで、同じ PHP ランタイムからコールします。もう一方はリモートのコンポーネントで、SOAP リクエストでコールします。 このコンポーネントは **getQuote()** メソッドをローカルおよびウェブサービスの両方で公開しているので、 ローカルからもリモートからもコール可能です。

要件

SCA を使用してウェブサービスの作成や使用を行うには PHP 5.2.0 以降が必要です。また soap 拡張モジュールが有効でなければなりません。ローカルのコンポーネントのみを使用し、ウェブサービスを使用しないのなら、 このバージョンの SCA for PHP は PHP 5.1.6 以降で動作します。

インストール手順

SCA は、SDO とともにひとつのパッケージとして PECL で公開されています。 SCA_SDO パッケージを PECL からインストールする方法については http://www.php.net/sdo#sdo.installation を参照ください。 SCA のコードは、PHP のインクルードパスに存在しなければなりません。 つまり、もし /usr/local/lib/php/SCA にインストールされたのなら、 include_path ディレクティブに /usr/local/lib/php を含める必要があります。

例

これ以降の節の例では、PHP for SCA の次のような側面を説明します。

- PHP のアノテーションを使用して PHP のクラスを SCA コンポーネントとして定義する方法、 そしてアノテーションを使用してサービスを定義する方法。
- SCA コンポーネントをウェブサービスとして公開する方法。
- SCA コンポーネントから別のウェブサービスを使用する方法。 使用するウェブサービスが別の SCA コンポーネントである場合と まったく SCA とは関係ないサービスである場合の両方を示します。
- SCA コンポーネントから、ローカル（同一プロセス、同一コールスタック）の別の SCA コンポーネントをコールする方法。
- SCA コンポーネントではないクライアントスクリプトが、 getService をコールして SCA コンポーネントのプロキシを使用する方法。
- 住所や発注データなどのデータ構造を Service Data Object で表現し、処理する方法。
- SCA コンポーネントを配置する方法。特に、サービスの WSDL がいつどのように作成されるか。
- コンポーネント間でのパラメータの受け渡しが、ローカルコールであっても（参照ではなく）値渡しとなること。これにより、 コンポーネントの場所にかかわらずコール方法が同じになります。
- サービスへの、位置指定によるパラメータをサポートする方法。 もとになる WSDL がドキュメントリテラルをラップしたもので、 通常は名前指定によるパラメータしかサポートしていない場合も含みます。
- ビジネスロジックのエラーや実行時例外の処理方法

サービスコンポーネントの構造

サービスコンポーネントは、クラスとして実装します。 サービスコンポーネントであることを示すために、@service アノテーションを使用します。 SCA ランタイムは、 スクリプトのファイル名にもとづく規約でコンポーネント名を決定します。 したがって、クラス名とスクリプトのファイル名は同じにしておく必要があります。

PHP SCA コンポーネントは常にサービスを公開しています。 コンポーネントは、ウェブサービスへのリクエストの結果としてコールされるか スクリプト内の別のコンポーネントから直接コールされるかのいずれかの方法でしか起動しません。 このため、有効な PHP SCA コンポーネントは常に @service アノテーションを含んでおり、少なくともひとつの public メソッドを持っています。

各 SCA コンポーネントは、SCA.php をインクルードする必要があります。 ここには SCA クラスの定義が含まれているだけでなく、 スクリプトがコールされた際に実行される、 コンポーネントとして振舞うための PHP コードも含まれています。

警告

もし別のファイルもインクルードするのなら、それは SCA.php をインクルードする前に行わなければなりません。 SCA.php の後で別のファイルをインクルードすると、 SCA ランタイムがクラスを処理する際に、 そのファイルの内容が処理されません。

以下の例で、これらすべての構造について説明します。

Example#2 SCA for PHP のコンポーネントの構造

```
<?php

// 何かをインクルードします

include "SCA/SCA.php";

/**
 * @service
 */

class ConvertedStockQuote {

      // インスタンス変数、ビジネスロジック、そして最低ひとつの public メソッドを含めます

}
?>
```

別のサービスコンポーネントへのプロキシの取得

ある SCA コンポーネントが、別の SCA コンポーネントのサービスをコールすることができます。 コンポーネントが提供するサービスは、すべて public メソッドでできています。 SCA for PHP では現在、あるコンポーネントから他のコンポーネントをコールする方法を 二通り提供しています。ローカルに (同一の PHP ランタイム、 同一のコールスタック上で) コールする方法と、リモートにコールする方法です。 リモートにコールする場合は、コールされるコンポーネントが ウェブサービスを公開していなければなりません。

あるコンポーネントが別のコンポーネントをコールするには、 呼び出し元のコンポーネント側に呼び出し先コンポーネントへのプロキシが必要です。 このプロキシは、通常は呼び出し元コンポーネントのインスタンス変数として用意します。 しかし、後でご覧いただくように SCA::getService() でプロキシを取得することも可能です。 コンポーネントを作成する際にプロキシを作成し、 別のコンポーネントを参照するインスタンス変数にそのプロキシを "注入 (inject)" します。プロキシは、そのコンポーネントがローカルにあるかリモートにあるかにかかわらず 常に用いられます。これにより、リモートだろうがローカルだろうが同じ方法でコールできるようになります (たとえば、ローカルへのコールの際もデータは常に値渡しとなります)。 プロキシは、そのコンポーネントがある場所と、 それをコールする方法を知っています。

サービスのプロキシを保持することを意図したインスタンス変数は、 PHPDocumentor 形式のふたつのアノテーション @reference および @binding で表します。それぞれのアノテーションは、 クラスのインスタンス変数のコメント部で指定します。以下のコードを参照ください。

インスタンス変数の前にある @reference アノテーションは、 そのインスタンス変数がコンポーネントへのプロキシとして初期化されることを示します。

@binding アノテーションには二通りの形式 @binding.php および @@binding.soap があり、それぞれそのプロキシが ローカルコンポーネント用なのかウェブサービス用なのかを表します。 @binding.php および @binding.soap のいずれの場合も、 対象の URI をアノテーションで指定します。

現時点では依存性の指定をアノテーションで行っているので、 参照先を変更する唯一の方法はコンポーネント内のアノテーションを書き換えることになります。

今回の例の ConvertedStockQuote では、インスタンス変数 $exchange_rate をローカルのコンポーネント ExchangeRate へのプロキシとして初期化しています。これは、 ConvertedStockQuote のインスタンスが作成されるたびに作成されます。

Example#3 ローカルの PHP クラスへのプロキシの取得

```
<?php
   /**
    * 使用する為替レートサービス
    *
     * @reference
       * @binding.php ../ExchangeRate/ExchangeRate.php
     */
    public $exchange_rate;
?>
```

@binding.php の場合の URI は、 そのコンポーネントの実装を含むスクリプトの場所を表します。 コンポーネントはローカルにコールされます。提供されるサービスは、コンポーネントの public メソッド群となります。 URI は、絶対パスあるいは相対パスで表す単純なパス名でなければなりません。 コンポーネントの読み込みには、PHP の include ディレクティブを使用します。 その際に、既に読み込まれていないかどうかを class_exists() で調べます。URI が相対パスの場合は、 そのアノテーションを含むコンポーネントの場所を基準としてパスを解決します。 これは、通常の PHP の振る舞いとは異なることに注意しましょう。 通常は、PHP は include_path を基準として解決します。 このようにしない理由は、複数コンポーネント間の参照関係を、 できるだけ場所の影響を受けないようにするためです。

この ExchangeRate サービスがリモートにあり、ウェブサービスとしてコールされるものだとすると、 変更するところは @binding の行だけです。PHP クラスの場所を指定するかわりに、 ウェブサービスについて記述した WSDL ファイルの場所を指定します。 今回のサンプルでは、二番目の参照でこの方式を説明しています。

Example#4 ウェブサービスへのプロキシの取得

```
<?php
/**
    * 使用する株価サービス
    *
     * @reference
       * @binding.soap ../StockQuote/StockQuote.wsdl
     */
```

```
    public $stock_quote;
?>
```

StockQuote コンポーネントはウェブサービスへのリクエストとしてコールされます。 今回の場合は WSDL の URI が単純なパス名となっていますが、 PHP のラッパーを使用して、たとえば file:// や http:// で始めることもできます。 単純にパス指定する場合は絶対パスでも相対パスでもかまいません。 相対パスの場合は、そのアノテーションを含むコンポーネントの位置からの相対パスと解釈されます。 これは @binding.php の振る舞いと同様で、通常の PHP の挙動とは異なることに注意しましょう。 通常は、PHP は現在の作業ディレクトリからの相対パスとして探します。 作業ディレクトリは、通常はそのスクリプトがコールされた場所となります。 このようにしない理由は、複数コンポーネント間の参照関係を、 まずバインド形式の違いに影響を受けないようにすること、 そして場所の影響を受けないようにするためです。

別のサービスコンポーネントのコール

ConvertedStockQuote の例では、これらのプロキシをコールすることで それぞれが参照するコンポーネントを使用しています。

Example#5 サービスのコール

```
<?php
$quote    = $this->stock_quote->getQuote($ticker);
$rate     = $this->exchange_rate->getRate($currency);
?>
```

StockQuote サービスのコールはローカルサービスへのコール、 一方 ExchangeRate サービスのコールはリモートサービスへのコールとなります。 それがローカルサービスであるかリモートサービスであるかにかかわらず、 コール方法は同じようになることに注意しましょう。

プロキシを使用することにより、コンポーネントのコール方法や振る舞いが ローカルであろうがリモートであろうが同じものになることが保証されます。 つまり、コンポーネントをコールする際に それがローカルにあるのかリモートにあるのかを気にしなくてもよいということです。 たとえば、ローカルサービスへのプロキシは、 引数をコピーしてそのコピーのみをサービスに渡します。 これにより、リモートコールの場合と同様に引数が値渡しとなることが保証されるわけです。 またリモートサービスへのプロキシは、受け取ったパラメータを SOAP リクエストのプロパティに変換し、戻ってきた内容を通常のパラメータリストに変換します。

上の例では、$ticker および $currency は明らかに PHP のスカラー型です。 コンポーネントには PHP のスカラー型である string、integer、float および boolean を渡すことができます。しかし、構造化されたデータは常に Service Data Objects (SDO) として渡します。 これ以降では、コンポーネントがどのようにして SDO を作成して ローカルコールやウェブサービスコールに渡すのか、 あるいはどのようにしてコンポーネントが返す SDO を作成するのかについて説明します。 PHP SDO プロジェクトのドキュメントに、SDO API の動作原理についての説明があります (SDO のページ を参照ください)。

SCA コンポーネントではないスクリプトからのサービスの指定およびコール

SCA コンポーネントは、インスタンス変数に @reference アノテーションを指定することで他のコンポーネントやサービスのプロキシを取得することができます。 しかし、コンポーネントではないスクリプトの場合はこれは不可能です。 クライアントスクリプトがコンポーネントではない場合は、静的メソッド **SCA::getService()** を使用して ローカルあるいはリモートサービスのプロキシを取得する必要があります。 **getService()** メソッドの引数には URI を指定します。 通常は、これはコンポーネントを含むローカル PHP スクリプトの場所か、 あるいは wsdl ファイルの場所となります。また、先ほどの節で説明した @binding アノテーションで指定するものと同じようになります。 つまり、相対 URI はクライアントスクリプトの場所を基準として解決し、 PHP の include_path や現在の作業ディレクトリは使用しません。

たとえば、あるスクリプトで ExchangeRate および StockQuote サービスのプロキシを取得する必要があるが、そのスクリプトがコンポーネントではない場合、 **getService()** メソッドを次のように使用します。

Example#6 getService を使用したプロキシの取得

```
<?php
$exchange_rate  = SCA::getService('../ExchangeRate/ExchangeRate.php');
$stock_quote    = SCA::getService('../StockQuote/StockQuote.wsdl');
?>
```

サービスのメソッドをコールするには、返されたプロキシを使用します。 これはコンポーネント内からコールする場合と同じです。

Example#7 プロキシ上でのコール

```
<?php
$quote    = $stock_quote->getQuote($ticker);
$rate     = $exchange_rate->getRate($currency);
?>
```

サービスコンポーネントの、ウェブサービスとしての公開

SCA for PHP は、サービスコンポーネント内のアノテーションから WSDL を作成することができます。これにより、 ウェブサービスとして公開することが簡単にできるようになります。 WSDL の作成に必要な情報を SCA に与えるには、@binding.soap アノテーションを @service アノテーションの下に追加しなければなりません。 そこに、メソッドのパラメータおよび返り値をそれぞれ @param アノテーションおよび @return アノテーションで指定します。 これらのアノテーションを読み込むことで WSDL が作成され、 パラメータの順序や型が WSDL の <schema> セクションの内容を決定します。

SCA for PHP は、常に document/literal でラップされたコンポーネントの WSDL を作成します。これにより、ウェブサービスを公開します。注意してほしいのは、 これは、SCA コンポーネント以外のウェブサービスや 別の形式の WSDL で表されるウェブサービスの使用を妨げるものではないということです。

@param アノテーションで使用できるスカラー型は、 一般的な PHP のスカラー型である boolean、integer、float そして string です。これらは、WSDL において それぞれ同名の XML スキーマ型に変換されます。 以下の例は、ConvertedStockQuote コンポーネントが呼び出す StockQuote サービスをごく平凡に実装したもので、 string および float 型の使用法をご確認いただけます。

Example#8 StockQuote サービス

```
<?php

include "SCA/SCA.php";

/**
 * リモートの StockQuote ウェブサービスの土台となる実装
 *
 * @service
 * @binding.soap
 *
 */
class StockQuote {

    /**
     * 指定したティッカーシンボルの株価を取得する
```

```
     *
      * @param string $ticker ティッカーシンボル
      * @return float 株価
      */
     function getQuote($ticker) {
         return 80.9;
   }
}
?>
```

このサービスの WSDL は、次のようになります (おそらく、サービスの場所は 'localhost' ではなく別のものになるでしょう)。

Example#9 作成された WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xsi:type="tDefinitions"
       xmlns:tns2="http://StockQuote" xmlns:tns="http://schemas.xmlsoap.org/wsdl/"
       xmlns:tns3="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" targetNamespace="http://StockQuote">
   <types>
       <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
         targetNamespace="http://StockQuote">
        <xs:element name="getQuote">
          <xs:complexType>
            <xs:sequence>
                <xs:element name="ticker" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="getQuoteResponse">
          <xs:complexType>
            <xs:sequence>
                <xs:element name="getQuoteReturn" type="xs:float"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:schema>
   </types>

   <message name="getQuoteRequest">
      <part name="getQuoteRequest" element="tns2:getQuote"/>
   </message>
   <message name="getQuoteResponse">
      <part name="return" element="tns2:getQuoteResponse"/>
   </message>
   <portType name="StockQuotePortType">
      <operation name="getQuote">
         <input message="tns2:getQuoteRequest"/>
         <output message="tns2:getQuoteResponse"/>
      </operation>
   </portType>
   <binding name="StockQuoteBinding" type="tns2:StockQuotePortType">
      <operation name="getQuote">
        <input>
            <tns3:body xsi:type="tBody" use="literal"/>
        </input>
        <output>
            <tns3:body xsi:type="tBody" use="literal"/>
        </output>
          <tns3:operation xsi:type="tOperation" soapAction=""/>
      </operation>
        <tns3:binding xsi:type="tBinding" transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
   </binding>
   <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns2:StockQuoteBinding">
          <tns3:address xsi:type="tAddress" location="http://localhost/StockQuote/StockQuote.php"/>
     </port>
   </service>
</definitions>

<!-- this line identifies this file as WSDL generated by SCA for PHP. Do not remove -->
```

SCA コンポーネントの配置

PHP SCA コンポーネントを配置するには、特別な方法は必要ありません。 コンポーネントの PHP スクリプトを、 ウェブサーバのドキュメントルート以下の適切なディレクトリに配置するだけです。 これはごく普通の PHP スクリプトと同様です。 スクリプトがコールされると、各コンポーネント内の **SCA::initComponent()** の行が実行されます。 またウェブサービスのコール、ローカルのコール、 あるいは WSDL へのリクエストに対してコンポーネントが適切に応答するようにします。

ウェブサービスとしてのサービスを提供する SCA コンポーネントの WSDL の取得

ウェブサービスのインターフェイスを公開している (つまり、@binding.soap アノテーションを持っている) SCA コンポーネントは、HTTP リクエストのパラメータに "wsdl" を指定されるとその WSDL 定義を返します。 つまり、WSDL を取得する一般的な方法は、URL の最後に "?wsdl" をつけることです。以下の例では、 file_get_contents() を使用してサービスの WSDL を取得し、それをテンポラリファイルに書き出します。 それをもとにして、サービスへのプロキシを取得します。 もちろん、WSDL をブラウザやその他の方法で取得して、自分でそれを保存することも可能です。

Example#10 作成された WSDL

```
<?php
$wsdl = file_get_contents('http://www.example.com/Services/Example.php?wsdl');
file_put_contents("service.wsdl",$wsdl); // wsdl をファイルに書き出します
$service = SCA::getService('service.wsdl');
?>
```

注意: wsdl が xsd をインポートしている場合は、 それを個別に取得する必要があります。

WSDL の作成方法について理解する

SCA for PHP は、@service アノテーションの後に @binding.soap アノテーションを含むコンポーネントについての WSDL を作成します。WSDL を作成するために、SCA はコンポーネントの情報を取得し、各 public メソッドの @param アノテーションおよび @return アノテーションの内容を調べます。 また、コンポーネント内の @types アノテーションの内容も調べます。 @param アノテーションおよび @return アノテーションからの情報を基にして、 WSDL の <types> セクションを構築します。また、@types アノテーションによる別のスキーマファイルの指定は、WSDL 内の <import> 要素に変換されます。

<service> 要素の location 属性

WSDL の最後は <service> 要素で、 この要素の location 属性によってサービスの URL を表します。 たとえば、次のようになります。

Example#11 location 属性

```
      <service name="ConvertedStockQuote"
...
location="http://localhost/ConvertedStockQuote/ConvertedStockQuote.php"/>
```

この location は、ウェブサーバのドキュメントルートからの相対位置であることに注意しましょう。 事前に解決することはできません。コンポーネントがウェブサーバ上の適切な場所に配置され、 ホスト名やポート番号がわかった時点ではじめて解決することができます。 WSDL を要求した URL からの情報を使用します。つまり、たとえば http://www.example.com:1111/ConvertedStockQuote/ConvertedStockQuote.php?wsdl へのリクエストの応答として WSDL が作成されたとすると、 WSDL の location 属性には http://www.example.com:1111/ConvertedStockQuote/ConvertedStockQuote.php が挿入されます。

WSDL および位置パラメータをラップするドキュメント/リテラル

SCA for PHP が作成する WSDL は、ドキュメント/リテラル でラップした形式となります。これは、パラメータを囲い込んで 'ラッパー' メソッド型として返すものです。この型は、 対応するメソッドの名前となります。WSDL の先頭にある <types> 要素で、これらのラッパーを定義します。ConvertedStockQuote のサンプルにおける **getQuote()** メソッドを考えてみましょう。

Example#12 ふたつの引数を持つメソッド

```
<?php
   /**
    * ティッカーシンボルと通貨を指定して、株価を取得する
    *
    * @param string $ticker ティッカーシンボル。
    * @param string $currency 値を変換する通貨単位。
    * @return float 指定した通貨単位で表した株価。
    */
   function getQuote($ticker, $currency)
   {
       $quote   = $this->stock_quote->getQuote($ticker);
       $rate    = $this->exchange_rate->getRate($currency);
       return  $rate * $quote;
   }
?>
```

このメソッドの WSDL は、メソッドとパラメータの名前を定義します。 また、パラメータに対応する XML スキーマ型を提供します。 WSDL の types セクションは、このようになります。

Example#13 パラメータ名を表す types セクション

```
  <types>
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
         targetNamespace="http://ConvertedStockQuote">
       <xs:element name="getQuote">
         <xs:complexType>
           <xs:sequence>
               <xs:element name="ticker" type="xs:string"/>
               <xs:element name="currency" type="xs:string"/>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
        <xs:element name="getQuoteResponse">
         <xs:complexType>
           <xs:sequence>
                <xs:element name="getQuoteReturn" type="xs:float"/>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
     </xs:schema>
  </types>
```

インターフェイス内で順番に指定したパラメータと soap リクエスト内の名前つきパラメータを含む XML を相互変換するために、 SCA ランタイムは特別な処理をします。この件について知るために、 さまざまなインターフェイスを使用する PHP スクリプトが SOAP コールようのパラメータリストをどのように作成するのかを考えてみましょう。 たとえば PHP の SoapClient を使用する PHP スクリプトは、 "ticker" および "currency" の値を指定したひとつのパラメータを SoapClient に渡す必要があります。おそらくこれは連想配列になるでしょう。 SCA コンポーネントがウェブサービス用にパラメータリストを作成する場合は、 相手がローカルであるかリモートであるかによって形式が異なります。 そのため、個別のアプローチが必要となります。

SCA が SCA コンポーネント用の WSDL を作成する際に、 その WSDL が SCA コンポーネントのインターフェイスであることを示すコメントを埋め込みます。 ある SCA コンポーネントが別のコンポーネントをウェブサービスとしてコールするとき、 呼び出し側の SCA ランタイムが受け取るパラメータは、 指定された順番に並べられたリストとなります。 これを、soap メッセージの名前つきパラメータにひとつひとつ割り当てるわけです。 たとえば、上で定義した **getQuote()** メソッドに値 'IBM' および 'USD' を渡す場合のコール方法は、このようになります。

```
$quote  =  $remote_service->getQuote('IBM','USD');
```

この結果は、次のような soap メッセージになります。

```
<getQuote>
   <ticker>IBM</ticker>
   <currency>USD</currency>
</getQuote>
```

サービス提供側の SCA コンポーネントは、soap メッセージから受け取った 名前つきパラメータをひとつひとつ受け取り、順番が指定されたパラメータリスト形式の ('IBM','USD') に再変換します。

警告

どちらの側の SCA ランタイムについても、 soap メッセージ内でのパラメータの登場順が メソッドのパラメータリストの順序と一致していることを前提としています。 これは、@param アノテーションの順序によって決まります。 これが、WSDL 内でのパラメータの登場順と soap メッセージ内でのパラメータの登場順を定義しています。 したがって、@param アノテーションの順序は メソッドのパラメータの順番と一致させておくことが必要不可欠です。

データ構造の扱い

SCA コンポーネントが受け渡しできる PHP のスカラー型は boolean、integer、float および string の四種類です。 それ以外のデータ構造を扱う際には、SCA コンポーネントは Service Data Objects (SDO) を使用します。SDO についての詳細は、このマニュアルの SDO のページ を参照ください。 SDO になじみのあるかたならご存知のとおり、 SDO は (XML のモデルのような) 構造化されたデータを扱うのに非常に適しています。 また、リモートコンポーネントやウェブサービスとのデータのやり取りの際に ごく自然にシリアライズすることができます。 データ構造の受け渡しを行う手段としては、現在 SDO のみをサポートしています。 その他の PHP オブジェクトや配列を受け渡しすることはできません。

SCA ランタイムは、データを常に値渡しします。これはローカルコールでも同じです。 そのため、SCA ランタイムは、パラメータリスト中の SDO のコピーを事前に作成します。これはスカラー型の場合と同じです。

データ構造を SCA コンポーネントで定義する方法

現在、データ構造の定義の場所を指定する手段としては XML スキーマファイルのでの型の指定しかありません。 しかし、将来的には別の方法も可能になるでしょう。 たとえば PHP のクラスやインターフェイスで定義したり、 連想配列で定義したりなどです。

SDO の使用法を説明するために、以下では新しいコンポーネント PortfolioMangement サービスを使用します。 これは、指定した顧客のポートフォリオを表す SDO を返します。

Example#14 データ構造を使用するコンポーネント

```
<?php

include "SCA/SCA.php";

/**
 * 顧客のポートフォリオを管理する
 *
 * @service
 * @binding.soap
 *
 * @types http://www.example.org/Portfolio PortfolioTypes.xsd
 *
 */
class PortfolioManagement {

    /**
     * 指定した顧客のポートフォリオを取得する
     *
      * @param integer $customer_id 顧客 ID
      * @return Portfolio http://www.example.org/Portfolio ポートフォリオ (銘柄と数量)
     */
    function getPortfolio($customer_id) {
        // データベースから取得しているものとします
        $portfolio = SCA::createDataObject('http://www.example.org/Portfolio', 'Portfolio');
        $holding = $portfolio->createDataObject('holding');
        $holding->ticker = 'AAPL';
        $holding->number = 100.5;
        $holding = $portfolio->createDataObject('holding');
        $holding->ticker = 'INTL';
        $holding->number = 100.5;
        $holding = $portfolio->createDataObject('holding');
        $holding->ticker = 'IBM';
        $holding->number = 100.5;
         return $portfolio;
    }

}
?>
```

@types アノテーションは次のようになります。

```
<?php
@types http://www.example.org/Portfolio PortfolioTypes.xsd
?>
```

これは、名前空間 http://www.example.org/Portfolio 内の型が、スキーマファイル URI PortfolioTypes.xsd で見つかることを表します。作成される WSDL では、この情報を次のような import ステートメントで再現します。

```
     <xs:import schemaLocation="PortfolioTypes.xsd"
                          namespace="http://www.example.org/Portfolio"/>
```

つまり、この URI は、絶対 URI であろうが相対 URI であろうが schemaLocation 属性で指定されたときに解決可能でなければなりません。

SDO の作成

SDO になじみの深い方は、SDO は 認められている構造 (あるいは 'スキーマ' や 'モデル' と呼ばれることもあります) でのみ作成されることをご存知でしょう。 また、'new' を使用するのではなくデータファクトリが必要となることもご存知でしょう。 たいていは、既存のデータオブジェクトをデータファクトリとして使用します。 しかし時には、特に最初のデータオブジェクトを取得するときなど、 何か別のものをデータファクトリとして使用する必要があります。

SCA では、SCA ランタイムクラスかあるいは (ローカルかリモートの) サービスのプロキシを SDO のデータファクトリとして使用できます。 どれを使用するのか、そしていつ使用するのかについて、 次の二つの節で説明します。

SDO の作成について説明するための新しい例を使用します。 サービスに SDO を渡し、そしてそのサービスから SDO を返すものです。

サービスに渡す SDO の作成

サービスをコールする際にデータ構造が必要となる場合、 そのサービスへのプロキシを 対応する SDO 用のデータファクトリとして使用します。 たとえば、ローカルの AddressBook コンポーネントが提供するサービスのプロキシとして使用するコンポーネントを考えて見ましょう。

```php
<?php
/**
  * @reference
  * @binding.local  AddressBook.php
  */
$address_book;
?>
```

コールしようとしている AddressBook コンポーネントの定義は、 次のようになります。

```php
<?php
/**
* @service
*  @binding.soap
*  @types  http://addressbook  ../AddressBook/AddressBook.xsd
*/
class AddressBook {

    /**
       * @param personType $person http://addressbook (person オブジェクト)
       * @return addressType http://addressbook (その person オブジェクトに対応する address オブジェクト)
      */
    function lookupAddress($person) {
        ...
    }
}
?>
```

AddressBook コンポーネントは、 **lookupAddress()** という名前のサービスメソッドを提供します。 これは、http://addressbook 名前空間の型を使用します。 lookupAddress メソッドは、personType 型のデータを受け取って addressType 型のデータを返します。それぞれの型は、スキーマファイル addressbook.xsd で定義されています。

AddressBook コンポーネントを使用するコンポーネントを作成すると、 インスタンス変数 $address_book にサービスへのプロキシが設定されます。 呼び出し元のコンポーネントは、$address_book に格納されたプロキシを使用して SDO を作成します。実際の例を以下に示します。

```php
<?php
$william_shakespeare          = $address_book->createDataObject('http://addressbook','personType');
$william_shakespeare  ->name = "William  Shakespeare";
$address                      = $address_book->lookupAddress($william_shakespeare);
?>
```

このようにしてプロキシから SDO を作成する方法は、 SCA コンポーネントに限った話ではないことに注意しましょう。 サービスが一般の PHP スクリプトからコールされ、 プロキシを **getService()** で取得した場合にも同様の方法を使用します。

```php
<?php
$address_book  = SCA::getService('AddressBook.php');
$william_shakespeare  = $address_book->createDataObject('http://addressbook','personType');
?>
```

コンポーネントから返す SDO の作成

データオブジェクトを作成して呼び出し元に返す必要のあるコンポーネントは、 データオブジェクトへのプロキシを使用しません。この場合は、SCA.php の静的メソッド **createDataObject()** を使用します。 したがって、もし先ほど説明した AddressBook コンポーネントが addressType 型のオブジェクトを名前空間 http://addressbook に作成するのなら、 次のようになります。

```php
<?php
$address  = SCA::createDataObject('http://addressbook','addressType');
?>
```

エラー処理

この節では、エラーの処理方法について説明します。 エラーには次の二種類があります。

- SCA の実行時例外。これは、 コンポーネントの実行やリモートサービスとのやりとりの際に問題が発生したことを表すものです。 ネットワークや設定の問題が原因で起こることがあります。
- ビジネスロジックの例外。これは、 プログラマが定義するものです。 PHP の Exception クラスを継承したクラスを作成し、 ビジネスロジックで問題が発生した場合に故意にスローします。

実行時例外の処理

SCA の実行時例外には二つの型があります。

- SCA_RuntimeException - SCA ランタイム内で問題が見つかった、あるいは問題が起こったであろうと考えられるということを通知します。これは、さまざまな理由で発生しえます。 その多くは、対象がローカルであるかリモートサービスであるかにかかわらず発生します。 コンポーネント内のアノテーションのエラーや、 WSDL や PHP ファイルが存在しないなどといった内容です。 ウェブサービスの場合に SCA_RuntimeException がスローされるのは、 リモートのウェブサービスから SoapFault を受け取り、 その失敗コードの内容が「再試行しても成功しないであろう」と判断されるときです。
- SCA_ServiceUnavailableException - これは SCA_RuntimeException のサブクラスです。 接続時やリモートサービスの使用時に問題が発生したものの、 再試行すれば成功する可能性がある場合にスローされます。 ウェブサービスの場合にこの例外がスローされるのは、 リモートのウェブサービスから SoapFault を受け取り、 その失敗コードの内容が「再試行すれば成功するであろう」と判断されるときです。

ビジネスロジックの例外の処理

ビジネスロジックの例外の定義やスローは、通常通りの方法でコンポーネントで行います。 そのコンポーネントがローカルにコールされるかリモートからコールされるかは関係ありません。 SCA ランタイムは、ローカルからコールされたコンポーネント内で発生した ビジネスロジック例外をキャッチしません。これは呼び出し元に返されます。 一方、コンポーネントがウェブサービス経由でコールされた場合は SCA ランタイムがそのビジネスロジック例外をキャッチし、再度スローします。 呼び出し元がその例外の定義を知っている（その例外の PHP クラス定義を含むファイルをインクルードしている）ものとして、再度スローされた冷害は元の例外と同じ内容を含みます。 つまり、たとえば **getLine()** メソッドや **getFile()** メソッドには ビジネスロジック内のどの位置で例外が発生したのかが含まれます。 例外は、soap fault の詳細フィールドにおいて "Client" というコードで渡されます。

定義済みクラス

SCA へのインターフェイスのほとんどは、SCA コンポーネントのアノテーションを用いて操作します。 そのため、パブリックなクラスやメソッドは少ししかありません。 スクリプトやコンポーネントからコールできるクラスは、 SCA クラスそのもののほかには プロキシクラスである SCA_LocalProxy と SCA_SoapProxy だけです。

SCA

SCA クラスのほとんどの作業は、SCA コンポーネントにファイル SCA.php をインクルードしたときに行われます。しかし、PHP スクリプトは SCA.php をインクルードしてから SCA クラスの **getService()** メソッドをコールし、 サービスのプロキシを取得することもできます。 コンポーネントは、これを行う必要はありません。というのも、 インスタンス変数と @reference アノテーションを定義することでプロキシを取得できるからです。

SDO を作成して呼び出し元に返す必要があるコンポーネントは、 データファクトリが必要となります。この目的のために、SCA クラスは **createDataObject()** メソッドをサポートしています。 これは、コンポーネントの @types アノテーションで定義したモデルに基づいた SDO を作成します。 **createDataObject()** への引数は、 SDO の XML データアクセスサービスと同じです。

メソッド

- getService - サービスのプロキシを取得する
- createDataObject - SDO を作成する

SCA_LocalProxy

getService() に 対象となるローカル PHP コンポーネントを指定してコールすると、 ローカルプロキシが返されます。 ローカルプロキシは、@reference および @binding.php アノテーションで定義されたコンポーネントのインスタンス変数に注入されます。 スクリプトあるいはコンポーネントがローカルプロキシをコールすると、 それが対象のコンポーネント自身に渡されます。

SDO を作成して呼び出し元に返す必要があるコンポーネントは、 データファクトリが必要となります。この目的のために、 SCA_LocalProxy クラスは createDataObject メソッドをサポートしています。 これは、コンポーネントの @types アノテーションで定義したモデルに基づいた SDO を作成します。 createDataObject への引数は、 SDO の XML データアクセスサービスと同じです。

メソッド

- createDataObject - SDO を作成する

SCA_SoapProxy

getService() に 対象となる WSDL ファイルを指定してコールすると、 SOAP プロキシが返されます。 SOAP プロキシは、@reference および @binding.soap アノテーションで定義されたコンポーネントのインスタンス変数に注入されます。 スクリプトあるいはコンポーネントが SOAP プロキシをコールすると、 それがウェブサービスへの SOAP リクエストに変換され、 PHP の Soap 拡張モジュールの助けを得て対象のコンポーネントに渡されます。

SDO を作成して呼び出し元に返す必要があるコンポーネントは、 データファクトリが必要となります。この目的のために、 SCA_SoapProxy クラスは createDataObject メソッドをサポートしています。 これは、対象の WSDL で定義したモデルに基づいた SDO を作成します。 createDataObject への引数は、 SDO の XML データアクセスサービスと同じです。

メソッド

- createDataObject - SDO を作成する

SCA_LocalProxy::createDataObject

(No version information available, might be only in CVS)

SCA_LocalProxy::createDataObject — SDO を作成する

説明

SDO_DataObject **SCA_LocalProxy::createDataObject** (string $type_namespace_uri , string $type_name)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドは、通常の PHP スクリプト内で使用するか、 SCA コンポーネントがローカルサービスとして使用する SDO を作成するために使用します。 パラメータとして、SDO の名前空間 URI と型名を指定します。 名前空間と型は、コールされるコンポーネントのインターフェイスで定義されている必要があります。 つまり、SCA_LocalProxy がプロキシとして働くコンポーネント内の @types アノテーションで指定されているスキーマファイルのいずれかで 名前空間と型が定義されていなければなりません。

パラメータ

type_namespace_uri

型の名前空間。

type_name

型の名前。

返り値

新しく作成した SDO_DataObject を返します。

エラー / 例外

SDO_TypeNotFoundException

namespaceURI および typeName が、 SCA_LocalProxy がプロキシとして働くコンポーネント内の @types アノテーションで指定されているスキーマファイルのいずれかで 定義されていない場合にスローされます。

SCA_SoapProxy::createDataObject

(No version information available, might be only in CVS)

SCA_SoapProxy::createDataObject — SDO を作成する

説明

SDO_DataObject **SCA_SoapProxy::createDataObject** (string $type_namespace_uri , string $type_name)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドは、通常の PHP スクリプト内で使用するか、 SCA コンポーネントがウェブサービスとして使用する SDO を作成するために使用します。 パラメータとして、SDO の名前空間 URI と型名を指定します。 名前空間と型は、ウェブサービスの WSDL で定義されている必要があります。ウェブサービス自身も SCA コンポーネントである場合は、 SCA_SoapProxy がプロキシとして働くコンポーネント内の @types アノテーションで指定されているスキーマファイルのいずれかで 名前空間と型が定義されていなければなりません。

パラメータ

type_namespace_uri

型の名前空間。

type_name

型の名前。

返り値

新しく作成した SDO_DataObject を返します。

エラー / 例外

SDO_TypeNotFoundException

namespaceURI および typeName が、 この SCA_SoapProxy を初期化する際に使用する WSDL で定義されていない場合にスローされます。

SCA::createDataObject

(No version information available, might be only in CVS)

SCA::createDataObject — SDO を作成する

説明

SDO_DataObject **SCA::createDataObject** (string $type_namespace_uri , string $type_name)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドは、SDO を作成して返す必要のある SCA コンポーネントの内部で使用します。 パラメータとして、SDO の名前空間 URI と型名を指定します。 名前空間と型は、コンポーネント内の @types アノテーションで指定されているスキーマファイルのいずれかで定義されていなければなりません。

パラメータ

type_namespace_uri

型の名前空間。

type_name

型の名前。

返り値

新しく作成した SDO_DataObject を返します。

エラー / 例外

SDO_TypeNotFoundException

namespaceURI および typeName が、 @types アノテーションで指定されているスキーマファイルのいずれかで 定義されていない場合にスローされます。

SCA::getService

(No version information available, might be only in CVS)

SCA::getService — サービスのプロキシを取得する

説明

mixed **SCA::getService** (string $target [, string $binding [, array $config]])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

対象を調べ、適切な種類のプロキシを作成して返します。 対象がローカル PHP コンポーネントの場合は SCA_LocalProxy を返します。 対象が WSDL ファイルの場合は SCA_SoapProxy を返します。

パラメータ

target

対象となるサービスあるいはサービスの説明（たとえば json-rpc サービスへの URL や PHP コンポーネント、WSDL ファイルなど）への絶対パスあるいは相対パス。 相対パスを指定した場合は、**getService()** をコールしたスクリプトの位置を基準としてパスを解決します。 include_path や現在の作業ディレクトリは考慮しません。

binding

サービスとの通信に使用するバインディング（つまりプロトコル）。 たとえば json-rpc サービスの場合は binding.jsonrpc となります。target パラメータの内容によっては、サービスの型を自動的に推定してくれることもあります（たとえば target パラメータの最後が .wsdl で終わっている場合、 SCA は binding.soap を使用すると判断します）。 アノテーションの中で使用できるあらゆるバインディングがここで指定できます。 たとえば、'binding.soap' は '@binding.soap' アノテーションと同じ意味となります。

config

そのバインディングで使用する任意の設定プロパティの配列（たとえば array('location' => 'http://example.org')）。 アノテーションの中で使用できるあらゆる設定オプションがここで指定できます。 たとえば 'location' は、soap サービスの場所を設定する '@location' アノテーションと同じ意味です。

返り値

SCA_LocalProxy あるいは SCA_SoapProxy を返します。

例

Example#1 SCA::getService() の例

この例は、http://example.org にある EmailService.wsdl で定義される email soap サービスへのプロキシを取得する方法を示すものです。

```
<?php
include 'SCA/SCA.php';
$service = SCA::getService('EmailService.wsdl', 'binding.soap', array('location' => 'http://example.org'));
$service->send(...);
?>
```

上の例の出力は以下となります。

目次

SDO 関数

導入

Service Data Objects (SDO) を使用すると、PHP アプリケーションから（データベースクエリ・XML ファイル・スプレッドシートなどの）さまざまな形式のデータへのアクセスが 同じインターフェイスで行えるようになります。

データソースにアクセスするには、それぞれのデータソースに対応した アクセス機能を提供するデータアクセスサービス (DAS) が必要となります。PHP アプリケーションからは、データソース内のデータを表す SDO インスタンスを作成するために DAS を使用します。そうすると、 標準的な SDO インターフェイスを使用して SDO インスタンスの データに対する読み書きができるようになります。最後に、 変更されたデータを DAS でデータソースに書き込みます。 このデータソースは、通常は最初に作成したのと同じものとなります。

現在使用可能なデータアクセスサービスの詳細は、 データアクセスサービスの一覧 を参照ください。提供されている DAS 以外にも、SDO は 他のサービスを実装するためのインターフェイスも提供しています（詳細は SDO データアクセスサービス インターフェイス を参照ください）。

この拡張モジュールは、 » Service Data Objects specification の概念に由来するものです。 » Apache Tuscany SDO for C++ project を含んでいます。

Service Data Object の構造

Service Data Object のインスタンスは、データオブジェクトのツリーで 構成されています。このツリーは、データオブジェクト間の閉じた関係で定義されます。例えば、Company（会社）データオブジェクトに いくつかの Department（部署）データオブジェクトが含まれていると すると、Company が Department を包含しているということになります。

SDO では、包含関係以外の関係もデータオブジェクトのツリー内で 保持することができます。例えば、ある Employee（従業員）データオブジェクトが、上司として別の Employee を参照していることが あるかもしれません。

データオブジェクト同士がお互いを参照するだけではなく、 オブジェクト以外のプロパティを保持することもできます。 例えば、Company データオブジェクトに文字列型のプロパティ "name" を含め、そこに会社名 ("Acme" など) を格納することもできます。

これらのデータオブジェクトのプロパティ、すなわち包含関係をもつもの、 包含関係を持たないもの、そしてプリミティブなプロパティは、 ひとつの値を持つこともあれば複数の値を持つこともあります。 上の例では、Departments は複数の値をとりますが Company の name はひとつの値しかとりません。

PHP では、SDO データオブジェクトは PHP のオブジェクトとして表されます。 データオブジェクトのプロパティへは、 オブジェクト構文あるいは連想配列形式の構文でアクセスすることができます。 後でこれらの例を示します。

要件

SDO 拡張モジュールを使用するには、PHP 5.1.0 以降が必要です。 また、libxml2 ライブラリも必要です。通常は、既に libxml2 はインストールされているでしょうが、もしインストールされていない場合は » http://www.xmlsoft.org/ からダウンロードできます。

インストール手順

注意: 初期のバージョンの SDO 拡張モジュールは、XML DAS 用に別の共有ライブラリが必要でした。これは今は使われていないので、php_sdo_das_xml.dll や sdo_das_xml.so への参照を php.ini から削除してください。

Unix システム

1. SDO コンポーネント、すなわち SDO コア・XML DAS およびリレーショナル DAS の最新版をダウンロード、 インストールするには、次のコマンドを使用します。 SDO を構成する 3 つのコンポーネントである SDO コア、XML DAS およびリレーショナル DAS は、ひとつの PECL プロジェクト Service Component Architecture (SCA) にまとめられており、SCA_SDO という名前になっています。つまり、以下のコマンドで、 SCA をダウンロードして SDO の全パーツをインストールすることができます。

```
pecl install SCA_SDO
```

このコマンドは SDO 共有ライブラリをビルドし、 SCA およびリレーショナル DAS の PHP ファイルをインストールします。

最新のベータ版を使用するには、代わりに次のコマンドを使用します。

```
pecl install SCA_SDO-beta
```

2. **pecl** コマンドは、SDO モジュールを自動的に PHP 拡張モジュールディレクトリにインストールします。 SDO 拡張モジュールを有効にするには、以下の行を php.ini に追加しなければなりません。

```
extension=sdo.so
```

PECL パッケージのビルド方法についての詳細は、マニュアルの PECL 拡張モジュールのインストール を参照ください。

Windows

1. SDO の最新版の DLL は、 » php_sdo.dll からダウンロードできます。

   現在は、» pecl4win では安定リリース版のバイナリが提供されていないことに注意しましょう。 最新版のみがダウンロード可能となっています。

2. **pecl** コマンドは、SDO モジュールを自動的に PHP 拡張モジュールディレクトリにインストールします。 SDO 拡張モジュールを有効にするには、以下の行を php.ini に追加しなければなりません。

```
extension=php_sdo.dll
```

3. リレーショナル DAS をダウンロード、インストールするには、 以下のコマンドを実行します。

```
pecl install -B sdo
```

リレーショナル DAS は PHP で書かれています。 sdo/DAS/Relational を含むディレクトリを指すように php.ini の include_path を設定しなければなりません。

Linux での SDO のビルド

この節では、Linux 上で SDO コアおよび XML DAS をビルドする方法を 説明します。これを知る必要があるのは、CVS からチェックアウトした 最新バージョンをビルドしたい場合のみです。

1. 拡張モジュールのメインディレクトリに移動します。 **cd < sdo のコードがある場所 >**
2. **phpize** を実行します。これにより、 SDO をコンパイルするための環境を設定します。
3. 次に **./configure; make; make install** を実行します。 拡張モジュールをインストールするには、root としてログインする 必要があることに注意しましょう。

4. *php.ini ファイルに **extension=sdo.so** を追加することで、このモジュールが PHP に読み込まれるようにします。*

データアクセスサービス

以下の表で、現在提供されている SDO データアクセスサービスの 一覧を示します。

DAS 名	説明
SDO_DAS_XML	XML データアクセスサービスは、SDO を XML ドキュメントとして 読み書きします。
SDO_DAS_Relational	PDO を基にしたデータアクセスサービスで、SDO をリレーショナル データベースとして読み書きします。同時更新に対しては 楽観的な実装となっています。

制限事項

実装上の制限

現在の SDO の実装には、以下の制限があります。

1. *マルチバイト文字セットはサポートしていません。 コミュニティの要望があれば、Unicode 対応版の PHP で対応するかもしれません。 Unicode 関数 を参照ください。*

SDO の制限

現在の PHP の実装では、以下の SDO 2.0 の概念はサポートされていません。 将来これらのすべてが実装されるとは限りません。実装されるかどうかは コミュニティの要望があるかどうかによります。

1. *双方向のリレーション*
2. *型およびプロパティのエイリアス*
3. *読み込み専用のプロパティ*
4. *SDO 2.0 で定義されているヘルパークラスは、直接には実装されていません。 しかし、同等の機能がより PHP らしい方法で提供されています。 例えば、**CopyHelper::copy()** の機能については、 データオブジェクトに PHP の clone キーワードを適用することで実現できます。*

例

以下の例では、XML データアクセスサービスを使用して、 下のスキーマおよびインスタンスをもとに作成した SDO の使用を想定しています。

以下で説明しているインスタンスドキュメントは、 'MegaCorp' という会社を扱っています。この会社には 'Advanced Technologies' という名前の部署がひとつあります。 Advanced Technologies 部門には 3 人の従業員がいます。 この会社の employeeOfTheMonth（今月の従業員）は、2 人目の従業員である 'Jane Doe' を指しています。

```
<?xml version="1.0" encoding="UTF-8" ?>
<company xmlns="companyNS" name="MegaCorp"
    employeeOfTheMonth="E0003">
    <departments name="Advanced Technologies" location="NY" number="123">
       <employees name="John Jones" SN="E0001"/>
       <employees name="Jane Doe" SN="E0003"/>
       <employees name="Al Smith" SN="E0004" manager="true"/>
    </departments>
</company>
```

このスキーマのルート要素は company です。company は departments を含んでおり、個々の department は employees を含んでいます。 各要素はいくつかの属性を保持しており、ここに名前やシリアル番号などを保存します。 最後に、company は IDREF 属性を保持しており、 特定の従業員を 'employeeOfTheMonth' として参照しています。

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:sdo="commonj.sdo"
    xmlns:sdoxml="commonj.sdo/xml"
    xmlns:company="companyNS"
    targetNamespace="companyNS">
    <xsd:element name="company" type="company:CompanyType"/>
    <xsd:complexType name="CompanyType">
      <xsd:sequence>
          <xsd:element name="departments" type="company:DepartmentType"
           maxOccurs="unbounded"/>
      </xsd:sequence>
       <xsd:attribute name="name" type="xsd:string"/>
       <xsd:attribute name="employeeOfTheMonth" type="xsd:IDREF"
          sdoxml:propertyType="company:EmployeeType"/>
    </xsd:complexType>
    <xsd:complexType name="DepartmentType">
      <xsd:sequence>
          <xsd:element name="employees" type="company:EmployeeType"
          maxOccurs="unbounded"/>
      </xsd:sequence>
       <xsd:attribute name="name" type="xsd:string"/>
       <xsd:attribute name="location" type="xsd:string"/>
       <xsd:attribute name="number" type="xsd:int"/>
    </xsd:complexType>
    <xsd:complexType name="EmployeeType">
       <xsd:attribute name="name" type="xsd:string"/>
       <xsd:attribute name="SN" type="xsd:ID"/>
       <xsd:attribute name="manager" type="xsd:boolean"/>
    </xsd:complexType>
</xsd:schema>
```

*XML データアクセスサービスは、スキーマを SDO に関連付けます。 "name" のような属性はプリミティブプロパティに、 一連の employees は複数の値をとる包含関係に、といったようになります。 包含関係は、複合型の中に別の複合型を含めることで表され、 非包含関係は ID や IDREF に特別な属性 **sdoxml:propertyType** を指定して表されることに注意しましょう。*

プロパティの値の設定および取得

以下の例では、上で示したスキーマおよびインスタンスドキュメントをもとに 作成したデータオブジェクトツリーのルート要素が **$company** に入っているものとして考えます。

Example#1 プロパティ名を使用したアクセス

データオブジェクトのプロパティへは、通常のオブジェクトの プロパティにアクセスするのと同じ構文でアクセス可能です。 以下の例では、会社の名前を 'Acme' に変更します。

```
<?php
   $company->name  = 'Acme';
?>
```

Example#2 配列のインデックスを使用したプロパティへのアクセス

連想配列形式の構文を使用して、プロパティにアクセスすることもできます。 最も簡単な負尾法は、プロパティ名を配列のインデックスとして使用する方法です。 例えば、以下の例では会社の名前を設定してから「今月の従業員」を取得しています。

```
<?php
   $company['name'] = 'UltraCorp';
  $eotm  = $company['employeeOfTheMonth'];
?>
```

Example#3 データオブジェクトの繰り返し処理

foreach を使用することにより、データオブジェクトのプロパティを 順に処理していくことができます。以下の例では、 今月の従業員のプロパティを順に処理します。

```
<?php
   $eotm = $company->employeeOfTheMonth;
   foreach ($eotm as $name => $value) {
       echo "$name:  $value¥n";
   }
?>
```

出力は以下のようになります。

```
name:  Jane  Doe
SN:  E0003
```

'manager' プロパティは表示されません。なぜなら設定されていないからです。

Example#4 多くの値をとりうるプロパティへの名前によるアクセス

多くの値をとりうるデータオブジェクトのプロパティへのアクセスも、 オブジェクトのプロパティに対するアクセス構文でアクセスできます。 例えば部署の一覧を取得するにはこのようにします。

```
<?php
   $departments  = $company->departments;
?>
```

Example#5 多くの値をとりうる要素へのアクセス

配列の構文を使用することで、多くの値をとりうる要素における 個々の要素にアクセスすることが可能です。以下の例では、 会社の中の最初の部署にアクセスしています。

```
<?php
   $ad_tech_dept  = $company->departments[0];
?>
```

Example#6 多くの値をとりうるプロパティの順次処理

foreach を使用して、多くの値をとりうるプロパティを順に処理することができます。 会社の各部門を順に処理するにはこのようにします。

```
<?php
  foreach  ($company->departments  as  $department) {
    //  ...
  }
?>
```

繰り返しのたびに、次の部署が変数 **$department** に代入されます。

Example#7 連結されたプロパティへのアクセス

プロパティへの参照を、ひとつの行に連結することができます。 以下の例では、最初の部署の名前を設定し、取得しています。

```
<?php
   $company->departments[0]->name  = 'Emerging  Technologies';
  $dept_name  = $company->departments[0]->name;
?>
```

同じことを連想配列構文で行うと、このようになります。

```
<?php
   $company['departments'][0]['name'] = 'Emerging  Technologies';
  $dept_name  = $company['departments'][0]['name'];
?>
```

どちらの場合でも、dept_name の内容は 'Emerging Technologies' となります。

Example#8 XPath によるナビゲーション

連想配列のインデックスとして、XPath 風の式を使用することができます。 実際に使用できる式は、XPath の拡大サブセットです。

多くの値をとりうるプロパティの表す方法として、 2 通りの方法がサポートされています。 最初の方法は、1 から始まる標準的な XPath の配列構文を使用するものです。 もうひとつは、SDO で XPath を拡張した構文で、 ゼロから始まるインデックスを使用します。 標準的な構文は以下のようになります。

```
<?php
   $jane_doe  = $company["departments[1]/employees[2]"];
```

```
?>
```

そして SDO の拡張版 XPath 構文はこのようになります。

```
<?php
   $jane_doe = $company["departments.0/employees.1"];
?>
```

どちらの例でも、最初の部署の 2 番目の従業員を取得します。

Example#9 XPath による問い合わせ

XPath を使用して、インスタンスのデータ内のデータオブジェクトを 検索することができます。以下の例では、部署 'Advanced Technologies' の manager を取得しています。

```
<?php
  $ad_tech_mgr =
   $company["departments[name='Advanced  Technologies']/employees[manager=true]"];
?>
```

Example#10 子オブジェクトの作成

データオブジェクトは、自分自身の中に子オブジェクトを保持することが できます。子データオブジェクトは自動的にデータグラフの一員となります。 以下の例では、部署 'Advanced Technologies' に新しい従業員を追加しています。

```
<?php
   $ad_tech_dept = $company["departments[name='Advanced  Technologies']"];
  $new_hire = $ad_tech_dept->createDataObject('employees');
  $new_hire->name = 'John  Johnson';
  $new_hire->SN = 'E0005';
  $new_hire->manager = false;
?>
```

Example#11 プリミティブなプロパティの開放

データオブジェクトに項目が登録されているかどうかを確かめたり 項目を削除したりする際には、それぞれ isset() および unset() 関数が使用可能です。

以下の例は、最初の部署の名前を消去します。

```
<?php
  unset($company->departments[0]->name);
?>
```

Example#12 データオブジェクトの開放

unset は、ツリーからデータオブジェクトを削除する場合にも使用できます。 以下の例は、John Jones を会社から削除します。

```
<?php
  unset($company->departments[0]->employees[0]);
?>
```

Example#13 参照されているデータオブジェクトの開放

以下の例では、会社から「今月の従業員」を削除します。 もしこれらの間に包含関係があった場合、参照先の従業員も 削除されてしまいます（その月の一番優秀な従業員を毎月クビにしていく なんてバカなことは、普通はしないでしょう!)。 しかし実際にはこれらは包含関係ではないので、参照先の従業員は会社の中の部署の中に残り続けます。ただ、 employeeOfTheMonth プロパティを通じてはアクセスできなくなります。

```
<?php
  if (isset($company->employeeOfTheMonth)) {
     unset($company->employeeOfTheMonth);
  }
?>
```

Example#14 プロパティのインデックスを使用したアクセス

配列構文を使用して、プロパティのインデックス経由で データオブジェクトのプロパティにアクセスすることができます。 プロパティのインデックスは、モデル（この場合は xml スキーマ）内でそのプロパティの定義が現れる位置となります。 上のスキーマの例では、会社の名前が company の 2 番目の属性であることがわかります（SDO のインターフェイスは、 XML の要素と属性を区別しません)。以下の例では、 会社の名前を 'Acme' に設定します。これは プロパティ名を使用したアクセス と同じ結果となります。

```
<?php
   $company[1] = 'Acme';
?>
```

このようにインデックスを直接使用すると、後で破綻してしまう可能性があります。 通常はプロパティ名を使用する方式を推奨します。 しかし、時にはインデックスによるアクセスが必要になる場合もあるでしょう。

シーケンスデータオブジェクトの使用

シーケンスデータオブジェクトは、データオブジェクト内のさまざまな プロパティの並び順を追跡する SDO です。ここには、非構造化 テキスト要素（SDO のいずれのプロパティにも属さないテキスト要素）を含めることも可能です。シーケンスデータオブジェクトは、 非構造化テキストを許可（つまり、mixed=true である）したり、要素が交互に現れる（

```
<A/><B/><A/>
```

）ような XML 文書を扱う場合に有用です。 これは、例えば order に choice を指定した複合型の要素に対して スキーマで maxOccurs>1 を定義している場合などに発生します。

以下の例では、XML データアクセスサービスを使用して、 下のスキーマおよびインスタンスをもとに作成した SDO の使用を想定しています。

このスキーマでは、手紙の書式を定義しています。letter（手紙）には オプションで 3 つのプロパティ date（日付)、firstName（名前)、 および lastName（苗字）を含めることが可能です。このスキーマでは **mixed="true"** としていますが、これは 3 つのプロパティの間に非構造化テキストをちりばめることが できるということを意味します。

```
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:letter="http://letterSchema"
    targetNamespace="http://letterSchema">
```

```
    <xsd:element  name="letters"  type="letter:FormLetter"/>
    <xsd:complexType  name="FormLetter"  mixed="true">
      <xsd:sequence>
          <xsd:element  name="date"  minOccurs="0"  type="xsd:string"/>
          <xsd:element  name="firstName"  minOccurs="0"  type="xsd:string"/>
          <xsd:element  name="lastName"  minOccurs="0"  type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

以下が、手紙のインスタンスです。これには 3 つのプロパティ date、firstName および lastName が含まれており、非構造化 テキスト要素として住所と本文が含まれています。

```
<letter:letters  xmlns:letter="http://letterSchema">
   <date>March 1, 2005</date>
   Mutual of Omaha
   Wild Kingdom, USA
   Dear
    <firstName>Casy</firstName>
    <lastName>Crocodile</lastName>
   Please buy more shark repellent.
   Your premium is past due.
</letter:letters>
```

このインスタンスが読み込まれると、letter データオブジェクトの シーケンス番号およびプロパティ番号は以下の表のようになります。

シーケンスのインデックス	プロパティのインデックス:名前	値
0	0:date	March 1, 2005
1	-	Mutual of Omaha
2	-	Wild Kingdom, USA
3	-	Dear
4	1:firstName	Casy
5	2:lastName	Crocodile
6	-	Please buy more shark repellent.
7	-	Your premium is past due.

シーケンス番号の整合性を保つため、シーケンスデータオブジェクトの 操作には SDO_Sequence インターフェイスを使用しなければなりません。 これにより、データオブジェクトに対しての操作を プロパティのインデックスではなくシーケンスのインデックスを使用して 行えるようになります (上の表を参照ください)。 以下の例では、letter のインスタンスが データオブジェクト **$letter** として 読み込まれているものと仮定します。

Example#15 SDO_Sequence インターフェイスの取得

データオブジェクトのシーケンスを取得するには **getSequence()** メソッドを使用します。 以下の例では、letter データオブジェクトのシーケンスを取得します。

```
<?php
   $letter_seq = $letter->getSequence();
?>
```

これ以降の例では、letter データオブジェクトのシーケンスが 変数 **$letter_seq** に代入されているものとします。

Example#16 シーケンスの値の取得/設定

シーケンスのインデックスを使用することで、(非構造化 テキストも含め) それぞれの値を取得したり設定したりすることが可能です。 以下の例では、ファーストネームを 'Snappy' に変更して シーケンスの最後の値 (非構造化テキスト 'Your premium is past due.') を取得します。

```
<?php
   $letter_seq[4] = 'Snappy';
  $text = $letter_seq[count($letter_seq) - 1];
?>
```

Example#17 シーケンスの繰り返し処理

foreach を使用して、個々のシーケンス値について処理を繰り返すことが できます。以下の例では、シーケンス順に個々の値を処理します。

```
<?php
  foreach ($letter->getSequence() as $value) {
    // ...
  }
?>
```

Example#18 シーケンスとデータオブジェクトの比較

データオブジェクトのインターフェイスから設定した値は、シーケンスの 一部とはならないことがあります。データオブジェクト経由で値を 設定した場合、そのプロパティが既にシーケンスに組み込まれていた 場合にのみシーケンスからアクセス可能となります。 以下の例では、データオブジェクト経由で lastName を設定して それをシーケンスから取得しています。これがうまくいくのは、 lastName が既にシーケンス内に存在するからです。もしこれが 事前に設定されていなかった場合、lastName には 'Smith' が 設定されますが、それはシーケンスの一部とはなりません。

```
<?php
   $letter[2] = 'Smith';
  $last_name  = $letter_seq[5];
?>
```

Example#19 シーケンスへの追加

シーケンスに新しい値を追加するには SDO_Sequence::insert() メソッドを使用します。以下の例では、プロパティ 'firstName' および 'lastName' が最初に削除されているものと仮定します。

```
<?php
  // シーケンスに firstName を追加します
   // 値: 'Smith'
```

```
  // シーケンスのインデックス: NULL (追加)
  // プロパティ ID: 1 (firtName プロパティのインデックス)
  $letter_seq->insert('Smith', NULL, 1);

  // シーケンスに lastName を追加します
  // 値: 'Jones'
  // シーケンスのインデックス: NULL (追加)
  // プロパティ ID: 'lastName' (lastName プロパティの名前)
  $letter_seq->insert('Jones', NULL, 'lastName');

  // 非構造化テキストを追加します
  // 値: 'Cancel Subscription.'
  // シーケンスのインデックス: 省略 (追加)
  // プロパティ ID: 省略 (非構造化テキスト)
  $letter_seq->insert('Cancel Subscription.');

  // 非構造化テキストを間に挿入します。それ以降のシーケンス値は
  // ひとつずれます。
  // 値: 'Care of:'
  // シーケンスのインデックス: 1 (2 番目の要素として挿入する)
  // プロパティ ID: 省略 (非構造化テキスト)
  $letter_seq->insert('Care of:', 1);
?>
```

Example#20 シーケンスからの削除

シーケンスに項目が登録されているかどうかを確かめたり シーケンスから項目を削除したりする際には、それぞれ isset() および unset() 関数が使用可能です (注意: unset() は、現時点では値を データオブジェクト内に残します。しかし、この挙動はおそらく今後 変更され、データオブジェクトからもデータを削除することに なるでしょう)。シーケンスの挙動は、リスト構造に似ています。 そのため、シーケンスの途中の項目を削除すると、それ以降の インデックスがひとつずつ小さいほうにずれます。以下の例では、 インデックスの最初の要素が存在するかどうか調べ、存在する場合に それを削除しています。

```
<?php
  if (isset($letter_seq[0])) {
    unset($letter_seq[0]);
  }
?>
```

サービスデータオブジェクトへのリフレクション

SDO は、自分自身を作成するもととなったオブジェクト (モデル) の構造を知っています。例えば、上の Company XML スキーマ を使用して 作成した Company SDO には DepartmentType データオブジェクトしか 含めることはできず、同様に DepartmentType データオブジェクトには EmployeeType データオブジェクトしか含めることはできません。

実行時にモデルの情報にアクセスできると、以下のような利点があります。 例えば、データオブジェクトに値を設定するためのユーザインターフェイスを 自動的に作成することができます。モデルの情報にアクセスするには、 リフレクションを使用します。

Example#21 データオブジェクトへのリフレクション

以下の例では、空の Employee データオブジェクトへのリフレクションの 方法を示します。

```
<?php
  // employee データオブジェクトを (例えば XML データアクセスサービスなどから) 作成します
  $employee = ...;
  $reflection = new SDO_Model_ReflectionDataObject($employee);
  print($reflection);
?>
```

上の例の出力は以下となります。

```
object(SDO_Model_ReflectionDataObject)#4 { - ROOT OBJECT - Type {
companyNS:EmployeeType[3] { commonj.sdo:String $name;
commonj.sdo:String $SN; commonj.sdo:Boolean $manager; } }
```

SDO_Model_ReflectionDataObject に対して print を使用すると、 データオブジェクトのモデルを出力します。この出力結果から、companyNS:EmployeeType 型には 3 つのプロパティがあることや それぞれのプロパティの名前と型を知ることができます。 プリミティブ型は、SDO の型として表示されることに注意しましょう (例: commonj.sdo namespace, String type)。これは SDO モデルであり、 アプリケーションからは PHP の同等の型 (例: string および boolean) として扱えることを知っておくとよいでしょう。

Example#22 型情報へのアクセス

リフレクションを使用して、データオブジェクトの型情報を取得することが できます。以下の例では、まず型がプリミティブ型ではなく データオブジェクトに対応するものであることを確認し、各プロパティの 名前を書き出しています ($type および $property は、それぞれ SDO_Model_Type および SDO_Model_Property オブジェクトです)。

```
<?php
    // employee データオブジェクトを (例えば XML データアクセスサービスなどから) 作成します
    $employee = ...;
    $reflection = new SDO_Model_ReflectionDataObject($employee);
    $type = $reflection->getType();
    if (! $type->isDataType()) {
        foreach ($type->getProperties() as $property) {
            print $property->getName() . "¥n";
        }
    }
?>
```

上の例の出力は以下となります。

```
name
SN
manager
```

定義済みクラス

SDO は、3 つの系統のインターフェイスから構成されています。ひとつめは 典型的な SDO アプリケーションで使用されるインターフェイスをカバーする もので、これらには 'SDO_' という名前がつけられています。2 番目は データオブジェクトのモデルを扱うためのもので、これらには 'SDO_Model_' という名前がつけられています。最後はデータアクセスサービスで 使用されるもので、これらには 'SDO_DAS_' という名前がつけられています。 SDO を使用するほとんどの人は、'SDO_Model_' や 'SDO_DAS_' を使用したり理解したりする必要はないでしょう。

SDO API

SDO_DataObject

データオブジェクトを操作する際のメインとなるインターフェイスです。 以下で説明するメソッドに加え、SDO_DataObject は ArrayAccess、SDO_PropertyAccess （プロパティへのアクセスのために **__get()** / **__set()** メソッドをオーバーロードします）、 Iterator および Countable インターフェイスを継承します。

メソッド

- getSequence - データオブジェクトのシーケンスを取得する
- createDataObject - 子データオブジェクトを作成する
- clear - データオブジェクトのプロパティの設定を解除する
- getContainer - このオブジェクトのコンテナ（あるいは「親」）を取得する
- getTypeName - このデータオブジェクトの型の名前を取得する
- getTypeNamespaceURI - このデータオブジェクトの型の名前空間 URI を取得する

SDO_Sequence

シーケンスデータオブジェクトが、データオブジェクトのプロパティの 並び順を保持したままアクセスをおこなったり 非構造化テキストを許可したりする際に使用するインターフェイスです。 SDO_Sequence は、インデックスが常に連続した数となるようにします。 そのため、要素を挿入したり削除したりすると、その他の要素のインデックスが 増減します。以下で説明するメソッドに加え、SDO_Sequence は ArrayAccess、Iterator および Countable インターフェイスを 継承しています。

メソッド

- getProperty - シーケンスのインデックスからプロパティを取得する
- move - 指定した要素を、現在のプロパティインデックスから別の位置に移動する
- insert - シーケンスに新しい値を挿入する

SDO_List

複数の値を保持するプロパティを操作するためのインターフェイスです。 以下に挙げるメソッドに加え、SDO_List は ArrayAccess、 Iterator および Countable を継承しています。 SDO_List は、常にインデックスが連続した数となるようにします。 そのため、要素を挿入したり削除したりすると 他の要素のインデックスが変化します。

メソッド

- insert - 新しい値をリストに挿入する

SDO_DataFactory

データオブジェクトを作成するためのインターフェイスです。 ファクトリのモデルに値を設定する（すなわち、作成可能なデータ オブジェクトの型および構造情報をもとにデータファクトリを設定する）のはデータアクセスサービスの役目です。また、オプションとして SDO_DataFactory インターフェイスのインスタンスを返すか、 それを実装します。

メソッド

- create - 新しいデータオブジェクトを作成する

SDO_Exception

SDO_Exception は、呼び出し元の要求を完了させられなかった際にスローされます。 SDO_Exception のサブクラスには以下のようなものがあります。

- SDO_PropertyNotSetException - 指定されたプロパティは存在するが、まだ値が設定されていません。 あるいはデフォルト値が設定されていません。
- SDO_PropertyNotFoundException - 指定されたプロパティはデータオブジェクトの型の一部ではありません。
- SDO_TypeNotFoundException - 指定された名前空間 URI あるいは型名が見つかりません。
- SDO_InvalidConversionException - 代入時に、型の変換を行うことができません。
- SDO_IndexOutOfBoundsException - データオブジェクトやシーケンス、リストにおける数値インデックスが 有効な範囲にありません。
- SDO_UnsupportedOperationException - その操作が許可されていないため、要求を完了できませんでした。 例えば読み込み専用のプロパティに値を設定しようとした場合などです。

メソッド

組み込みの Exception クラスから継承したものに加え、次のメソッドが追加されています。

- getCause - この SDO_Exception の原因を取得する

SDO リフレクション API

SDO_Model_ReflectionDataObject

データオブジェクトのインスタンス自身について、モデルの型や プロパティの情報を取得するためのインターフェイスです。 PHP 5 で使用可能となったリフレクションパターンに従うように 設計されています。

コンストラクタ

- __construct - 新しい SDO_Model_ReflectionDataObject を作成する

メソッド

- export - データオブジェクトを表す文字列を取得する
- getType - データオブジェクトの SDO_Model_Type を取得する
- getInstanceProperties - データオブジェクトのインスタンスのプロパティを取得する
- getContainmentProperty - データオブジェクトとの包含関係を定義しているプロパティを取得する

SDO_Model_Type

データオブジェクトの型に関する情報を取得するための インターフェイスです。このインターフェイスは、型の名前および 名前空間 URI・オープン型のデータを許可するかどうかなどを 調べるために使用します。

メソッド

- getName - 型の名前を取得する
- getNamespaceURI - 型の名前空間 URI を取得する
- isInstance - データオブジェクトが特定の型のインスタンスであるかどうかを調べる
- getProperties - 型のプロパティを取得する
- getProperty - 型のプロパティを取得する
- isDataType - この型が基本的なスカラー型であるかどうかを調べる
- isSequencedType - この型がシーケンス型であるかどうかを調べる
- isOpenType - この型がオープン型であるかどうかを調べる
- isAbstractType - この型が抽象型であるかどうかを調べる
- getBaseType - この型の基底型を（もし存在すれば）取得する

SDO_Model_Property

データオブジェクトのプロパティに関する情報を取得するための インターフェイスです。このインターフェイスは、プロパティの型・ デフォルト値の存在・包含関係の有無・カーディナリティなどを 調べるために使用します。

メソッド

- getName - プロパティの名前を取得する
- getType - プロパティの型を取得する
- isMany - プロパティが複数の値を保持するかどうかを調べる
- isContainment - プロパティが包含関係を保持するかどうかを調べる
- getContainingType - このプロパティを含んでいる型を取得する
- getDefault - プロパティのデフォルト値を取得する

SDO データアクセスサービス 開発者用インターフェイス

SDO_DAS_DataObject

データアクセスサービスが、データオブジェクトの SDO_DAS_ChangeSummary にアクセスするためのインターフェイスです。これは、変更した内容をデータベースに書き戻す際に、データアクセスサービスがデータの 衝突を調べるために使用されます。

メソッド

- getChangeSummary - データオブジェクトの変更内容を取得する

SDO_DAS_ChangeSummary

データオブジェクトの変更履歴にアクセスするためのインターフェイスです。 ログ取得が有効になってからデータオブジェクトに発生したあらゆる変更が、 変更内容に含まれます。削除あるいは修正の場合は、変更前の情報も保持されます。

ログ取得が有効になっていない場合、ログ取得が無効にされた時点までの 変更内容が保存されます。ログ取得を再開すると、過去の変更内容は消去されます。 これは、変更内容を DAS で書き出した後にデータオブジェクトを 再利用する場合に有用です。

メソッド

- beginLogging - データオブジェクトの変更履歴のログ取得を開始する
- endLogging - データオブジェクトの変更履歴のログ取得を終了する
- isLogging - ログ取得が有効になっているかどうかを調べる
- getChangedDataObjects - 変更されたデータオブジェクトの一覧を取得する
- getChangeType - データオブジェクトの変更の型を取得する
- getOldValues - データオブジェクトの変更前の値を取得する
- getOldContainer - 削除されたデータオブジェクトの削除前のコンテナを取得する

SDO_DAS_Setting

プロパティの古い値にアクセスするためのインターフェイスです。 設定の一覧は、 getOldValues() で返されます。 .

メソッド

- getPropertyIndex - 変更されたプロパティのプロパティインデックスを取得する
- getPropertyName - 変更されたプロパティのプロパティ名を取得する
- getValue - 変更されたプロパティの古い値を取得する
- getListIndex - 複数の値を持つプロパティだった場合に、古い値のインデックスの 一覧を取得する
- isSet - 変更前に、プロパティが設定されていたかどうかを調べる

SDO_DAS_DataFactory

SDO_DataObject のモデルを作成するインターフェイスです。 SDO_DAS_DataFactory は抽象クラスで、それを実装した具象データファクトリを 返すスタティックメソッドを提供します。この実装クラスを使用して、 データアクセスサービスが SDO モデルを作成します。 例えばリレーショナル データアクセスサービスは、リレーショナル データベースのスキーマをもとにして SDO_DAS_DataFactory モデルを 作成し、そこに値を設定します。

メソッド

- getDataFactory - 具象データファクトリのインスタンスを取得するためのスタティックメソッド
- addType - SDO モデルに新しい型を追加する
- addPropertyToType - SDO モデルの型定義に新しいプロパティを追加する

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

SDO_DAS_ChangeSummary::NONE=0 (integer)
変更の型 'none (変更していない)' を表します。
SDO_DAS_ChangeSummary::MODIFICATION=1 (integer)
変更の型 'modification (修正)' を表します。
SDO_DAS_ChangeSummary::ADDITION=2 (integer)
変更の型 'addition (追加)' を表します。
SDO_DAS_ChangeSummary::DELETION=3 (integer)
変更の型 'deletion (削除)' を表します。

SDO_DAS_ChangeSummary::beginLogging

(No version information available, might be only in CVS)

SDO_DAS_ChangeSummary::beginLogging — 変更内容の記録を開始する

説明

void **SDO_DAS_ChangeSummary::beginLogging** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject への変更内容の記録を開始します。

パラメータ

なし。

返り値

何も返しません。

SDO_DAS_ChangeSummary::endLogging

(No version information available, might be only in CVS)

SDO_DAS_ChangeSummary::endLogging — 変更内容の記録を終了する

説明

void **SDO_DAS_ChangeSummary::endLogging** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject への変更内容の記録を終了します。

パラメータ

なし。

返り値

何も返しません。

SDO_DAS_ChangeSummary::getChangeType

(No version information available, might be only in CVS)

SDO_DAS_ChangeSummary::getChangeType — SDO_DataObject への変更の型を取得する

説明

int **SDO_DAS_ChangeSummary::getChangeType** (SDO_DataObject $dataObject)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定した SDO_DataObject に対する変更の型を取得します。

パラメータ

dataObject

変更された SDO_DataObject。

返り値

行われた変更の型を返します。変更の型は列挙型で表され、 以下の 4 つの値のうちいずれかです。

- SDO_DAS_ChangeSummary::NONE
- SDO_DAS_ChangeSummary::MODIFICATION
- SDO_DAS_ChangeSummary::ADDITION
- SDO_DAS_ChangeSummary::DELETION

SDO_DAS_ChangeSummary::getChangedDataObjects

(No version information available, might be only in CVS)

SDO_DAS_ChangeSummary::getChangedDataObjects — 変更内容から、変更されたデータオブジェクトを取得する

説明

SDO_List **SDO_DAS_ChangeSummary::getChangedDataObjects** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変更された SDO_DataObjects の SDO_List を取得します。 取得したデータオブジェクトから、それらの変更前の値や 変更の型を取得できます。

パラメータ

なし。

返り値

SDO_DataObjects の SDO_List を返します。

SDO_DAS_ChangeSummary::getOldContainer

(No version information available, might be only in CVS)

SDO_DAS_ChangeSummary::getOldContainer — 削除された SDO_DataObject の削除前のコンテナを取得する

説明

SDO_DataObject **SDO_DAS_ChangeSummary::getOldContainer** (SDO_DataObject $data_object)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

削除された SDO_DataObject の削除前のコンテナ (SDO_DataObject) を取得します。

パラメータ

data_object

削除された SDO_DataObject で、そのコンテナを調べたいもの。

返り値

削除された SDO_DataObject の削除前のコンテナオブジェクトを返します。

SDO_DAS_ChangeSummary::getOldValues

(No version information available, might be only in CVS)

SDO_DAS_ChangeSummary::getOldValues — 変更された SDO_DataObject の変更前の値を取得する

説明

SDO_List **SDO_DAS_ChangeSummary::getOldValues** (SDO_DataObject $data_object)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変更された SDO_DataObject の変更前の値のリストを取得します。 SDO_DataObject で変更されたプロパティの、変更前の値を示す SDO_DAS_Settings のリストを返します。

パラメータ

data_object

変更されたデータオブジェクト。

返り値

SDO_DataObject で変更されたプロパティの、変更前の値を示す SDO_DAS_Settings のリストを返します。 変更の型が SDO_DAS_ChangeSummary::ADDITION の場合は、 これは空のリストとなります。

SDO_DAS_ChangeSummary::isLogging

(No version information available, might be only in CVS)

SDO_DAS_ChangeSummary::isLogging — 変更内容が記録されるようになっているかどうかを調べる

説明

bool **SDO_DAS_ChangeSummary::isLogging** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変更内容が記録されるようになっているかどうかを調べます。

パラメータ

なし。

返り値

変更内容の記録が有効になっている場合に **TRUE**、それ以外の場合に **FALSE** を返します。

SDO_DAS_DataFactory::addPropertyToType

(No version information available, might be only in CVS)

SDO_DAS_DataFactory::addPropertyToType — 型にプロパティを追加する

説明

void **SDO_DAS_DataFactory::addPropertyToType** (string $parent_type_namespace_uri , string $parent_type_name , string $property_name , string $type_namespace_uri , string $type_name [, array $options])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

型にプロパティを追加します。この型は、常に SDO_DAS_DataFactory が 知っていなければなりません (つまり、addType() を使用して追加されたもので なければなりません)。このプロパティは、型のプロパティとなります。 これを使用することによって、SDO_DataObject の構造を表すグラフモデルが 作成されます。

パラメータ

parent_type_namespace_uri

親となる型の名前空間 URI。

parent_type_name

親となる型の名前。

property_name

親となる型から参照するためのプロパティ名。

type_namespace_uri

プロパティの型の名前空間 URI。

type_name

プロパティの型の名前。

options

このプロパティに設定する属性を、キー=>値 の組み合わせで保持する配列。以下のキーワードが指定可能です。

many

このプロパティが複数の値を保持するかどうかを宣言するフラグ。 'true' を指定すると、このプロパティが複数の値を保持するものとして 扱われます (デフォルトは 'false' です)。

readOnly

このプロパティが読み込み専用であるかどうかを宣言するフラグ。 'true' を指定すると、SDO アプリケーション API からは このプロパティを変更できなくなります (デフォルトは 'false' です)。

containment

このプロパティと親の間に包含関係があるかどうかを宣言するフラグ。 'true' を指定すると、親との間に包含関係があるということになります。 'false' の場合は、包含関係のない単なる参照となります (デフォルトは 'true' です)。 このフラグは、データオブジェクト型のプロパティを 追加するときにのみ使用されます。それ以外の場合は無視されます。

default

このプロパティのデフォルト値。このキーを省略した場合は、 プロパティがデフォルト値を持たないことを意味します。 デフォルト値を保持することができるのは、プロパティが 単一の値をとるデータ型 (プリミティブ型) の場合のみです。

返り値

None.

変更履歴

バージョン	説明
0.5.2	オプションのパラメータ many、 readOnly および containment は廃止予定で、options 配列の使用が 推奨されます。

例

Example#1 SDO_DAS_DataFactory::addPropertyToType() の例

以下の例では、親の型にプロパティ 'addressline' を追加します。 親の型は、名前空間 'PersonNS' にある 'PersonType' 型です。 'addressline' プロパティの型は、複数の値を保持する SDO データ型 (プリミティブ) で、名前空間 'commonj.sdo' および名前 'String' で表されるものです。

```
<?php
  $df->addPropertyToType('PersonNS', 'PersonType',
    'addressline', 'commonj.sdo', 'String', array('many'=>true));
?>
```

SDO_DAS_DataFactory::addType

(No version information available, might be only in CVS)

SDO_DAS_DataFactory::addType — モデルに新しい型を追加する

説明

```
void SDO_DAS_DataFactory::addType ( string $type_namespace_uri , string $type_name [, array $options ] )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

名前空間および名前を指定して、SDO_DAS_DataFactory に新しい型を 追加します。この型はデータオブジェクトのモデルの一部となり、 データファクトリからこのモデルを作成可能となります。

パラメータ

type_namespace_uri

型の名前空間。

type_name

型の名前。

options

型の属性値として設定する、ひとつあるいは複数の key=>value のペアをこの配列に保持します。オプションのキーワードは次のとおりです。

open

型がオープンかどうかを指定するフラグです。 オープン型の SDO_DataObject は、 その型以外のプロパティを追加することが可能です。 これは、スキーマで <xsd:any> 要素のようなオープンなコンテンツを サポートしている XML 文書を扱う場合に使用されます。 デフォルト値は 'false' です。

sequenced

型がシーケンスをサポートしているかどうかを指定するフラグです。 シーケンス型は、複数のプロパティにまたがってその順序を管理することが可能で、 非構造化テキストを扱うことができます。 デフォルト値は 'false' です。シーケンス型についての詳細な情報は シーケンスデータオブジェクトの使用 を参照ください。

basetype

指定されている場合は、この型の継承元の名前空間 URI および型名の配列です。 これを使用するのは、例えば XML スキーマで <extension base="..."> を使用して他の型から継承している場合などです。

返り値

なし。

例

Example#1 SDO_DAS_DataFactory::addType() の例

以下の例では、名前空間 'CompanyNS' に属する 'CompanyType' という 名前の新しいデータオブジェクト型を追加します。

```
<?php
   $df->addType('CompanyNS', 'CompanyType');
?>
```

SDO_DAS_DataFactory::getDataFactory

(No version information available, might be only in CVS)

SDO_DAS_DataFactory::getDataFactory — データファクトリのインスタンスを取得する

説明

```
SDO_DAS_DataFactory SDO_DAS_DataFactory::getDataFactory ( void )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DAS_DataFactory のインスタンスを取得するための スタティックメソッドです。このインスタンスは、取得当初は 基本的な SDO 型しか使用できないように設定されています。 データファクトリモデルを設定するのは データアクセスサービスの役目となります。 モデルを設定することで、PHP アプリケーションから SDO_DataFactory インターフェイスを使用して モデルを基にした SDO を作成できるようになります。 PHP アプリケーションは、データファクトリを常にデータアクセスサービスから 取得すべきです。このインターフェイスを使用すべきではありません。

パラメータ

なし。

返り値

SDO_DAS_DataFactory を返します。

SDO_DAS_DataObject::getChangeSummary

(No version information available, might be only in CVS)

SDO_DAS_DataObject::getChangeSummary — データオブジェクトの変更内容を取得する

説明

SDO_DAS_ChangeSummary **SDO_DAS_DataObject::getChangeSummary** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DAS_DataObject の SDO_DAS_ChangeSummary、 あるいは存在しない場合に NULL を取得します。

パラメータ

なし。

返り値

SDO_DAS_DataObject の SDO_DAS_ChangeSummary、 あるいは存在しない場合に NULL を返します。

SDO_DAS_Setting::getListIndex

(No version information available, might be only in CVS)

SDO_DAS_Setting::getListIndex — 複数の値を持つプロパティのインデックスを取得する

説明

int **SDO_DAS_Setting::getListIndex** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

複数の値を持つプロパティの要素に変更が加えられた際に、その インデックスを取得します。 例えば、複数の値を持つプロパティの 3 番目の要素を変更したときには、 対応する変更内容オブジェクトから SDO_DAS_Setting を取得することができます。 取得した SDO_DAS_Setting 上で **getListIndex()** をコールすると、2 が返されます (リストのインデックスは 0 から数え始めます)。

パラメータ

なし。

返り値

複数の値を持つプロパティのうち、変更が加えられた要素の インデックスを返します。

SDO_DAS_Setting::getPropertyIndex

(No version information available, might be only in CVS)

SDO_DAS_Setting::getPropertyIndex — 変更されたプロパティのプロパティインデックスを取得する

説明

int **SDO_DAS_Setting::getPropertyIndex** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変更されたプロパティのプロパティインデックスを返します。 このインデックスにより、データオブジェクト内で変更されたプロパティを 識別します。

パラメータ

なし。

返り値

変更されたプロパティのプロパティインデックスを返します。

SDO_DAS_Setting::getPropertyName

(No version information available, might be only in CVS)

SDO_DAS_Setting::getPropertyName — 変更されたプロパティのプロパティ名を取得する

説明

string **SDO_DAS_Setting::getPropertyName** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変更されたプロパティのプロパティ名を返します。 この名前により、データオブジェクト内で変更されたプロパティを 識別します。

パラメータ

なし。

返り値

変更されたプロパティのプロパティ名を返します。

SDO_DAS_Setting::getValue

(No version information available, might be only in CVS)

SDO_DAS_Setting::getValue — 変更されたプロパティの変更前の値を取得する

説明

mixed **SDO_DAS_Setting::getValue** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変更されたプロパティの変更前の値を返します。 これは、データアクセスサービスが更新されたデータをデータソースに 書き込む際に使用可能です。DAS は、変更前の値をデータソースの 現在の値を比較することによって衝突を検出します。 もし 2 つが一致しなかったとしたら、データオブジェクトが作成された後に データソースが変更されているということであり、そのまま新しいデータを 書き込むとデータの整合性を崩す恐れがあります。

パラメータ

なし。

返り値

変更されたプロパティの変更前の値を返します。

SDO_DAS_Setting::isSet

(No version information available, might be only in CVS)

SDO_DAS_Setting::isSet — 変更前にプロパティが設定されていたかどうかを調べる

説明

bool **SDO_DAS_Setting::isSet** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

変更前にプロパティが設定されていたかどうかを調べます。 もし変更前に設定されていた場合、SDO_DAS_Setting は 変更前の値も含んでいます。

パラメータ

なし。

返り値

変更前にプロパティが設定されていた場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

SDO_DataFactory::create

(No version information available, might be only in CVS)

SDO_DataFactory::create — SDO_DataObject を作成する

説明

void **SDO_DataFactory::create** (string $type_namespace_uri , string $type_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

名前空間 URI および型の名前を指定して、新しい SDO_DataObject を作成します。

パラメータ

type_namespace_uri

作成する型の名前空間。

type_name

作成する型の名前。

返り値

新しく作成した SDO_DataObject を返します。

エラー / 例外

SDO_TypeNotFoundException

このデータファクトリが知らない namespaceURI および typeName が指定された場合にスローされます。

SDO_DataObject::clear

(No version information available, might be only in CVS)

SDO_DataObject::clear — SDO_DataObject のプロパティを消去する

説明

void ***SDO_DataObject::clear*** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject のプロパティを消去します。 読み込み専用のプロパティは影響を受けません。データオブジェクトに 対するこれ以降の isset() のコールは **FALSE** を返します。

パラメータ

なし。

返り値

なにも値を返しません。

SDO_DataObject::createDataObject

(No version information available, might be only in CVS)

SDO_DataObject::createDataObject — 新しい子 SDO_DataObject を作成する

説明

SDO_DataObject ***SDO_DataObject::createDataObject*** *(mixed $identifier)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したプロパティのデフォルト型で、子データオブジェクト SDO_DataObject を作成します。データオブジェクトは 自動的にツリーに挿入され、オブジェクトへのリファレンスが返されます。

パラメータ

identifier

作成されるデータオブジェクト型のプロパティを指定します。 プロパティ名 (string) 、プロパティのインデックス (int) あるいは SDO_Model_Property のいずれかが指定可能です。

返り値

新しく作成した SDO_DataObject を返します。

エラー / 例外

SDO_PropertyNotFoundException

identifier に対応するデータオブジェクトのプロパティが 存在しない場合にスローされます。

SDO_DataObject::getContainer

(No version information available, might be only in CVS)

SDO_DataObject::getContainer — データオブジェクトのコンテナを取得する

説明

SDO_DataObject **SDO_DataObject::getContainer** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このデータオブジェクトを保有しているデータオブジェクトを取得します。

パラメータ

なし。

返り値

この SDO_DataObject を保有している SDO_DataObject、あるいは もしこのオブジェクトが階層のルートである場合（つまり、コンテナが存在しない場合）は NULL を返します。

SDO_DataObject::getSequence

(No version information available, might be only in CVS)

SDO_DataObject::getSequence — データオブジェクトのシーケンスを取得する

説明

SDO_Sequence **SDO_DataObject::getSequence** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この SDO_DataObject に対応する SDO_Sequence を取得します。 SDO_Sequence インターフェイスを使用したアクセスの場合も 同じ SDO_DataObject インスタンスのデータに対して操作を行いますが、 複数プロパティにまたがった並び順を管理することができます。

パラメータ

なし。

返り値

この SDO_DataObject に対応する SDO_Sequence、 あるいは SDO_DataObject がシーケンスを保持できない型であった場合に NULL を返します。

SDO_DataObject::getTypeName

(No version information available, might be only in CVS)

SDO_DataObject::getTypeName — データオブジェクトの型の名前を取得する

説明

string **SDO_DataObject::getTypeName** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

データオブジェクトの型の名前を返します。 SDO_Model_ReflectionDataObject::getType().getName() に対応する便利なメソッドです。

パラメータ

なし。

返り値

データオブジェクトの型の名前を返します。

SDO_DataObject::getTypeNamespaceURI

(No version information available, might be only in CVS)

SDO_DataObject::getTypeNamespaceURI — このデータオブジェクトの型の名前空間 URI を取得する

説明

string **SDO_DataObject::getTypeNamespaceURI** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

データオブジェクトの型の名前空間 URI を返します。 SDO_Model_ReflectionDataObject::getType().getNamespaceURI() に対応する便利な

メソッドです。

パラメータ

なし。

返り値

データオブジェクトの型の名前空間 URI を返します。

SDO_Exception::getCause

(No version information available, might be only in CVS)

SDO_Exception::getCause — 例外の原因を取得する

説明

mixed **SDO_Exception::getCause** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この例外の原因を返します。原因が存在しない場合や不明な場合には NULL を返します。典型的な原因は SDO_CPPException オブジェクトで、 追加の診断情報を取得するために使用されます。

パラメータ

なし。

返り値

この例外の原因を返します。原因が存在しない場合や不明な場合には NULL を返します。

SDO_List::insert

(No version information available, might be only in CVS)

SDO_List::insert — リストに挿入する

説明

void **SDO_List::insert** (mixed $value [, int $index])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

リスト内の指定した位置に、新しい要素を挿入します。 リストのそれ以降の項目は、ひとつ後ろにずれます。

パラメータ

value

挿入される新しい値。 プリミティブ型あるいは SDO_DataObject のいずれかです。

index

新しい要素を挿入する位置。 指定しない場合は、新しい値は最後に追加されます。

返り値

なし。

エラー / 例外

SDO_IndexOutOfBoundsException

インデックスがゼロより小さい場合、あるいはインデックスが リストのサイズより大きい場合にスローされます。

SDO_InvalidConversionException

新しい値の型がリストに使用できない場合（例: 多くの値をとりうるプロパティ）にスローされます。

SDO_Model_Property::getContainingType

(No version information available, might be only in CVS)

SDO_Model_Property::getContainingType — このプロパティを含む SDO_Model_Type を取得する

説明

SDO_Model_Type ***SDO_Model_Property::getContainingType*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このプロパティを含む SDO_Model_Type を返します。

パラメータ

なし。

返り値

このプロパティを含む SDO_Model_Type を返します。

SDO_Model_Property::getDefault

(No version information available, might be only in CVS)

SDO_Model_Property::getDefault — プロパティのデフォルト値を取得する

説明

mixed ***SDO_Model_Property::getDefault*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

プロパティのデフォルト値を返します。 プリミティブデータ型のプロパティのみが デフォルト値を持つことができます。

パラメータ

なし。

返り値

プロパティのデフォルト値を返します。

SDO_Model_Property::getName

(No version information available, might be only in CVS)

SDO_Model_Property::getName — SDO_Model_Property の名前を取得する

説明

string ***SDO_Model_Property::getName*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_Model_Property の名前を返します。

パラメータ

なし。

返り値

SDO_Model_Property の名前を返します。

SDO_Model_Property::getType

(No version information available, might be only in CVS)

SDO_Model_Property::getType — プロパティの SDO_Model_Type を取得する

説明

SDO_Model_Type ***SDO_Model_Property::getType*** *(void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

プロパティの SDO_Model_Type を取得します。 SDO_Model_Type には、型の名前や名前空間 URI、プリミティブデータ型かどうか など、プロパティの型に関する情報が含まれます。

パラメータ

なし。

返り値

プロパティ型に関する情報を含む SDO_Model_Type を返します。

SDO_Model_Property::isContainment

(No version information available, might be only in CVS)

SDO_Model_Property::isContainment — プロパティが包含関係を定義しているかどうかを調べる

説明

bool **SDO_Model_Property::isContainment** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このプロパティが包含関係を表すものかどうかを調べます。 包含関係を定義したものである場合に **TRUE**、 単なる参照である場合に **FALSE** を返します。

パラメータ

なし。

返り値

プロパティが包含関係を定義したものである場合に **TRUE**、 単なる参照である場合に **FALSE** を返します。

SDO_Model_Property::isMany

(No version information available, might be only in CVS)

SDO_Model_Property::isMany — プロパティが複数の値を持つかどうかを調べる

説明

bool **SDO_Model_Property::isMany** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

プロパティが複数の値を持つかどうかを調べます。 複数の値を持つプロパティであった場合に **TRUE**、それ以外の場合に **FALSE** を返します。

パラメータ

なし。

返り値

複数の値を持つプロパティであった場合に **TRUE**、それ以外の場合に **FALSE** を返します。

SDO_Model_ReflectionDataObject::__construct

(No version information available, might be only in CVS)

SDO_Model_ReflectionDataObject::__construct — SDO_Model_ReflectionDataObject を作成する

説明

SDO_Model_ReflectionDataObject **SDO_Model_ReflectionDataObject::__construct** (SDO_DataObject $data_object)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject へのリフレクションのための SDO_Model_ReflectionDataObject を作成します。 SDO_DataObject へのリフレクションにより、モデル自身の情報への アクセスが可能になります。モデルに含まれる情報には、 データオブジェクトの型・シーケンス型（プロパティを超えた並び順を 保持する）かどうか・オープン型（個々のインスタンスがモデルを 拡張できる）かどうかなどがあります。また、データオブジェクトの プロパティやデフォルト値などの情報も含みます。

パラメータ

data_object

リフレクションの対象となる SDO_DataObject。

返り値

何も値を返しません。

SDO_Model_ReflectionDataObject::export

(No version information available, might be only in CVS)

SDO_Model_ReflectionDataObject::export — SDO_DataObject を表す文字列を取得する

説明

mixed **SDO_Model_ReflectionDataObject::export** (SDO_Model_ReflectionDataObject $rdo [, bool $return])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject を表す文字列を取得します。 デフォルトでは標準出力に出力されますが、パラメータ return に **TRUE** を指定すると、結果を文字列で返します。

パラメータ

rdo

SDO_Model_ReflectionDataObject。

return

TRUE の場合は結果を文字列で返します。 それ以外の場合は標準出力に出力します。

返り値

return パラメータに TRUE が設定されている場合は出力内容、 それ以外の場合は NULL を返します。

SDO_Model_ReflectionDataObject::getContainmentProperty

(No version information available, might be only in CVS)

SDO_Model_ReflectionDataObject::getContainmentProperty — データオブジェクトとの包含関係を定義しているプロパティを取得する

説明

SDO_Model_Property **SDO_Model_ReflectionDataObject::getContainmentProperty** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject を含んでいる SDO_Model_Property を取得します。 このメソッドは、リフレクション対象との包含関係を定義している 親オブジェクトのプロパティをたどる際に使用可能です。

パラメータ

なし。

返り値

親オブジェクトが SDO_DataObject を参照しているプロパティを表す SDO_Model_Property、あるいはルート SDO_DataObject の場合は **NULL** を返します。

SDO_Model_ReflectionDataObject::getInstanceProperties

(No version information available, might be only in CVS)

SDO_Model_ReflectionDataObject::getInstanceProperties — SDO_DataObject のインスタンスプロパティを取得する

説明

array **SDO_Model_ReflectionDataObject::getInstanceProperties** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject のインスタンスプロパティを取得します。 インスタンスプロパティには、データオブジェクトの型で定義されている プロパティのほかに（もしデータオブジェクトがオープン型の場合は）オープン型からのプロパティが含まれます。

パラメータ

なし。

返り値

SDO_Model_Property オブジェクトの配列を返します。

SDO_Model_ReflectionDataObject::getType

(No version information available, might be only in CVS)

SDO_Model_ReflectionDataObject::getType — SDO_DataObject の SDO_Model_Type を取得する

説明

SDO_Model_Type **SDO_Model_ReflectionDataObject::getType** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject の SDO_Model_Type を返します。SDO_Model_Type には 名前空間 URI・型の名前・プリミティブデータ型かどうかなど データオブジェクトの型についてのすべての情報が含まれます。

パラメータ

なし。

返り値

SDO_DataObject の SDO_Model_Type を返します。

SDO_Model_Type::getBaseType

(No version information available, might be only in CVS)

SDO_Model_Type::getBaseType — この型の基底型を取得する

説明

SDO_Model_Type **SDO_Model_Type::getBaseType** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この型の基底型を取得します。この型が別の型を派生したものである場合に SDO_Model_Type、それ以外の場合に **NULL** を返します。基底型が存在するのは、 例えば XML スキーマで 以下のようにして他の型を継承した型を定義した場合です。

```
<extension base="...">
```

パラメータ

なし。

返り値

この型が別の型を派生したものである場合に SDO_Model_Type、 それ以外の場合に **NULL** を返します。

SDO_Model_Type::getName

(No version information available, might be only in CVS)

SDO_Model_Type::getName — 型の名前を取得する

説明

string **SDO_Model_Type::getName** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

型の名前を返します。型の名前および名前空間 URI を組み合わせることで 型を特定します。

パラメータ

なし。

返り値

型の名前を返します。

SDO_Model_Type::getNamespaceURI

(No version information available, might be only in CVS)

SDO_Model_Type::getNamespaceURI — 型の名前空間 URI を取得する

説明

string **SDO_Model_Type::getNamespaceURI** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

型の名前空間 URI を返します。型の名前および名前空間 URI を組み合わせることで 型を特定します。

パラメータ

なし。

返り値

型の名前空間 URI を返します。

SDO_Model_Type::getProperties

(No version information available, might be only in CVS)

SDO_Model_Type::getProperties — 型で定義されている SDO_Model_Property オブジェクトを取得する

説明

array **SDO_Model_Type::getProperties** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_Model_Type で定義されているプロパティを指す SDO_Model_Property オブジェクトの配列を取得します。 個々の SDO_Model_Property にはプロパティ名、デフォルト値 などといった情報が含まれます。

パラメータ

なし。

返り値

SDO_Model_Property オブジェクトの配列を返します。

SDO_Model_Type::getProperty

(No version information available, might be only in CVS)

SDO_Model_Type::getProperty — 型の SDO_Model_Property を取得する

説明

SDO_Model_Property **SDO_Model_Type::getProperty** (mixed $identifier)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この型の SDO_Model_Property を、 プロパティのインデックスや名前を指定して取得します。

パラメータ

identifier

プロパティのインデックスあるいは名前。

返り値

SDO_Model_Property を返します。

SDO_Model_Type::isAbstractType

(No version information available, might be only in CVS)

SDO_Model_Type::isAbstractType — この SDO_Model_Type が抽象データ型であるかどうかを調べる

説明

bool **SDO_Model_Type::isAbstractType** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この SDO_Model_Type が抽象データ型であるかどうかを調べます。 抽象データ型である場合は TRUE を返します。この場合、 この型の SDO_DataObject を直接インスタンスかすることはできず、 そこから継承した他の型を使用することになります。

パラメータ

なし。

返り値

この型が抽象データ型である場合に TRUE、 それ以外の場合に FALSE を返します。

SDO_Model_Type::isDataType

(No version information available, might be only in CVS)

SDO_Model_Type::isDataType — この SDO_Model_Type がプリミティブなデータ型であるかどうかを調べる

説明

bool **SDO_Model_Type::isDataType** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この SDO_Model_Type がプリミティブなデータ型であるかどうかを調べます。 この型がプリミティブなデータ型である場合に TRUE、それ以外の場合に FALSE を返します。

パラメータ

なし。

返り値

この型がプリミティブなデータ型である場合に TRUE、それ以外の場合に FALSE を返します。

SDO_Model_Type::isInstance

(No version information available, might be only in CVS)

SDO_Model_Type::isInstance — SDO_DataObject が、この SDO_Model_Type のインスタンスであるかどうかを調べる

説明

bool **SDO_Model_Type::isInstance** *(SDO_DataObject $data_object)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SDO_DataObject が、この SDO_Model_Type のインスタンスであるかどうかを 調べます。 指定した SDO_DataObject が、この SDO_Model_Type あるいはその派生クラスのインスタンスである場合に TRUE、それ以外の場合に FALSE を返します。

パラメータ

data_object

調べようとしている SDO_DataObject。

返り値

指定した SDO_DataObject が、この SDO_Model_Type あるいはその派生クラスのインスタンスである場合に TRUE、それ以外の場合に FALSE を返します。

SDO_Model_Type::isOpenType

(No version information available, might be only in CVS)

SDO_Model_Type::isOpenType — この型がオープン型であるかどうかを調べる

説明

bool **SDO_Model_Type::isOpenType** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この型がオープン型であるかどうかを調べます。 この型がオープン型である場合に TRUE、それ以外の場合に FALSE を返します。 オープン型である SDO_DataObject は、定義されていない型のプロパティを 保持することができます。この機能は、XML ドキュメントのスキーマで

<xsd:any>

要素が定義されているような場合をサポートするために 使用されます。

パラメータ

なし。

返り値

この型がオープン型である場合に TRUE、それ以外の場合に FALSE を返します。

SDO_Model_Type::isSequencedType

(No version information available, might be only in CVS)

SDO_Model_Type::isSequencedType — この型がシーケンス型であるかどうかを調べる

説明

bool **SDO_Model_Type::isSequencedType** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この型がシーケンス型であるかどうかを調べる この型がシーケンス型である場合に TRUE、それ以外の場合に FALSE を返します。 シーケンス型は複数のプロパティをまたがって要素の順序を管理することが 可能で、非構造化テキストを含むことができます。シーケンス型についての 詳細な情報は、シーケンスデータオブジェクトの 使用 を参照ください。

パラメータ

なし。

返り値

この型がシーケンス型である場合に TRUE、それ以外の場合に FALSE を返します。

SDO_Sequence::getProperty

(No version information available, might be only in CVS)

SDO_Sequence::getProperty — 指定したシーケンスインデックスに対応するプロパティを返す

説明

SDO_Model_Property **SDO_Sequence::getProperty** (int $sequence_index)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

指定したシーケンスインデックスに対応するプロパティを返します。

パラメータ

sequence_index

シーケンス内での要素の位置。

返り値

SDO_Model_Property を返します。 NULL が返された場合は、そのシーケンス要素がプロパティに属していない、 すなわち非構造化テキストであることを意味します。

エラー / 例外

SDO_IndexOutOfBoundsException

シーケンスのインデックスがゼロより小さい、 あるいはシーケンスのサイズより大きい場合にスローされます。

SDO_Sequence::insert

(No version information available, might be only in CVS)

SDO_Sequence::insert — シーケンスに挿入する

説明

void ***SDO_Sequence::insert*** *(mixed $value [, int $sequenceIndex [, mixed $propertyIdentifier]])*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

シーケンス内の指定した位置に、新しい要素を挿入します。 シーケンスのそれ以降の項目は、ひとつ後ろにずれます。

パラメータ

value

挿入される新しい値。 プリミティブ型あるいは SDO_DataObject のいずれかです。

sequenceIndex

新しい要素を挿入する位置。デフォルトは NULL で、 この場合は新しい値がシーケンスの最後に追加されます。

propertyIdentifier

プロパティのインデックス、プロパティ名あるいは SDO_Model_Property。 シーケンスに対応する SDO_DataObject プロパティを指定するために使用します。NULL は、非構造化テキストを意味します。

返り値

なし。

エラー / 例外

SDO_IndexOutOfBoundsException

シーケンスのインデックスがゼロより小さい場合、あるいは シーケンスのサイズより大きい場合にスローされます。

SDO_InvalidConversionException

新しい値の型が、指定されたデータオブジェクトプロパティの型に 対応していない場合にスローされます。

SDO_Sequence::move

(No version information available, might be only in CVS)

SDO_Sequence::move — シーケンス上の指定した位置に項目を移動する

説明

void ***SDO_Sequence::move*** *(int $toIndex, int $fromIndex)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

シーケンス内での項目の位置を変更します。 SDO_DataObject のプロパティの値には変更を加えません。

パラメータ

toIndex

移動先のシーケンスインデックス。この値がゼロより小さかったり シーケンスのサイズより大きかったりした場合は、項目は最後に追加されます。

fromIndex

移動元のシーケンスインデックス。

返り値

なし。

エラー / 例外

SDO_IndexOutOfBoundsException

シーケンスのインデックス fromIndex がゼロより小さい場合、あるいは シーケンスのサイズより大きい場合にスローされます。

目次

- SDO_DAS_DataFactory::getDataFactory — データファクトリのインスタンスを取得する
- SDO_DAS_DataObject::getChangeSummary — データオブジェクトの変更内容を取得する
- SDO_DAS_Setting::getListIndex — 複数の値を持つプロパティのインデックスを取得する
- SDO_DAS_Setting::getPropertyIndex — 変更されたプロパティのプロパティインデックスを取得する
- SDO_DAS_Setting::getPropertyName — 変更されたプロパティのプロパティ名を取得する
- SDO_DAS_Setting::getValue — 変更されたプロパティの変更前の値を取得する
- SDO_DAS_Setting::isSet — 変更前にプロパティが設定されていたかどうかを調べる
- SDO_DataFactory::create — SDO_DataObject を作成する
- SDO_DataObject::clear — SDO_DataObject のプロパティを消去する
- SDO_DataObject::createDataObject — 新しい子 SDO_DataObject を作成する
- SDO_DataObject::getContainer — データオブジェクトのコンテナを取得する
- SDO_DataObject::getSequence — データオブジェクトのシーケンスを取得する
- SDO_DataObject::getTypeName — データオブジェクトの型の名前を取得する
- SDO_DataObject::getTypeNamespaceURI — このデータオブジェクトの型の名前空間 URI を取得する
- SDO_Exception::getCause — 例外の原因を取得する
- SDO_List::insert — リストに挿入する
- SDO_Model_Property::getContainingType — このプロパティを含む SDO_Model_Type を取得する
- SDO_Model_Property::getDefault — プロパティのデフォルト値を取得する
- SDO_Model_Property::getName — SDO_Model_Property の名前を取得する
- SDO_Model_Property::getType — プロパティの SDO_Model_Type を取得する
- SDO_Model_Property::isContainment — プロパティが包含関係を定義しているかどうかを調べる
- SDO_Model_Property::isMany — プロパティが複数の値を持つかどうかを調べる
- SDO_Model_ReflectionDataObject::__construct — SDO_Model_ReflectionDataObject を作成する
- SDO_Model_ReflectionDataObject::export — SDO_DataObject を表す文字列を取得する
- SDO_Model_ReflectionDataObject::getContainmentProperty — データオブジェクトとの包含関係を定義しているプロパティを取得する
- SDO_Model_ReflectionDataObject::getInstanceProperties — SDO_DataObject のインスタンスプロパティを取得する
- SDO_Model_ReflectionDataObject::getType — SDO_DataObject の SDO_Model_Type を取得する
- SDO_Model_Type::getBaseType — この型の基底型を取得する
- SDO_Model_Type::getName — 型の名前を取得する
- SDO_Model_Type::getNamespaceURI — 型の名前空間 URI を取得する
- SDO_Model_Type::getProperties — 型で定義されている SDO_Model_Property オブジェクトを取得する
- SDO_Model_Type::getProperty — 型の SDO_Model_Property を取得する
- SDO_Model_Type::isAbstractType — この SDO_Model_Type が抽象データ型であるかどうかを調べる
- SDO_Model_Type::isDataType — この SDO_Model_Type がプリミティブなデータ型であるかどうかを調べる
- SDO_Model_Type::isInstance — SDO_DataObject が、この SDO_Model_Type のインスタンスであるかどうかを調べる
- SDO_Model_Type::isOpenType — この型がオープン型であるかどうかを調べる
- SDO_Model_Type::isSequencedType — この型がシーケンス型であるかどうかを調べる
- SDO_Sequence::getProperty — 指定したシーケンスインデックスに対応するプロパティを返す
- SDO_Sequence::insert — シーケンスに挿入する
- SDO_Sequence::move — シーケンス上の指定した位置に項目を移動する

SDO XML データアクセスサービス関数

導入

SDO 用の XML データアクセスサービスを使用するには、 SDO の背景となる概念を理解しておく必要があります。例えば データグラフ、データオブジェクト、XPath とプロパティの表現法などを理解しておきましょう。 これらの概念になじみがない場合は、まず最初に SDO の節を見るとよいでしょう。

XML DAS の役割は、アプリケーションと XML データソースとの間のデータ移動を行うことです。 XML データソースは、ファイルあるいは URL のどちらでも使用可能です。 SDO の作成およびメンテナンスは、常にモデルに基づいて行われます。 型の名前やその型が保持するプロパティの名前などがモデルに定義されています。 XML から取得するデータの場合は、XML スキーマ言語（xsd ファイル）で記述されたスキーマファイルに基づいて SDO モデルが作成されます。 XMLDAS を初期化する際に、このスキーマファイルが作成メソッドに渡されることが多くなります。 » SDO 2.0 仕様で、XML と SDO の間での型の対応が定義されています。 PHP の実装では、制限がいくつかあり、 仕様に書かれている内容がすべて実装されているわけではありません。 この制限については後でまとめます。

要件

SDO XML データアクセスサービスには PHP 5.1.0 以降が必要です。 また、SDO 拡張モジュールに同梱されているものであるため、 SDO がインストールされていなければなりません。 インストール方法の詳細については、 SDO のインストール手順 を参照ください。

インストール手順

XML データアクセスサービスは、 SDO 拡張モジュール の一部として提供されており、同時にインストールされます。 SDO のインストール手順 を参照ください。

例

以下の例のいくつかは、 SDO のドキュメントで説明した letter の例を使用しています。 この例では、litter の XML スキーマが letter.xsd に、そしてインスタンスが letter.xml に保存されていることを想定しています。 これらの 2 つのファイルは次のようになります。

```
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:letter="http://letterSchema"
    targetNamespace="http://letterSchema">
    <xsd:element  name="letters"  type="letter:FormLetter"/>
    <xsd:complexType  name="FormLetter"  mixed="true">
      <xsd:sequence>
          <xsd:element  name="date"  minOccurs="0"  type="xsd:string"/>
          <xsd:element  name="firstName"  minOccurs="0"  type="xsd:string"/>
          <xsd:element  name="lastName"  minOccurs="0"  type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>

<letter:letters  xmlns:letter="http://letterSchema">
   <date>March  1,  2005</date>
   Mutual  of  Omaha
   Wild  Kingdom,  USA
   Dear
    <firstName>Casy</firstName>
    <lastName>Crocodile</lastName>
   Please  buy  more  shark  repellent.
   Your  premium  is  past  due.
</letter:letters>
```

Example#1 XML ドキュメントの読み込み・変更そして保存

次の例では、XML ドキュメントをファイルから読み込んでその内容を変更し、 もう一度書き戻す方法を示します。

```
<?php
/**
 * XML ドキュメントの読み込み・更新そして保存
 */
try {
    $xmldas  = SDO_DAS_XML::create("letter.xsd");
    $document  = $xmldas->loadFile("letter.xml");
    $root_data_object  = $document->getRootDataObject();
    $root_data_object->date = "September  03,  2004";
    $root_data_object->firstName  = "Anantoju";
    $root_data_object->lastName  = "Madhu";
    $xmldas->saveFile($document, "letter-out.xml");
    echo "New  file  has  been  written:¥n";
    print file_get_contents("letter-out.xml");
} catch (SDO_Exception  $e) {
    print($e->getMessage());
}
?>
```

まず最初に、**SDO_DAS_XML::create()** メソッドで XML DAS のインスタンスを取得します。 このメソッドは、SDO_DAS_XML クラスの静的メソッドです。パラメータとして xsd の場所を渡します。 スキーマを指定して XML DAS のインスタンスを初期化したら、 **loadFile()** メソッドを使用してインスタンスドキュメントを読み込むことができます。 もし文字列から XML インスタンスドキュメントを読み込みたいのなら、 **loadString()** メソッドを使用することも可能です。 インスタンスドキュメントの読み込みに成功したら SDO_DAS_XML_Document 型のオブジェクトが返されます。このオブジェクトに対して **getRootDataObject()** メソッドをコールすると、SDO データグラフのルートとなる SDO データオブジェクトが取得できます。 その後、SDO に対する操作によってグラフを変更します。 この例では date、 firstName そして lastName のプロパティを変更しています。 その後で **saveFile()** メソッドを使用して、 ドキュメントをファイルシステムに書き戻します。 saveFile メソッドにはオプションで整数型の引数を指定することができ、 これが設定されていると、XML DAS が結果の XML の書式を整形します。 指定された整数値のぶんだけ、ドキュメントの字下げが行われます。

この例の結果、以下のような内容が letter-out.xml に書き出されます。

```
<?xml  version="1.0"  encoding="UTF-8"?>
<FormLetter  xmlns="http://letterSchema"  xsi:type="FormLetter"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <date>September  03,  2004</date>
   Mutual  of  Omaha
   Wild  Kingdom,  USA
   Dear
    <firstName>Anantoju</firstName>
    <lastName>Madhu</lastName>
   Please  buy  more  shark  repellent.
   Your  premium  is  past  due.
</FormLetter>
```

Example#2 新しい XML ドキュメントの作成

さきほどの例ではファイルからドキュメントを読み込みました。 この例では、メモリ上で SDO データグラフを作成する方法を示します。 この例では、データグラフが XML 文字列として保存されます。 さらに、letter の中に構造化されたデータとされていないデータの両方が含まれていることから、 シーケンス API を使用してプロパティを割り当て、データグラフを作成します。

```
<?php
/**
 * スクラッチからの XML ドキュメントの作成
 */
try {
    $xmldas  = SDO_DAS_XML::create("letter.xsd");
    try {
        $doc  = $xmldas->createDocument();
        $rdo  = $doc->getRootDataObject();
        $seq  = $rdo->getSequence();
        $seq->insert("April  09,  2005", NULL, 'date');
        $seq->insert("Acme  Inc.  ", NULL, NULL);
        $seq->insert("United  Kingdom.  ");
        $seq->insert("Dear", NULL, NULL);
        $seq->insert("Tarun", NULL, "firstName");
        $seq->insert("Nayaraaa", NULL, "lastName");
        $rdo->lastName  = "Nayar";
        $seq->insert("Please  note  that  your  order  number  ");
        $seq->insert(12345);
        $seq->insert("  has  been  dispatched  today.  Thanks  for  your  business  with  us.");
        print($xmldas->saveString($doc));
    } catch (SDO_Exception  $e) {
        print($e);
    }
```

```
} catch (SDO_Exception $e) {
    print("Problem creating an XML document: " . $e->getMessage());
}
?>
```

XML DAS の **createDocument()** メソッドは、空のドキュメント要素に対応するルートデータオブジェクトを ひとつだけ含むドキュメントオブジェクトを返します。 ドキュメント要素の要素名はスキーマファイルから取得します。 ドキュメント要素をはっきり特定できない場合（例えば同一の XML DAS に対して複数のスキーマを読み込むことも可能です）は、 要素名に加えて名前空間 URI を **createDocument()** メソッドに渡します。

この例は、次のような内容を出力します（読みやすくするために適宜改行を挿入しています）。

```
<?xml version="1.0" encoding="UTF-8"?>
<FormLetter xmlns="http://letterSchema" xsi:type="FormLetter"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<date>April 09, 2005</date>
Acme Inc. United Kingdom.
Dear
<firstName>Tarun</firstName>
<lastName>Nayar</lastName>
Please note that your order number 12345 has been
dispatched today. Thanks for your business with us.
</FormLetter>
```

Example#3 XML ドキュメントのプロパティの設定

3 番目のこの例では、ドキュメントに対して XML のバージョンとエンコーディングを設定する方法を示します。 これらは、XML を書き出す際に使用されます。 XML 宣言が不要な場合（XML を文字列として作成してどこかに埋め込むなどの場合がそうでしょう）は、 **setXMLDeclaration()** メソッドを使用して宣言を抑制することができます。

```
<?php
/**
 * XML 宣言を制御するためのコール方法
 */
   $xmldas = SDO_DAS_XML::create("letter.xsd");
   $document = $xmldas->loadFile("letter.xml");
   $document->setXMLVersion("1.1");
   $document->setEncoding("ISO-8859-1");
    print($xmldas->saveString($document));
?>
```

XML のバージョンおよびエンコーディングは、 XML ドキュメントの先頭にあたる XML 宣言で設定されます。

```
<?xml version="1.1" encoding="ISO-8859-1"?>
.../...
```

Example#4 オープン型の使用

4 番目の例では、SDO オープン型および **createDataObject()** メソッドの使用法を説明します。 この例では、次のふたつのスキーマを使用します。

```
<schema
    xmlns="http://www.w3.org/2001/XMLSchema">

   <element name="jungle">
     <complexType>
       <sequence>
           <any minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
     </complexType>
   </element>

</schema>
```

定義の中に any 要素が存在することに注目しましょう。 この最初のスキーマでは、複合型 jungle を定義しています。その内容は、その他の任意の型のシーケンスです。 この例で使用している方は、2 番目のスキーマファイルで定義されています。

```
<schema xmlns= "http://www.w3.org/2001/XMLSchema">

     <complexType name="snakeType">
       <sequence>
          <element name= "name" type="string"/>
           <element name= "length" type="positiveInteger" />
       </sequence>
     </complexType>

     <complexType name="bearType">
       <sequence>
          <element name= "name" type="string"/>
           <element name= "weight" type="positiveInteger" />
       </sequence>
     </complexType>

     <complexType name="pantherType">
       <sequence>
          <element name= "name" type="string"/>
          <element name= "colour" type="string" />
       </sequence>
     </complexType>

</schema>
```

これらのふたつのスキーマファイルを使用する PHP コードの例は、次のようになります。

```
<?php

/**
 * オープン型および addTypes() メソッドの使用法
 */

$xmldas = SDO_DAS_XML::create();
```

```
$xmldas->addTypes("jungle.xsd"); //  これはオープン型です。つまり、
                                 // "any" 型を使用できることが
                                 // xsd ファイルで指定されています
$xmldas->addTypes('animalTypes.xsd');

$baloo              = $xmldas->createDataObject('','bearType');
$baloo->name        = "Baloo";
$baloo->weight      = 800;

$bagheera           = $xmldas->createDataObject('','pantherType');
$bagheera->name     = "Bagheera";
$bagheera->colour   = 'inky  black';

$kaa                = $xmldas->createDataObject('','snakeType');
$kaa->name          = "Kaa";
$kaa->length        = 25;

$document           = $xmldas->createDocument();
$do                 = $document->getRootDataObject();
$do->bear           = $baloo;
$do->panther        = $bagheera;
$do->snake          = $kaa;

print($xmldas->saveString($document,2));

?>
```

これらのふたつのスキーマファイルは、 最初の **create()** および **addTypes()** メソッドで XML DAS に読み込まれます。 **createDataObject()** メソッドを使用して、 3 つに分かれたデータオブジェクトを作成します。それぞれについて、 名前空間 URI および型名が **createDataObject()** メソッドに渡されます。この例では、スキーマで名前空間が使用されていないため、 名前空間 URI は空の文字列です。bear (クマ)、panther (ヒョウ)、snake (ヘビ) を表す 3 つのデータオブジェクトが作成された後で、 **createDocument()** メソッドでドキュメントオブジェクトが作成されます。 この例の場合、ドキュメントの要素が何になるかは明確です。 2 番目のスキーマに含まれているのは複合型の定義のみであり、 ドキュメント要素は最初のスキーマに含まれているグローバルな jungle 要素だけだからです。 このドキュメントは 1 つのルートデータオブジェクトを保持しており、 これが空のドキュメント要素 jungle に対応しています。 この要素はオープン型なので、お好みのプロパティを追加することができます。最初に代入が行われている部分が $do->bear で、 ここではルートデータオブジェクトにプロパティ bear が追加されています。それ以降の 2 つの代入についても同様です。 **saveString()** メソッドでドキュメントを書き出した結果は、 以下のようになります。

```
<?xml  version="1.0"  encoding="UTF-8"?>
<jungle  xsi:type="jungle"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <bear  xsi:type="bearType">
      <name>Baloo</name>
      <weight>800</weight>
   </bear>
   <panther  xsi:type="pantherType">
      <name>Bagheera</name>
      <colour>inky  black</colour>
   </panther>
   <snake  xsi:type="snakeType">
      <name>Kaa</name>
      <length>25</length>
   </snake>
</jungle>
```

Example#5 ドキュメントから何が得られるのかを知る

この例で説明しているのは、 XML Document オブジェクトから要素名および要素の名前空間を知る方法、 XML データオブジェクトのルートデータオブジェクトから SDO 型および名前空間を知る方法、 そしてそれらがお互いどのように関連しているかといった内容です。 これを理解するのは少し難しいかもしれません。というのも、ここでは 4 つのメソッドがコールされているからです。それらのうちの 2 つは Document オブジェクトに対するコールで、 残りの 2 つはルートデータオブジェクトを含む任意のデータオブジェクトに対するコールです。 XML モデルから SDO モデルを構築する際に定義されている規則のため、 もし今扱っているデータオブジェクトが ドキュメントのルートオブジェクトに対応するものならば、 これらの 4 つのメソッドコールから返される可能性のある値は 3 つだけです。

ドキュメントに対してコールされる 2 つのメソッドは、 **getRootElementName()** および **getRootEelementURI()** です。 これらはそれぞれ、要素名およびドキュメント要素の名前空間を返します。

任意のデータオブジェクトに対してコールされる 2 つのメソッドは、 **getTypeName()** および **getTypeNamespaceURI()** です。 これらはそれぞれ、SDO の型名およびデータオブジェクトの型の名前空間を返します。

ドキュメントオブジェクトに対して **getRootElementURI()** をコールした際の返り値は、ルートデータオブジェクトに対して **getNamespaceURI()** をコールした際の返り値と常に同じです。 本質的に、この情報はすべてスキーマファイルの最初の数行から得られるものです。 そこには 3 種類の情報が含まれています。 詳しく説明するために、もう一度ここで letter.xsd の最初の数行を示します。

```
<xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:letter="http://letterSchema"
               targetNamespace="http://letterSchema">

   <xsd:element  name="letters"  type="letter:FormLetter"/>

   <xsd:complexType  name="FormLetter"  mixed="true">
     .../...
```

重要な 3 つの値は以下のとおりです。

- letters がドキュメント要素の名前です。
- FormLetter がドキュメント要素の複合型の名前です。 これはまた、ルートデータオブジェクトの SDO 型の名前でもあります。
- http://letterSchema がドキュメント要素の属する名前空間です。 これはまた、ルートデータオブジェクトの SDO 型の名前空間 URI でもあります。

XML-SDO 間の対応の規則により、 SDO モデルがスキーマファイルから作成される場合には ルート要素の型名および SDO 型の名前空間 URI は それが存在するドキュメント要素の複合型から取得されます。 したがって、この例ではルートデータオブジェクトの型名は FormLetter となります。 ドキュメント要素の複合型の定義が別に分かれておらずインラインで定義されており、 名前がつけられていない場合は、SDO の型名は要素名と同じになります。

次のプログラムは、letter ドキュメントを読み込んでから 4 つのコールの返り値を調べます。

```
<?php
/**
 * ドキュメントおよびドキュメント要素から得られる情報を調べます。
```

```
 * 4 種類のコールがあるため、理解するのは少し難しいでしょう。
 * ドキュメントに対するコールが 2 種類、
 * ルートデータオブジェクトおよびそのモデルに対するコールが 2 種類となります。
 * SDO-XML の対応規則および SDO モデルが XML モデルから作成されている
 * ということから、これらの 4 つのコールから得られる値は 3 つだけです。
 *  常に $document->getRootElementURI() == (ルートデータオブジェクトの型)->namespaceURI
 * となります。
 * 本質的に、これらはすべて xsd の最初の数行から得られるものです。
 *  <xsd:schema  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 *     xmlns:letter="http://letterSchema"
 *     targetNamespace="http://letterSchema">
 *     <xsd:element  name="letters"  type="letter:FormLetter"/>
 */

$xmldas  = SDO_DAS_XML::create("letter.xsd");
$document  = $xmldas->loadFile("letter.xml");
$root_do  = $document->getRootDataObject();

/**
 * "root element name" は、ドキュメント要素の要素名です。
 * この場合は 'letters' となります。
 * これは xsd のドキュメント要素の 'name' 属性と一致し、
 * xml の要素名とも一致します。
 */
echo "ドキュメント要素の名前は  "  . $document->getRootElementName() . "  です。\n";
assert($document->getRootElementName() == 'letters'); //  ドキュメントのプロパティ

/**
 * "root element URI" は、ドキュメント要素の要素名の名前空間です。
 * この場合は 'http://letterSchema' となります。
 * というのも 'letters' がこの名前空間にあるからです。
 * これは xsd から得られ、xml から取得した名前空間と一致します。
 */
echo "ドキュメント要素の名前空間は  "  . $document->getRootElementURI() . "  です。\n";
assert($document->getRootElementURI() == 'http://letterSchema'); //  ドキュメントのプロパティ

/**
 * 型名は SDO モデルから取得します。
 * XML-SDO の対応規則では、これは下のいずれかです。
 *    complexType が存在する場合は、その名前 (今回はこちらです)
 *    complexType がない場合は、ドキュメント要素の名前
 * これは xsd から得られます。
 */
echo "ルートデータオブジェクトの型名は  "  . $root_do->getTypeName() . "  です。\n";
assert($root_do->getTypeName() == 'FormLetter');

/**
 * 型の namespaceURI は SDO モデルから取得します。
 * XML-SDO の対応規則では、これは常にドキュメント要素の名前空間 URI と同じ
 * であることが保証されています。
 */
echo "ルートデータオブジェクトの  namespaceURI  は  "  . $root_do->getTypeNamespaceURI() . "  です。\n";
assert($root_do->getTypeNamespaceURI() == 'http://letterSchema');

?>
```

このプログラムの出力は次のようになります。

```
ドキュメント要素の名前は  letters  です。
ドキュメント要素の名前空間は  http://letterSchema  です。
ルートデータオブジェクトの型名は  FormLetter  です。
ルートデータオブジェクトの  namespaceURI  は  http://letterSchema  です。
```

Example#6 SDO モデルの表示

XML DAS に読み込まれた型やプロパティの情報を見るための簡単な手段が提供されています。 php の "print" あるいは "echo" を使用すると、型およびプロパティの情報が出力されます。

```
<?php
/**
 * モデルの情報の出力
 */

$xmldas  = SDO_DAS_XML::create("letter.xsd");
print  $xmldas;

?>
```

このプログラムの出力は次のようになります。

```
object(SDO_XML_DAS)#1  {
18  types  have  been  defined.  The  types  and  their  properties  are::
1.  commonj.sdo:BigDecimal
2.  commonj.sdo:BigInteger
3.  commonj.sdo:Boolean
4.  commonj.sdo:Byte
5.  commonj.sdo:Bytes
6.  commonj.sdo:ChangeSummary
7.  commonj.sdo:Character
8.  commonj.sdo:DataObject
9.  commonj.sdo:Date
10.  commonj.sdo:Double
11.  commonj.sdo:Float
12.  commonj.sdo:Integer
13.  commonj.sdo:Long
14.  commonj.sdo:Short
15.  commonj.sdo:String
16.  commonj.sdo:URI
17.  http://letterSchema:FormLetter
      - date (commonj.sdo:String)
      - firstName (commonj.sdo:String)
```

```
      - lastName (commonj.sdo:String)
18.  http://letterSchema:RootType
      - letters (http://letterSchema:FormLetter)
```

定義済みクラス

XML DAS は、主に 2 つのクラスから成り立っています。 ひとつめは SDO_DAS_XML で、XML ソースからデータを取得したり データを書き戻したりする際に使用するクラスです。 もうひとつが SDO_DAS_XML_Document クラスで、 これが XML ドキュメント内のデータを表します。

これ以外に、xsd や xml ファイルの検索・パース中にエラーが発生した場合にスローされる 例外クラスがいくつかあります。

SDO_DAS_XML

これが XML DAS の中心となるクラスです。 xml ソースからデータを取得したり、データを書き戻す際に使用します。 xml ファイルの読み込みや書き込み以外のメソッドもあります。

メソッド

- create は SDO_DAS_XML クラスの静的メソッドです。 SDO_DAS_XML オブジェクトを作成するために使用します。
- addTypes は **create()** と同じような動作をしますが、 これはすでに作成済みの XML DAS に対して後からスキーマファイルを追加します。
- createDataObject は、指定した型の SDO データオブジェクトを作成するために使用します。
- createDocument は、XML ドキュメントオブジェクトをスクラッチから作成するために使用します。
- loadFile は、xml インスタンスドキュメントをファイルから読み込みます。 ローカルファイルシステム上、あるいはリモートホスト上のどちらのファイルも読み込み可能です。
- loadString は上のメソッドと同じですが、これは xml インスタンスドキュメントを文字列から読み込みます。
- saveFile は SDO_DAS_XML_Document オブジェクトを xml ファイルに保存します。
- saveString は SDO_DAS_XML_Document オブジェクトを xml 文字列に保存します。

SDO_DAS_XML_Document

このクラスの使用目的は、ドキュメント要素の名前および名前空間を取得すること、 そしてドキュメントのルートデータオブジェクトを取得することです。 最後に、ドキュメントの出力時に XML のバージョンおよびエンコーディングを指定することもできます。

メソッド

- getRootDataObject は、ルート DataObject を取得します。
- getRootElementName は、ルート DataObject の名前を取得します。
- getRootElementURI は、ルート DataObject の URI を取得します。
- setEncoding は、指定した値をエンコーディング文字列に設定します。
- setXMLDeclaraion は、xml 宣言を設定/解除します。
- setXMLVersion は、指定した値を xml バージョンに設定します。

SDO_DAS_XML_ParserException

SDO_Exception のサブクラスです。 xsd/xml ファイルの読み込み中にパースエラーが発生した場合にスローされます。

SDO_DAS_XML_FileException

SDO_Exception のサブクラスです。 ファイルからデータを読み込むメソッドで、 ファイルが見つからなかった場合にスローされます。

SDO 2.0 仕様と比較した場合の制限事項

» SDO 2.0 仕様 で、XML と SDO の間での型の対応が定義されています。 Java SDO では、この対応は XMLHelper で実装されています。 PHP 用の SDO では、この対応が XML データアクセスサービスで実装されています。 XML DAS の実装には、SDO 2.0 仕様で定義されている対応に対していくつか制限があります。 制限事項の詳細は以下のとおりです。

XML の単純な型

1. Simple Type with sdoJava:instanceClass - PHP では同等の機能は提供されていません。
2. Simple Type with sdoJava:extendedInstanceClass - PHP では同等の機能は提供されていません。
3. Simple Type with list of itemType.
4. Simple Type with union.

XML の複合型

1. Complex Type with sdo:aliasName - PHP では SDO 型のエイリアスをサポートしていません。

XSD の属性

1. Attribute with sdo:aliasName - PHP では SDO プロパティのエイリアスをサポートしていません。
2. Attribute with default value - PHP では SDO プロパティのデフォルト値をサポートしていません。
3. Attribute with fixed value - PHP では SDO の読み込み専用プロパティやデフォルト値をサポートしていません。

4. *Attribute referencing a DataObject with sdo:propertyType - sdo:propertyType="..." はサポートしていません。*
5. *Attribute with bi-directional property to a DataObject with sdo:oppositeProperty and sdo:propertyType - PHP では SDO opposite をサポートしていません。*

XSD の要素

1. *Element with sdo:aliasName - PHP では SDO プロパティのエイリアスをサポートしていません。*
2. *Element with substitution group.*

XSD Elements with Simple Type

1. *Element of SimpleType with default - PHP では SDO のデフォルト値をサポートしていません。*
2. *Element of SimpleType with fixed value - PHP では SDO の読み込み専用プロパティやデフォルト値をサポートしていません。*
3. *Element of SimpleType with sdo:string - sdo:string="true" はサポートしていません。*
4. *Element referencing a DataObject with sdo:propertyType - sdo:propertyType="..." はサポートしていません。*
5. *Element with bi-directional reference to a DataObject with sdo:oppositeProperty and sdo:propertyType - PHP では SDO opposite をサポートしていません。*

SDO_DAS_XML_Document::getRootDataObject

(No version information available, might be only in CVS)

SDO_DAS_XML_Document::getRootDataObject — ルート SDO_DataObject を返す

説明

SDO_DataObject ***SDO_DAS_XML_Document::getRootDataObject*** *(void)*

ルート SDO_DataObject を返します。

パラメータ

返り値

ルート SDO_DataObject を返します。

SDO_DAS_XML_Document::getRootElementName

(No version information available, might be only in CVS)

SDO_DAS_XML_Document::getRootElementName — ルート要素の名前を返す

説明

string ***SDO_DAS_XML_Document::getRootElementName*** *(void)*

ルート要素の名前を返します。

パラメータ

返り値

ルート要素の名前を返します。

SDO_DAS_XML_Document::getRootElementURI

(No version information available, might be only in CVS)

SDO_DAS_XML_Document::getRootElementURI — ルート要素の URI 文字列を返す

説明

string ***SDO_DAS_XML_Document::getRootElementURI*** *(void)*

ルート要素の URI 文字列を返します。

パラメータ

返り値

ルート要素の URI 文字列を返します。

SDO_DAS_XML_Document::setEncoding

(No version information available, might be only in CVS)

SDO_DAS_XML_Document::setEncoding — エンコーディングを文字列で設定する

説明

void ***SDO_DAS_XML_Document::setEncoding*** *(string $encoding)*

エンコーディングを文字列で設定します。

パラメータ

encoding

エンコーディング文字列。

返り値

何も返しません。

SDO_DAS_XML_Document::setXMLDeclaration

(No version information available, might be only in CVS)

SDO_DAS_XML_Document::setXMLDeclaration — xml 宣言を設定する

説明

void ***SDO_DAS_XML_Document::setXMLDeclaration*** *(bool $xmlDeclatation)*

XML ドキュメントの最初に XML 宣言を生成するかどうかを制御します。 true を設定すると XML 宣言が生成され、 false にすると宣言を抑制します。

パラメータ

xmlDeclatation

XML 宣言を設定するための Boolean 値。

返り値

何も返しません。

SDO_DAS_XML_Document::setXMLVersion

(No version information available, might be only in CVS)

SDO_DAS_XML_Document::setXMLVersion — xml バージョンを文字列で設定する

説明

void ***SDO_DAS_XML_Document::setXMLVersion*** *(string $xmlVersion)*

xml バージョンを文字列で設定します。

パラメータ

xmlVersion

xml バージョン文字列。

返り値

何も返しません。

SDO_DAS_XML::addTypes

(No version information available, might be only in CVS)

SDO_DAS_XML::addTypes — 2 番目以降のスキーマファイルを SDO_DAS_XML オブジェクトに読み込む

説明

void ***SDO_DAS_XML::addTypes*** *(string $xsd_file)*

事前に静的メソッド ***create()*** *で作成した XML DAS に対して、2 番目以降のスキーマファイルを読み込みます。 有効なスキーマファイルならなんでも指定できますが、 このメソッドを使用法としてよくあるものは、 追加の複合型や名前の定義を含むスキーマファイルを追加することです。 親ドキュメントの例 4 を参照ください。*

パラメータ

xsd_file

XSD スキーマファイルへのパス。

返り値

成功した場合は何も返しません。 それ以外の場合は以下に示す例外がスローされます。

エラー / 例外

SDO_TypeNotFoundException

もとになるモデルで型が定義されていない場合にスローされます。

SDO_DAS_XML_ParserException

XSD ファイルのパース時に何らかの問題が発生した場合にスローされます。

SDO_DAS_XML_FileException

指定されたファイルが見つからない場合にスローされます。

SDO_DAS_XML::create

(No version information available, might be only in CVS)

SDO_DAS_XML::create — スキーマファイルを指定して SDO_DAS_XML オブジェクトを作成する

説明

SDO_DAS_XML ***SDO_DAS_XML::create*** *([mixed $xsd_file [, string $key]])*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

これは、SDO_DAS_XML クラスの唯一の静的メソッドです。 SDO_DAS_XML オブジェクトを作成するために使用します。

パラメータ

xsd_file

XSD スキーマファイルへのパス。このパラメータはオプションです。 省略した場合は、SDO 基本型のみが定義されている DAS を作成します。 スキーマファイルは ***addTypes()*** *メソッドで読み込むことができます。 文字列か、あるいは値の配列となります。*

key

返り値

成功した場合は SDO_DAS_XML オブジェクトを返します。 それ以外の場合は以下に示す例外がスローされます。

エラー / 例外

SDO_TypeNotFoundException

もとになるモデルで型が定義されていない場合にスローされます。

SDO_DAS_XML_ParserException

XSD ファイルのパース時に何らかの問題が発生した場合にスローされます。

SDO_DAS_XML_FileException

指定されたファイルが見つからない場合にスローされます。

SDO_DAS_XML::createDataObject

(No version information available, might be only in CVS)

SDO_DAS_XML::createDataObject — 名前空間 URI および型名を指定して SDO_DataObject を作成する

説明

SDO_DataObject ***SDO_DAS_XML::createDataObject*** *(string $namespace_uri , string $type_name)*

名前空間 URI および型名を指定して SDO_DataObject を作成します。 型名はモデルで定義されていなければなりません。 定義されていない場合は SDO_TypeNotFoundException がスローされます。

パラメータ

namespace_uri

型名の名前空間 URI。

type_name

型名。

返り値

成功した場合に SDO_DataObject を返します。

エラー / 例外

SDO_TypeNotFoundException

もとになるモデルで型が定義されていない場合にスローされます。

SDO_DAS_XML::createDocument

(No version information available, might be only in CVS)

SDO_DAS_XML::createDocument — ファイルや文字列から読み込まずに、 XML ドキュメントオブジェクトをスクラッチから作成する

説明

SDO_DAS_XML_Document **SDO_DAS_XML::createDocument** ([string $document_element_name])
SDO_DAS_XML_Document **SDO_DAS_XML::createDocument** (string $document_element_namespace_URI , string $document_element_name [, SDO_DataObject $dataobject])

XML Document オブジェクトを返します。空のルート要素がひとつだけ含まれ、 プロパティは何も設定されていません。このメソッドの目的は、ファイルや文字列からドキュメントを読み込むのではなく スクラッチから XML ドキュメントを作成することです。 作成されたドキュメントは、まるで何の要素も属性も持たない 空のドキュメント要素がひとつだけあるようなドキュメントが読み込まれたかのようになります。

createDocument() は、 ドキュメント要素が何であるかを知っている必要があります。 単純な場合はこれは必須ではありません。 省略しても名前を特定できる場合には、メソッドに渡すパラメータを省略することができます。 しかし同一 XML DAS に複数のスキーマファイルを読み込むことも可能であり、 この場合には複数のドキュメント要素が定義される可能性があります。 さらに、名前空間が違うだけで同じ名前のドキュメント要素が存在する可能性もあります。 これらの場合にうまく対応するため、ドキュメント要素の名前を指定したり 要素名と名前空間を指定することができます。

パラメータ

document_element_name

ドキュメント要素の名前。複数の値をとる可能性がある場合にのみ必要となります。

document_element_namespace_URI

ドキュメント要素名のうち、名前空間の部分。 同じ名前のドキュメント要素が複数存在する場合にのみ必要となります。

dataobject

返り値

成功した場合に SDO_XML_DAS_Document オブジェクトを返します。

エラー / 例外

SDO_UnsupportedOperationException

要素名、あるいは要素名と名前空間 URI が渡されたものの、 それがモデルで見つからなかった場合にスローされます。

SDO_DAS_XML::loadFile

(No version information available, might be only in CVS)

SDO_DAS_XML::loadFile — xml インスタンスドキュメントへのパスを指定し、 SDO_DAS_XML_Document オブジェクトを返す

説明

SDO_XMLDocument **SDO_DAS_XML::loadFile** (string $xml_file)

指定したアドレスの xml インスタンスドキュメントから SDO_DataObjects のツリーを作成します。 SDO_DAS_XML_Document オブジェクトを返します。 ルートデータオブジェクトを取得するには SDO_DAS_XML_Document::getRootDataObject メソッドを使用します。

パラメータ

xml_file

インスタンスドキュメントへのパス。 ローカルファイルへのパス、あるいは URL が指定できます。

返り値

成功した場合は SDO_DAS_XML_Document オブジェクトを返します。 それ以外の場合は以下に示す例外がスローされます。

エラー / 例外

SDO_TypeNotFoundException

もとになるモデルで型が定義されていない場合にスローされます。

SDO_PropertyNotFoundException

もとになるモデルで型のプロパティが定義されていない場合にスローされます。

SDO_DAS_XML_ParserException

XSD ファイルのパース時に何らかの問題が発生した場合にスローされます。

SDO_DAS_XML_FileException

指定されたファイルが見つからない場合にスローされます。

SDO_DAS_XML::loadString

(No version information available, might be only in CVS)

SDO_DAS_XML::loadString — xml インスタンス文字列を指定し、 SDO_DAS_XML_Document オブジェクトを返す

説明

SDO_DAS_XML_Document **SDO_DAS_XML::loadString** (string $xml_string)

指定した xml インスタンス文字列から SDO_DataObjects のツリーを作成します。 SDO_DAS_XML_Document オブジェクトを返します。 ルートデータオブジェクトを取得するには SDO_DAS_XML_Document::getRootDataObject メソッドを使用します。

パラメータ

xml_string

xml 文字列。

返り値

成功した場合は SDO_DAS_XML_Document オブジェクトを返します。 それ以外の場合は以下に示す例外がスローされます。

エラー / 例外

SDO_TypeNotFoundException

もとになるモデルで型が定義されていない場合にスローされます。

SDO_PropertyNotFoundException

もとになるモデルで型のプロパティが定義されていない場合にスローされます。

SDO_DAS_XML_ParserException

XSD ファイルのパース時に何らかの問題が発生した場合にスローされます。

SDO_DAS_XML::saveFile

(No version information available, might be only in CVS)

SDO_DAS_XML::saveFile — SDO_DAS_XML_Document オブジェクトをファイルに保存する

説明

void **SDO_DAS_XML::saveFile** (SDO_XMLDocument $xdoc , string $xml_file [, int $indent])

SDO_DAS_XML_Document オブジェクトをファイルに保存します。

パラメータ

xdoc

SDO_DAS_XML_Document オブジェクト。

xml_file

xml ファイル。

indent

オプションの引数で、xml を整形するかどうかを指定します。 負でない整数値を指定すると、それが xml の字下げ幅になります。 つまり、例えば 2 を指定すると、xml の子要素は親要素から 空白 2 つ分右にずれた場所から始まることになります。0 を指定すると、 xml は完全に左寄せされた形式になります。-1 を指定すると、一切整形を行いません。 xml は長い 1 行のファイルとなります。

返り値

None.

エラー / 例外

SDO_DAS_XML_FileException

指定されたファイルが見つからない場合にスローされます。

SDO_DAS_XML::saveString

(No version information available, might be only in CVS)

SDO_DAS_XML::saveString — SDO_DAS_XML_Document オブジェクトを文字列に保存する

説明

string **SDO_DAS_XML::saveString** (SDO_XMLDocument $xdoc [, int $indent])

SDO_DAS_XML_Document オブジェクトを文字列に保存します。

パラメータ

xdoc

SDO_DAS_XML_Document オブジェクト。

indent

オプションの引数で、xml を整形するかどうかを指定します。 負でない整数値を指定すると、それが xml の字下げ幅になります。 つまり、例えば 2 を指定すると、xml の子要素は親要素から 空白 2 つ分右にずれた場所から始まることになります。0 を指定すると、 xml は完全に左寄せされた形式になります。-1 を指定すると、一切整形を行いません。 xml は長い 1 行のデータとなります。

返り値

xml 文字列を返します。

目次

SDO リレーショナルデータアクセスサービス関数

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

SDO でリレーショナルデータアクセスサービスを使用するには、 SDO の元となっている概念を理解しておく必要があります。例えば データグラフ、データオブジェクト、切断された動作、変更履歴、 XPath およびプロパティの表現方法などです。これらの考え方に なじみがない場合は、まず最初に SDO の節 を参照したほうがよいでしょう。 さらに、リレーショナル DAS では、バックエンドのデータベース固有の 実装を意識せずに使用させるために PDO 拡張モジュールを使用しています。 リレーショナル DAS を使用するには、PDO データベース接続を作成して 渡す必要があります。そのため、 PDO の節 も参照したほうがよいでしょう。

リレーショナル DAS の役割は、リレーショナルデータベースと アプリケーションの間でのデータのやり取りを行うことです。 これを行うためには、データベースのエンティティ - テーブル、カラム、主キー、外部キー - および SDO モデルの要素 - 型、プロパティ、包含関係など の対応関係をリレーショナル DAS が知っておく必要があります。 リレーショナル DAS を作成する際に、これらの情報をメタデータとして 指定します。

操作の概要

1. まずはじめに リレーショナル DAS のコンストラクタをコールし、 データベースと SDO モデルの関連付けを定義したメタデータを渡します。以下でいくつかの例を示します。
2. 次に行うことは、リレーショナル DAS の **executeQuery()** メソッドをコールして 実行する SQL 文を渡すか、あるいは **executePreparedQuery()** メソッドをコールして プレースホルダつきのプリペアドステートメントと挿入する値のリストを 渡すことでしょう。また、実行するクエリ自身についてのメタデータも 指定する必要があります。DAS は、このメタデータによって データベースからどんなカラムがどんな順番で返ってくるのかを 知ります。そのほか、PDO データベース接続も渡さなければなりません。

   executeQuery() あるいは **executePreparedQuery()** から返される値は、 結果セットのすべてのデータを含む正規化されたデータグラフです。 複数のテーブルから取得したデータを返すクエリの場合、このグラフには 複数のデータオブジェクトが含まれ、それらは SDO の包含関係で 連結されています。データの中には、SDO の（包含関係ではない）参照も含まれるかもしれません。

   クエリが実行されてデータグラフが作成された後は、リレーショナル DAS やデータベース接続のインスタンスは必要ありません。 データベース上でロックを保持することはありません。 リレーショナル DAS および PDO データベース接続は、それぞれ メモリから削除されます。

3. データグラフの中のデータに変更が加えられることも大いにありえるでしょう。 データグラフは PHP のセッション内にシリアライズすることが可能で、 いちどきりのクライアント - サーバのやり取りだけではなく複数に またがって使用できます。データオブジェクトを作成してグラフに 追加する・グラフ内の既存のデータオブジェクトを削除する・ グラフ内のデータオブジェクトを変更するなどが可能です。

4. 最後に、リレーショナル DAS の **applyChanges()** メソッドを使用して、データグラフに対する変更内容をデータベースに 書き戻します。そのためには、先ほどと同じメタデータを使用して リレーショナル DAS の新しいインスタンスを作成する必要があります。 また、データベースへの接続も確立しなければなりません。 接続およびデータグラフが、**applyChanges()** に渡されます。ここで、リレーショナル DAS は変更の内容を吟味して INSERT、UPDATE および DELETE のうち適切な SQL 文を作成します。 UPDATE および DELETE 文を実行するには、データベース内のデータが もとのままでなければなりません。そのため、もしデータベース内の データが変更されていた場合は、それが検出されます。ここでは そのような衝突は発生しなかったことにしましょう。これにより、 変更内容がデータベースに適用されました。この後は、 データグラフにさらに変更を加えて再度適用することもできますし、 あるいはデータグラフを削除することもできます。

別の方法でデータベース内のデータを扱うこともできます。 例えば、事前に **executeQuery()** をコールせずに、 単にデータオブジェクトを作成してそれをデータベースに書き戻すことができるのです。 この方法については、下の 例 で説明します。

インストール手順

すべての SDO コンポーネントのインストール手順は、 SDO のドキュメントにある インストール手順 に書かれています。

リレーショナル DAS が PHP で書かれているので、 include_path の場所に配置する必要があるということに注意しましょう。

もちろん、アプリケーションではこのようにしてリレーショナル DAS をインクルードする必要があります。

```
<?php
require_once  'SDO/DAS/Relational.php';
?>
```

要件

リレーショナル DAS を使用するには、SDO 拡張モジュールがインストール されている必要があります。SDO 拡張モジュールを使用するには、 PHP 5.1 以降のバージョンが必要で、リレーショナル DAS を使用するには より新しいバージョンが必要となります。これには PDO の重要な修正が 含まれています。PHP の要件についての最新情報は、PECL パッケージの CHANGELOG を参照ください。これを書いている時点では、リレーショナル DAS に必要なのは PHP 5.1 の最新のベータ版、すなわち PHP 5.1.0 です。

リレーショナル DAS は、PDO を使用してリレーショナルデータベースに アクセスします。そのため、さまざまなリレーショナルデータベースで 実行できなければなりません。これを書いている時点では、以下の環境で テスト済みです。

- MySQL 4.1.14 の Windows 版。 リレーショナル DAS は、PHP 5.1.0 のビルド済みバイナリに 含まれる php_pdo_mysql ドライバで正常に動作します。
- MySQL 4.1.13 の Linux 版。 MySQL 用 PDO ドライバの最新版が必要です。これは PHP 5.1.0 に 組み込まれています。もし既に PECL から通常のドライバを インストールしている場合は、 **pear uninstall pdo_mysql** でそれをアンインストールします。その後、 **--with-pdo-mysql** オプションを指定して PHP の configure を行います。
- DB2 8.2 Personal Edition の Windows 版。 リレーショナル DAS は、PHP 5.1.0 のビルド済みバイナリに 含まれる php_pdo_odbc ドライバで正常に動作します。
- DB2 8.2 Personal Developer's Edition の Linux 版。 PHP の configure およびビルドの際に必要なインクルードファイルが 含まれているため、Developer's Edition を使用しなければなりません。 **--with-pdo-odbc=ibm-db2** オプションを指定して PHP の configure を行います。

リレーショナル DAS は、変更をデータベースに適用する際に トランザクションを使用します。つまり、変更を適用する前には **PDO::beginTransaction()** をコールし、変更が完了すると **PDO::commit()** あるいは **PDO::rollback()** をコールします。どのデータベースを選択するにしても、そのデータベース および PDO ドライバはこれらのコールをサポートしていなければなりません。

制限事項

現在のリレーショナル DAS のリリースには、以下の制限があります。

- null をサポートしていません。SQL の NULL 型をサポートしていません。 データオブジェクトのプロパティに PHP の NULL を代入することは できず、リレーショナル DAS はそれをデータベースに NULL としては 書き込みません。クエリの結果に null が見つかった場合、対応する プロパティには何も設定されません。
- SDO のリレーションシップのうち、2 つの形式にしか対応していません。 以下で説明するリレーショナル DAS のメタデータは、 「複数の値をとる包含関係」「単一の値をとる（包含関係ではない）参照」 の 2 つの SDO リレーションシップにしか対応していません。 SDO では、値が単一であるか複数であるかと包含関係の有無は それぞれ独立して指定可能です。つまり、SDO で定義されている すべてのリレーションには対応していないということです。 これらのリレーションを使用できれば有用でしょうが、現在の実装では 管理することができないということです。 例えば、「単一の値をとる包含関係」は使用できません。
- SDO のデータ型のすべてに対応しているわけではありません。 リレーショナル DAS では、SDO モデルのすべてのプリミティブ型 プロパティを文字列型として扱います。 SDO では integer、float、boolean そして日付や時刻などの さまざまな型が定義されています。リレーショナル DAS では、文字列だけでこれらのデータを十分に扱うことができます。 なぜなら PHP、PDO とデータベースの間で、 データベースに保存する際に適切な型変換を行ってくれるからです。 他の DAS との間でデータグラフのやりとりをする際に、 これが何がしかの問題を起こすことがあります。
- ひとつのテーブルに対して複数の外部キーを定義することはできません。 メタデータでは、テーブルごとにひとつの外部キーしか指定できません。 この外部キーは、サポートしている 2 つの SDO リレーションシップのうちの どちらかひとつに関連付けられます。この制限のもとでは表すことのできない状況があるのは明らかです。例えば、 ひとつのテーブルから他のテーブルに対して、(包含関係でない）参照をふたつ以上を設定することができません。

例

この節では、リレーショナル DAS を使用してリレーショナルデータベースの データを作成し、取得し、更新して削除する方法を説明します。 以下の例のほとんどで使用しているのは、3 つのテーブル（company、department、employee）からなるデータベースで、 会社の中には部署が存在し、部署には従業員が所属しているという構造です。 これは、SDO の資料中の例でも使用されています。 » Service Data Objects specification の例か、SDO 拡張モジュールのマニュアルの 例 を参照ください。

リレーショナル DAS は、リレーショナルデータベースの定義 およびそれをどのように SDO に関連付けるかを定義したメタデータを 使用して作成されます。これ以降で、メタデータの構造 およびリレーショナル DAS の作成方法について説明します。 以下の例では、インクルードされた PHP ファイル内に メタデータが存在するものとします。

以下で説明する例およびその他の例は、リレーショナル DAS パッケージの Scenarios ディレクトリ内に存在します。

メタデータに間違いがあった場合、あるいはデータベースに対する SQL 文の実行時にエラーが発生した場合には、リレーショナル DAS は例外をスローします。簡潔に説明するため、以下の例では リレーショナル DAS のコールの際の try/catch ブロックの使用を 省略しています。

これらの例は、SDO が想定している使用法とは 以下の 2 つの点で大きく異なります。

まず、ここではデータベースに対するすべての操作を ひとつのスクリプト内で完結させています。これは現実的ではありませんが、 リレーショナル DAS の使用法を説明するという点でこの方法を選択しました。 通常は、データベースに対する処理はいくつかに分割され、そのたびに PHP セッションへのデータグラフの保存やそこからの引き出しが発生します。 これにより、アプリケーションがユーザと対話的なやりとりをすることになるでしょう。

2 番目に、データベースに対するすべてのクエリがハードコーディングされており、 変数の置換を使用していません。例の中で使用するのは安全な **executeQuery()** コールであり、 例について説明するためにこのようにしています。 しかし、実際に使用する場合は、すべての SQL 文が安全であるとは限りません。 SQL クエリ内での変数の置換を安全に行い、SQL インジェクションを防ぐには、 **executePreparedQuery()** でプレースホルダを含むプリペアドステートメントを使用し、 置換する値のリストを渡します。

メタデータを指定する

最初の少し長めの節では、データベースや SDO モデルについて表したメタデータが、リレーショナル DAS からどのように返されるのかを説明します。

リレーショナル DAS のコンストラクタが実行される際に、 情報を受け取る必要があります。この情報は、 コンストラクタへの最初の引数として連想配列で渡します。 これにより、リレーショナル DAS に対して、 必要なデータベース情報を伝えます。 テーブル名、カラム、主キーや外部キーの情報がここに含まれます。 必要な内容を理解することはきわめて簡単でしょう。 一度書いてしまえば、それを PHP ファイルに保存して 必要に応じてインクルードすることができます。 コンストラクタの二番目、三番目の引数として、リレーショナル DAS がオブジェクト間の関連やデータグラフの形式を知るために必要な情報を渡します。 これによって、最終的にデータベースのデータをグラフに正規化する方法が決まります。

データベースのメタデータ

コンストラクタに対する最初の引数で、対象となる リレーショナルデータベースを定義します。

各テーブルは、最大で 4 つのキーからなる連想配列で表されます。

キー	値
name	テーブルの名前。
columns	カラムの名前を含んだ配列。順不同。
PK	主キーとなるカラムの名前。
FK	ふたつのエントリ 'from' および 'to' からなる配列で、 外部キーを含むカラムおよび外部キーが参照するテーブルを指定します。 テーブルに外部キーが存在しない場合には 'FK' を指定する必要はありません。指定できる外部キーの数は ひとつだけです。また、外部キーが参照する先は、 そのテーブルの主キーでなければなりません。

```php
<?php
/*****************************************************************
* データベースを定義するメタデータ
*****************************************************************/
$company_table  = array (
    'name' => 'company',
    'columns' => array('id', 'name',  'employee_of_the_month'),
    'PK' => 'id',
    'FK' => array (
        'from' => 'employee_of_the_month',
        'to' => 'employee',
        ),
    );
$department_table  = array (
    'name' => 'department',
    'columns' => array('id', 'name', 'location', 'number', 'co_id'),
    'PK' => 'id',
    'FK' => array (
        'from' => 'co_id',
        'to' => 'company',
        )
    );
$employee_table  = array (
    'name' => 'employee',
    'columns' => array('id', 'name', 'SN', 'manager', 'dept_id'),
    'PK' => 'id',
    'FK' => array (
        'from' => 'dept_id',
        'to' => 'department',
        )
    );
$database_metadata  = array($company_table, $department_table, $employee_table);
?>
```

このメタデータに対応するリレーショナルデータベースは、 例えば MySQL の場合は以下のように定義されます。

```
 create table company (
    id integer auto_increment,
    name char(20),
     employee_of_the_month integer,
    primary key(id)
 );
 create table department (
    id integer auto_increment,
    name char(20),
    location char(10),
    number integer(3),
    co_id integer,
    primary key(id)
 );
 create table employee (
    id integer auto_increment,
    name char(20),
    SN char(4),
    manager tinyint(1),
    dept_id integer,
    primary key(id)
```

```
 );
```

あるいは DB2 なら以下のようになります。

```
create table company ( ¥
      id integer not null generated by default as identity,  ¥
      name varchar(20), ¥
       employee_of_the_month integer, ¥
      primary key(id) )
create table department ( ¥
      id integer not null generated by default as identity, ¥
      name varchar(20), ¥
      location varchar(10), ¥
      number integer, ¥
      co_id integer, ¥
      primary key(id) )
create table employee ( ¥
      id integer not null generated by default as identity, ¥
      name varchar(20), ¥
      SN char(4), ¥
      manager smallint, ¥
      dept_id integer, ¥
      primary key(id) )
```

この例ではデータベースに外部キーが定義されておらず、 データベースには参照整合性が強制されていません。しかし、 department テーブルの co_id カラム および employee テーブルの dept_id カラムは、 それぞれ company あるいは department レコードの主キーを含むように 意図されています。そのため、これらの 2 つのカラムは外部キーのように動作します。

この例には 3 つめの外部キーがあります。それは company レコードの employee_of_the_month カラムが employee テーブルのひとつの行をさすというものです。この外部キーと、 その他の 2 つのキーとの違いに注意しましょう。 employee_of_the_month カラムは単一の値をとる関係を表します。ある会社には、 「今月の従業員」はひとりしか存在しえません。 co_id および dept_id は複数の値をとる関係を表します。会社には多くの部署がありますし、 ひとつの部署には多くの従業員が属しています。 この差は、 メタデータの残りの部分で company-department および department-employee の関係を包含関係として扱う時に明らかになります。

データベースのメタデータを作成する際に従わなければならない規則は 以下のとおりです。

- すべてのテーブルには主キーが存在し、その主キーがメタデータで 指定されている必要があります。主キーがなければ、オブジェクトの 一意性を確保することができません。テーブルを作成する SQL 文を見てわかるように、主キーは自動生成することができます。 つまり、レコードが挿入される際に自動で値が設定されるということです。 この場合、自動生成された主キーはデータベースから取得可能で、 データベースに行が追加されると、すぐにデータオブジェクトに反映されます。
- データベースに存在するすべてのカラムについてメタデータで記述する必要はありません。 実際に使用するものだけを指定します。例えば、company テーブルの中に SDO 経由でのアクセスを必要としないカラムがある場合、そのカラムを メタデータに記述する必要はありません。一方、それを記述したとしても 何の問題もありません。アプリケーションから値を取得したり代入したり することがなければ、未使用のカラムを記述していても何の影響も及ぼしません。
- データベースのメタデータで、外部キーの指し示す先として カラム名ではなくテーブル名が指定されていることにお気づきでしょう。 厳密に言えば、リレーショナルモデルでは 主キー以外を対象とした外部キーを指定することも可能です。 しかし、SDO モデルを作成する際に使用できるのは 主キーを対象とした外部キーだけなので、 メタデータではテーブル名だけを指定します。 これにより、外部キーの指し示す先がそのテーブルの主キーであると解釈されます。

これらの規則に従い、データベースを定義する SQL 文があれば、データベースのメタデータを作成するのは簡単です。

リレーショナル DAS でのメタデータの扱い

リレーショナル DAS は、データベースのメタデータを使用して SDO モデルの大半を作成します。データベースのメタデータ内にある 各テーブルに対して、SDO 型が定義されます。 プリミティブ型を表すカラム（外部キーとして定義されていないカラム）は、SDO 型のプロパティとして追加されます。

すべてのプリミティブなプロパティは、SDO モデルにおいては 文字列型で表されます。これは、SQL での型が何であっても同じです。 データをデータベースに書き戻す際も、リレーショナル DAS が作成する SQL 文では文字列として扱います。 そして、データベースがそれを適切な型に変換します。

外部キーの扱いかたには二通りの方法があります。 どちらが使用されるかは、コンストラクタの第三引数（SDO の包含関係について定義するもの）によって決まります。 そのため、これについては以下の SDO の包含関係 の節で改めて説明します。

アプリケーションのルート型の指定

コンストラクタの二番目の引数に指定するのが、アプリケーションのルート型です。 各データグラフの真のルートとなるのは特別なルート型のオブジェクトで、 すべてのアプリケーションデータオブジェクトはその下のどこかに位置します。 SDO モデルのさまざまなアプリケーション型のなかで、 どれかひとつの型がデータグラフのルート直下にある必要があります。 データベースのメタデータ内にテーブルがひとつしかない場合は、 アプリケーションのルート型は推定できます。そのため、この引数は省略可能です。

SDO 包含関係の指定

コンストラクタの三番目の引数で、モデルの型をどのように連結して グラフを構成するかを定義します。複数の型によってグラフが構成される場合に、 その親子関係を指定します。この関係は、 データ内の外部キーでサポートされている必要があります。 これについては後ほど説明します。

メタデータは、ひとつあるいは複数の連想配列を含む配列となります。 個々の連想配列が親と子を表します。以下に company（会社）と department（部署）、そして department（部署） と employee（従業員）の親子関係の例を示します。 これらのそれぞれが SDO のプロパティとなり、 SDO モデルにおける「複数の値をとる包含関係」を定義するものになります。

```
<?php
$department_containment  = array( 'parent' => 'company', 'child' => 'department');
$employee_containment  = array( 'parent' => 'department', 'child' => 'employee');

$SDO_containment_metadata  = array($department_containment, $employee_containment);
?>
```

データベースのメタデータ内にある外部キーは、 複数の値を持つ包含関係のプロパティか 単一の値を持つ包含関係でないプロパティのいずれかと解釈されます。どちらになるかは、 対応する SDO の包含関係がメタデータ内で定義されているかどうかによって決まります。 この例では、department から company への外部キー（department テーブルのカラム co_id）および employee から department への外部キー（employee テーブルのカラム dept_id）が SDO の包含関係と解釈されます。 SDO の包含関係メタデータで示されているそれぞれの関連については、 対応するデータベースメタデータで外部キーが存在しなければなりません。 外部キー列の値は、データオブジェクト内には現れません。その代わりに、 親から子に対する包含関係として表されます。つまり、例えばデータベースの department テーブルにあるカラム co_id は、department 型のプロパティにはなりません。その代わりに、 company 型のほうに department という名前の関係ができます。 ここで、外部キーと親子関係は向きが逆であることに注意しましょう。 外部キーは、department から company に対して指定しますが、 親子関係は company から department に向けて指定します。

この例の三番目の外部キーである employee_of_the_month は、異なる方法で処理されます。これは、SDO の包含関係メタデータとはならず、 もうひとつの方法で解釈されます。company オブジェクト上での 単一の値をとる包含関係でない参照となり、employee 型の SDO データオブジェクトへの参照をそこに代入します。 これは、company 型が持つプロパティとなります。 SDO データグラフ内でこのプロパティに値を代入するには、employee オブジェクトを含むグラフを取得し、そこに値を代入します。 この方法については、後の例で説明します。

ひとつのテーブルによる例

これ以降の例でのリレーショナル DAS のデータグラフの構成は次のようになります。 アプリケーションデータオブジェクトがひとつだけ、ひとつの会社に関する情報が company テーブルに格納されています。これらの例は SDO の機能を活用するものではありませんし、 直接 SQL 文を発行したほうがずっと効率的でしょう。 これらの例は、リレーショナル DAS をどのように使用するのかを説明するためのものです。

そのため、データベースのメタデータは極力シンプルになるようにしました。 company テーブルだけしか含まないようにしています。 コンストラクタで使用している二番目、三番目の引数や、 クエリの例で使用しているカラム指定子は、オプションとなります。

Example#1 データオブジェクトの作成

もっとも単純な例として、データオブジェクトをひとつ作成して それをデータベースに書き込むことを考えます。この例では、 company オブジェクトがひとつ作成されます。その名前を 'Acme' と指定したうえで、リレーショナル DAS をコールして 変更内容をデータベースに書き込みます。会社名を設定する際には、 プロパティ名を使用します。オブジェクトのプロパティにアクセスする その他の方法については、SDO の 例 を参照ください。

データオブジェクトを作成できるのは、 あらかじめ先立つデータオブジェクトがある場合のみです。そのため、 リレーショナル DAS への最初のコールは、 まずルートオブジェクトを取得するためのものとなります。 これは、空のデータグラフの作成をすることで行います。 特別なルートオブジェクトは、木構造の真のルートとなるものです。 company データオブジェクトは、その後ルートオブジェクトに対して **createDataObject()** をコールすることで作成します。 これは、company データオブジェクトを作成して、 データグラフのルートオブジェクトの 'company' という名前のプロパティにそれを挿入します。

リレーショナル DAS が変更を適用する際には、シンプルな insert 文 'INSERT INTO company (name) VALUES ("Acme");' が実行されます。自動生成された主キーがデータオブジェクトに設定され、 変更概要がリセットされます。そのため、同じデータオブジェクトを そのまま使い続け、さらに変更した内容をもう一度適用することもできます。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*************************************************************
 * メタデータから、DAS を作成します。
  *************************************************************/
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);

/*************************************************************
 * ルートオブジェクトを取得し、その下に company オブジェクトを
 * 作成します。そしてデータオブジェクトに対して変更を加えます。
  *************************************************************/
$root  = $das   -> createRootDataObject();
$acme  = $root  -> createDataObject('company');

$acme->name  = "Acme";

/*************************************************************
 * データベースに接続し、オブジェクトをデータベースに書き込みます。
  *************************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);
$das  -> applyChanges($dbh, $root);
?>
```

Example#2 データオブジェクトの取得

この例では、データベースから取得されるデータオブジェクトはひとつです。 あるいは、もし 'Acme' という名前の会社を複数登録していたのなら、複数返される可能性もあります。返された会社データのそれぞれについて、 プロパティ name および id の値を出力します。

この例における **executeQuery()** の 3 番目の引数では、カラム名に修飾子が必要です。 というのも、メタデータ内の他のテーブルで name および id という名前のカラムが用いられているからです。 もし名前が重複するような心配がないのなら、これは省略できます。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*************************************************************
 * メタデータから、DAS を作成します。
  *************************************************************/
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);

/*************************************************************
 * データベース接続を取得します。
  *************************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);

/*************************************************************
 * クエリを発行し、company オブジェクトを取得します。
 * 複数存在するかもしれません。
  *************************************************************/
$root  = $das->executeQuery($dbh,
            'select  name,  id  from  company  where  name="Acme"',
              array('company.name', 'company.id') );

/*************************************************************
 * name および id を出力します。
  *************************************************************/
foreach  ($root['company'] as $company) {
    echo "データベースから取得した会社の名前は  " .
    $company['name'] . "  で、その  id  は  " . $company['id'] . "  です\n";
}
?>
```

Example#3 データオブジェクトの更新

この例は、これまでのふたつを組み合わせたものです。というのも、 オブジェクトを更新するには、まずそれを取得しなければならないからです。 この例のコードは、会社名を逆さにし（つまり 'Acme' が 'emcA' となります）、 オブジェクトを作成したときと同じ方法でそれをデータベースに書

き戻します。 この例のクエリではその両方の会社名で検索しているので、 プログラムを繰り返し実行すると、そのたびに会社名が反転します。

この例では、同じリレーショナル DAS のインスタンスが **applyChanges()** で再利用されます。 ちょうど PDO データベースハンドルと同じような扱いです。 このように使用しても問題ありません。また、既存のインスタンスを破棄して 新しいインスタンスを作り直すこともできます。 データグラフをアプリケーションに返した後は、 その状態に関する情報はリレーショナル DAS から削除されます。 必要なデータはすべてデータグラフ自身が保持しており、 もしそれが存在しない場合はメタデータから再作成されます。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*************************************************************
 * メタデータから、DAS を作成します。
  ************************************************************/
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);

/*************************************************************
 * データベース接続を取得します。
  ************************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);

/*************************************************************
 * クエリを発行し、company オブジェクトを取得します。
 * 複数存在するかもしれません。
  ************************************************************/
$root  = $das->executeQuery($dbh,
    'select  name,  id  from  company  where  name="Acme"  or  name="emcA"',
    array('company.name', 'company.id') );

/*************************************************************
 * 最初の会社の名前を変更します。
  ************************************************************/
$company  = $root['company'][0];
echo "取得した会社の名前は  "  . $company->name  .  "  です\n";
$company->name  = strrev($company->name);

/*************************************************************
 * 変更内容を書き戻します。
  ************************************************************/
$das->applyChanges($dbh,$root);
?>
```

Example#4 データオブジェクトの削除

'Acme'、あるいはその逆の 'emcA' という名前の会社をすべて取得することができました。 次に、これらをすべてグラフから削除するために unset を使用します。

この例では、ひとつの動作ですべてが削除されます。つまり、該当する（包含関係を定義する）プロパティを解放します。 これらを個別に削除していくことも可能です。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*************************************************************
 * メタデータから、DAS を作成します。
  ************************************************************/
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);

/*************************************************************
 * データベース接続を取得します。
  ************************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);

/*************************************************************
 * クエリを発行し、company オブジェクトを取得します。
 * 複数存在するかもしれません。
  ************************************************************/
$root  = $das->executeQuery($dbh,
    'select  name,  id  from  company  where  name="Acme"  or  name="emcA"',
    array('company.name', 'company.id') );

/*************************************************************
 * データグラフのすべての会社を削除します。
  ************************************************************/
unset($root['company']);

/*************************************************************
 * 変更内容を書き戻します。
  ************************************************************/
$das->applyChanges($dbh,$root);
?>
```

二つのテーブルの例

これ以降の例では、company データベース内のふたつのテーブル company と department を使用します。これらの例は、 リレーショナル DAS の機能をより活用するためのものです。

この一連の例では、まず company および department を作成し、 それを取得し、更新して最後に削除します。複数のオブジェクトからなる データグラフの動きについて、ここで説明します。 この例では、まず最初に company および department のふたつのテーブルの中身を削除していることに注意しましょう。 これにより、クエリの結果がどのようになるのかをはっきりさせています。

これらの例については、リレーショナル DAS パッケージの Scenarios ディレクトリ内にあるスクリプト 1cd-CRUD にまとめてあります。

Example#5 会社がひとつ、部署がひとつの例 - 作成

先の、company データオブジェクトをひとつだけ作成する例で示したように、 リレーショナル DAS を作成した後でまず行うことは **createRootDataObject()** をコールして空のデータグラフを持つ特別なルートオブジェクトを取得することです。 その後、このルートオブジェクトの子要素として company オブジェクトを作成し、company オブジェクトの子要素として department オブジェクトを作成します。

変更内容を適用する際には、包含関係を定義する外部キー制約を満たすため、 リレーショナルが特別な処理を行います。特に、 自動生成される主キーが含まれる場合の処理が大切です。 この例では、company テーブルで自動生成された主キー id と department テーブルのカラム co_id の関係を指定しなければなりません。 company および department に最初にデータを挿入する際、 リレーショナル DAS はまず最初に company に行を挿入してから PDO の **getLastInsertId()** メソッドで 自動生成された主キーを取得し、department に行を挿入する際の co_id カラムにその値を指定しなければなりません。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*****************************************************************************
* ふたつのテーブルを空にします。
*****************************************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);
$pdo_stmt  = $dbh->prepare('DELETE  FROM  COMPANY;');
$rows_affected  = $pdo_stmt->execute();
$pdo_stmt  = $dbh->prepare('DELETE  FROM  DEPARTMENT;');
$rows_affected  = $pdo_stmt->execute();

/***********************************************************
* Acme という名前の会社、Shoe という名前の部署を作成します。
***********************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);

$root  = $das  -> createRootDataObject();

$acme  = $root  -> createDataObject('company');
$acme  -> name  = "Acme";

$shoe  = $acme->createDataObject('department');
$shoe->name  = 'Shoe';

$das  -> applyChanges($dbh, $root);

?>
```

Example#6 会社がひとつ、部署がひとつの例 - 取得および更新

この場合、**executeQuery()** に渡す SQL クエリは inner join を使用して company テーブルと department テーブルを連結します。company テーブルと department テーブルの両方の主キーがクエリに含まれている必要があります。 結果セットは正規化されたデータグラフ形式になります。 **executeQuery()** をコールする際に、 三番目の引数としてカラム指定子を渡していることに注意しましょう。 これにより、どのカラムが結果セットのどこに対応するのかを リレーショナル DAS に教えています。

クエリ内でカラム co_id が使用されていますが、 これは結果セットでは必要ないことに注意しましょう。 データグラフを作成する際にリレーショナル DAS がどのような処理をするのかを理解するには、 結果セットがどのようになるのかを図示してみるといいでしょう。 データベース内のデータは正規化されており、 複数の部署データが外部キーを通じてひとつの会社データを参照するようになっていますが、 結果セット内のデータは正規化されていません。つまり、 もしひとつの会社に複数の部署があるのなら、 会社のデータはその行の数だけ繰り返されます。 リレーショナル DAS は、この手順を逆転させて 結果セットを正規化されたデータグラフに戻し、company オブジェクトがひとつだけになるようにしなければなりません。

この例では、リレーショナル DAS が結果セットとカラム指定子を調べ、 company テーブルと department テーブルからデータを取得し、 それぞれの主キーを検索し、各 department とその親となる company を関連付けます。該当する company をまだ取得していない場合は (これは主キーで判断します)、新しい company オブジェクトを作成して department オブジェクトをその下に関連付けます。 すでに company オブジェクトが存在する場合は、 その下に department オブジェクトを作成するだけです。

この方法によって、リレーショナル DAS は複数の会社と複数の部署が含まれるデータを 取得して再度正規化します。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/***********************************************************
 * 会社、そして部署 Shoe を取得します。
 * それから Shoe を削除して IT を追加します。
  ***********************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);

$root  = $das->executeQuery($dbh,
'select  c.id,  c.name,  d.id,  d.name  from  company  c,  department  d  where  d.co_id  =  c.id',
array('company.id','company.name','department.id','department.name'));

$acme  = $root['company'][0];            // 最初の会社を取得します - 'Acme' となるでしょう。
$shoe  = $acme['department'][0];         // その下の最初の部署を取得します - 'Shoe' となるでしょう。

unset($acme['department'][0]);

$it  = $acme->createDataObject('department');
$it->name  = 'IT';

$das  -> applyChanges($dbh, $root);
?>
```

Example#7 会社がひとつ、部署がふたつの例 - 取得および削除

この例では、会社と部署を取得し、それから削除します。 それぞれを個別に削除する必要はありません (そうすることも可能です)。 データグラフから company オブジェクトを削除すると、 その下に関連付けられている department も削除されます。

company オブジェクトを削除する際に、PHP の unset をコールしていることに注意しましょう。unset は、 それを保持しているプロパティに対して行わなければなりません。 この場合は、ルートオブジェクトの company プロパティに対して行います。 つまり、このようにしなければなりません。

```
<?php
unset($root['company'][0]);
?>
```

次のようにしてはいけません。

```
<?php
unset($acme); // 間違い
?>
```

単に $acme を unset しただけだと、 変数は削除されますが、データグラフ内のデータはそのまま残ってしまいます。

```php
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*************************************************************
 * 会社、そして部署 IT を取得します。
 * 会社全体を削除します。
  *************************************************************/
$dbh  =  new  PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);
$das  =  new  SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);

$root  =  $das->executeQuery($dbh,
'select  c.id,  c.name,  d.id,  d.name  from  company  c,  department  d  where  d.co_id  =  c.id',
array('company.id','company.name','department.id','department.name'));

$acme  =  $root['company'][0];
$it  =  $acme['department'][0];

unset($root['company'][0]);

$das  ->  applyChanges($dbh,  $root);

?>
```

三つのテーブルの例

これ以降の例では、company データベース内の三つのテーブル company と department そして employee をすべて使用します。 これらの例では、これまでに説明してこなかった機能を説明します。 それは、包含関係ではない参照である employee_of_the_month です。

先に説明した company と department の例と同様、 これ以降の例でもデータグラフに対する一連の処理をすべて説明していきます。

Example#8 会社がひとつ、部署がひとつ、従業員が一人の例 - 作成

この例では、ひとつの会社にひとつの部署があり、 そこに従業員が一人だけ所属しているという構成になります。 この例では、まず最初に三つのテーブルの中身を削除していることに注意しましょう。 これにより、クエリの結果がどのようになるのかをはっきりさせています。

company、department および employee を作成した後で、company の employee_of_the_month プロパティを作成して新しい従業員を参照させているところに注目しましょう。 これは包含関係ではない参照なので、データグラフ内で employee オブジェクトを作成するまでは参照させることができません。 包含関係ではない参照の管理には注意が必要です。例えば、 ある employee が department から削除されたとすると、employee_of_the_month プロパティを削除するか代入しなおさない限り、 データグラフを保存できなくなります。SDO データグラフの制約上、 包含関係でない参照の対象となっているオブジェクトは、 同時にいずれかの包含関係から到達可能である必要があります。

グラフの内容をデータベースに挿入する際の手順は、 company と department だけの時の例と似ています。しかし、 employee_of_the_month があるために少し複雑になります。 リレーショナル DAS は、包含関係で構成される木構造の順に、 オブジェクトを挿入していく必要があります。つまりまず company、 次に department、そして employee となります。なぜなら、 親データの自動生成された主キーの値を、子のデータに含めなければならないからです。 しかし、company を挿入した時点ではまだ「今月の従業員」 になる employee が挿入されておらず、その主キーの値がわかりません。 そこで、employee レコードが挿入されてその主キーが判明した時点で、 最後の処理が行われます。つまり company レコードを employee の主キーで更新します。

```php
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*****************************************************************************
* 3 つのテーブルを空にします。
*****************************************************************************/
$dbh  =  new  PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);
$pdo_stmt  =  $dbh->prepare('DELETE  FROM  COMPANY;');
$rows_affected  =  $pdo_stmt->execute();
$pdo_stmt  =  $dbh->prepare('DELETE  FROM  DEPARTMENT;');
$rows_affected  =  $pdo_stmt->execute();
$pdo_stmt  =  $dbh->prepare('DELETE  FROM  EMPLOYEE;');
$rows_affected  =  $pdo_stmt->execute();

/*****************************************************************************
* 小規模ながらも完全なる会社を作成します。
*   会社の名前は Acme。
*   部署はひとつだけで、Shoe。
*   従業員は一人だけで、Sue。
*   「今月の従業員」は Sue。
*****************************************************************************/
$das  =  new  SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);
$dbh  =  new  PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);

$root              =  $das   ->  createRootDataObject();
$acme              =  $root  ->  createDataObject('company');
$acme  ->  name    =  "Acme";
$shoe              =  $acme  ->  createDataObject('department');
$shoe  ->  name    =  'Shoe';
$shoe  ->  location  =  'A-block';
$sue               =  $shoe  ->  createDataObject('employee');
$sue  ->  name     =  'Sue';
$acme  ->  employee_of_the_month  =  $sue;

$das  ->  applyChanges($dbh,  $root);

echo  "Acme  のデータを、ひとつの部署と一人の従業員とともに書き戻します\n";
?>
```

Example#9 会社がひとつ、部署がひとつ、従業員が一人の例 - 取得および更新

この場合、リレーショナル DAS に渡す SQL 文は、inner join を使用して三つのテーブルからのデータを取得します。ここでは、 これまでの例に出てこなかった新しい内容は特にありません。

新しい部署と従業員の追加によってグラフが更新され、 グラフ内の既存のオブジェクトの name プロパティに少し変更を加えます。 その後、それらの変更内容を書き戻します。リレーショナル DAS は、 データグラフへの追加・変更および削除の内容を適用します。

```php
<?php
require_once  'SDO/DAS/Relational.php';
```

```
require_once  'company_metadata.inc.php';

/*****************************************************************************
 * 会社のデータを取得し、いろいろ変更します。
 *  会社、部署そして従業員の名前を変更します。
 *  新しい部署と従業員を追加します。
 *  「今月の従業員」を変更します。
  *****************************************************************************/
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);

$root  = $das->executeQuery($dbh,
    "select  c.id,  c.name,  c.employee_of_the_month,  d.id,  d.name,  e.id,  e.name  " .
    "from  company  c,  department  d,  employee  e  " .
    "where  e.dept_id  =  d.id  and  d.co_id  =  c.id  and  c.name='Acme'",
     array('company.id','company.name','company.employee_of_the_month',
     'department.id','department.name','employee.id','employee.name'));
$acme          = $root['company'][0];

$shoe          = $acme->department[0];
$sue           = $shoe  ->  employee[0];

$it            = $acme->createDataObject('department');
$it->name      = 'IT';
$it->location  = 'G-block';
$billy         = $it->createDataObject('employee');
$billy->name   = 'Billy';

$acme->name    = 'MegaCorp';
$shoe->name    = 'Footwear';
$sue->name     = 'Susan';

$acme->employee_of_the_month  = $billy;
$das  ->  applyChanges($dbh,  $root);
echo "会社のデータを、追加した部署や従業員そして変更後の名前 (Megacorp/Footwear/Susan) などとともに書き戻します\n";

?>
```

Example#10 会社がひとつ、部署がふたつ、従業員が二人の例 - 取得および削除

company は、5 つのデータオブジェクトを含むデータグラフとして取得されます。 5 つとは、すなわちその会社、ふたつの部署、そして二人の従業員です。 company オブジェクトをグラフから削除すると、 その配下にあるすべてのオブジェクトがグラフから削除されます。 つまり、5 つの SQL DELETE 文が作成されて実行されます。 取得したすべてのフィールドの内容が WHERE 句に指定されるので、 別のプロセスがデータベースの内容を変更していた場合にはそれを検出することができます。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*****************************************************************************
 * もう一度読み込み、それを削除します。
 * 一部を削除し、変更を適用し、同じグラフで作業を続けることもできますが、
 * 破綻しないように注意が必要です。「今月の従業員」になっている従業員を削除するには、
 * まずその割り当てを変更しておかなければなりません。安全のため、ここでは
 * 会社ごとすべて削除しています。
  *****************************************************************************/
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_containment_metadata);
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);

$root  = $das->executeQuery($dbh,
    "select  c.id,  c.name,  c.employee_of_the_month,  d.id,  d.name,  e.id,  e.name  " .
    "from  company  c,  department  d,  employee  e  " .
    "where  e.dept_id  =  d.id  and  d.co_id  =  c.id  and  c.name='MegaCorp';",
     array('company.id','company.name','company.employee_of_the_month',
     'department.id','department.name','employee.id','employee.name'));
$megacorp  = $root['company'][0];

unset($root['company']);
$das  ->  applyChanges($dbh,  $root);

echo "Deleted  the  company,  departments  and  employees  all  in  one  go.¥n";

?>
```

トレース

変更をデータベースに書き戻す際に、実際にはどのような SQL 文が生成されるのかが気になることもあるでしょう。 SDO/DAS/Relational.php の先頭で、SQL 文の作成および実行の処理を追跡するための定数が定義されていることがわかります。 生成された SQL 文を見るには、DEBUG_EXECUTE_PLAN を **TRUE** にしてみましょう。

定義済みクラス

リレーショナル DAS では 2 つのクラスが提供されています。それは リレーショナル DAS 自身、および例外時にスローされる Exception のサブクラスです。リレーショナル DAS で使用できるパブリックコールは 4 つです。まずコンストラクタ、それから 空のデータグラフからルートオブジェクトを取得するための **createRootDataObject()**、 リレーショナルデータベースからのデータを含むデータグラフを取得するための **executeQuery()**、 データグラフへの変更をリレーショナルデータベースに書き戻すための **applyChanges()** です。

SDO_DAS_Relational

SDO_DAS_Relational_Exception 以外で唯一、 アプリケーションと直接やりとりすることが想定されているオブジェクトです。

メソッド

- __construct - 渡されたメタデータに基づくモデルからリレーショナル DAS を構築します。
- createRootDataObject - 特別なルートオブジェクトを含む空ではないデータグラフを取得します。

- executeQuery - リテラル文字列で渡された SQL クエリを実行し、 正規化されたデータグラフ形式で結果を返します。
- executePreparedQuery - プリペアドステートメントとして渡された SQL クエリに プレースホルダを置換する値のリストを指定して実行し、 正規化されたデータグラフ形式でデータを返します。
- applyChanges - データグラフの変更の概要を調べ、それをデータベースに書き戻します。 楽観的な同時並行性 (concurrency) に従います。

SDO_DAS_Relational_Exception

PHP の Exception のサブクラスです。 Exception に対して何も機能を追加しません。 メタデータ内でエラーが発生したり SQL の実行時に予期せぬ失敗が 発生した際に、有用な説明を含めてスローされます。

SDO_DAS_Relational::applyChanges

(No version information available, might be only in CVS)

SDO_DAS_Relational::applyChanges — データグラフに対する変更を、データベースに書き戻す

説明

void **SDO_DAS_Relational::applyChanges** (PDO $database_handle , SDODataObject $root_data_object)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

PDO データベースハンドルとデータグラフのルートオブジェクトを受け取り、 データグラフの変更内容を吟味したうえで、その変更内容をデータベースに適用します。 適用される変更の内容は、データオブジェクトの作成や削除、 そしてデータオブジェクトのプロパティへの変更です。

パラメータ

PDO_database_handle

PDO 拡張モジュールを使用して作成されます。 PDO データベースハンドルを作成する典型的な方法は、このようになります。

```
$dbh = new PDO("mysql:dbname=COMPANYDB;host=localhost",DATABASE_USER,DATABASE_PASSWORD);
```

root_data_object

特別なルートオブジェクトで、これはすべての SDO データグラフのトップレベルにあります。

返り値

なし。しかし、渡されたデータグラフは何も変更されず、 そのまま使用可能なことに注意しましょう。 さらに、もしデータオブジェクトの作成の際にテーブルの主キーを自動生成していたのなら、 その主キーの値がここでデータオブジェクトに設定されます。 変更内容の書き込みが正常に終了すると、そのデータグラフに関連する 変更概要が消去されます。これにより、 そのデータグラフにさらに変更を加えてはそれを適用するといったことが可能になります。 この方法により、同じデータグラフを使用して繰り返し変更内容を適用することができます。

エラー / 例外

SDO_DAS_Relational::applyChanges() は、変更内容を正しく適用できなかった場合に SDO_DAS_Relational_Exception をスローします。

リレーショナル DAS は、変更内容の適用を始める前に データベースのトランザクションを開始します。 そして、変更内容がすべて正常に適用できた場合にのみそれをコミットします。 リレーショナル DAS は、更新あるいは削除する行を指定するための適切な where 句を含む update 文あるいは delete 文を作成します。これは、 データを最初に取得した際の値に基づいて作成されます。 これにより、楽観的な同時並行性 (concurrency) が実装されています。 この update 文や delete 文が更新、削除に失敗したとすると、 その原因はおそらく、最初にデータを取得した後で データベース内のデータが変更されたことでしょう。 どんな場合でも、もし何らかの理由で更新が失敗したら トランザクションはロールバックされます。そして例外がスローされます。 例外の中には、失敗した SQL 文が含まれています。

リレーショナル DAS は、PDO の例外も捕捉します。その中に含まれる PDO の診断情報を SDO_DAS_Relational_Exception に格納し、それをスローします。

例

リレーショナル DAS についての一般的な情報は 例 を参照ください。ここにはこのメソッドをコールする例がたくさんあります。 また、リレーショナル DAS が作成した SQL 文を見るには、 トレース を参照ください。

SDO_DAS_Relational::__construct

(No version information available, might be only in CVS)

SDO_DAS_Relational::__construct — リレーショナルデータアクセスサービスのインスタンスを作成する

説明

SDO_DAS_Relational **SDO_DAS_Relational::__construct** (array $database_metadata [, string $application_root_type [, array $SDO_containment_references_metadata]])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

渡されたメタデータを使用して、 リレーショナルデータアクセスサービスのインスタンスを作成します。

パラメータ

database_metadata

ひとつあるいは複数のテーブルの定義を含む配列で、 配列の各要素は連想配列となります。この連想配列に含まれるキーは name、 columns、 PK そしてオプションで FK となります。 メタデータについての完全な説明は、リレーショナル DAS の全般的な情報の中にある メタデータ の節を参照ください。

application_root_type

各データグラフのルートは特別なルート型のオブジェクトであり、 アプリケーションデータオブジェクトはこの下に作成されます。 SDO モデルのさまざまなアプリケーション型の中で、 どれかひとつをデータグラフのルート直下に置く必要があります。 データベースのメタデータにひとつだけのテーブルしかない場合は、 アプリケーションのルート型が推定できるので この引数を省略できます。

SDO_containment_references_metadata

ひとつあるいは複数の包含関係の定義を含む配列。 配列の各要素は連想配列で、parent および child のふたつのキーを持ちます。 包含関係によって、モデル内の型を木構造に連結する方法を定義します。 アプリケーションのルート型として指定された型は、 包含関係の親要素のひとつとして存在しなければなりません。 そのアプリケーションが一度にひとつのテーブルしか扱わず、 かつモデル内に包含関係が存在しない場合には、この引数は省略できます。 メタデータに関する議論については、リレーショナル DAS 全般に関する情報の中の メタデータ に関する節を参照ください。

返り値

成功した場合に SDO_DAS_Relational オブジェクトを返します。

エラー / 例外

***SDO_DAS_Relational::__construct()** は、メタデータに何らかの問題が見つかった場合に SDO_DAS_Relational_Exception をスローします。*

例

メタデータについての完全な説明は、リレーショナル DAS の全般的な情報の中にある メタデータ の節を参照ください。

SDO_DAS_Relational::createRootDataObject

(No version information available, might be only in CVS)

SDO_DAS_Relational::createRootDataObject — 別の空のデータグラフ内の、特別なルートオブジェクトを返す。 データグラフをスクラッチから作成する際に使用する

説明

*SDODataObject **SDO_DAS_Relational::createRootDataObject** (void)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

*別の空のデータグラフのトップレベルにある、特別なルートオブジェクトを返します。 **executeQuery()** をコールしてデータグラフを作成するのではなく、 アプリケーション内でスクラッチからデータグラフを作成したい場合に使用します。*

*この特別なルートオブジェクトには、複数の値をとることのできる包含関係のプロパティが ひとつあります。プロパティの名前は、 リレーショナル DAS を作成した際に渡された、アプリケーションのルート型の名前となります。 プロパティの値には、このルート型の値しか指定できません。 アプリケーションがルート型に対してできることは、 **createDataObject()** をコールしてアプリケーションのルート型を渡し、 そのアプリケーション型のデータオブジェクトを作成させることだけです。*

パラメータ

なし。

返り値

ルートオブジェクトを返します。

エラー / 例外

なし。

例

このメソッドをコールする多くの例については、 リレーショナル DAS についての一般的な情報の、 例 を参照ください。

SDO_DAS_Relational::executePreparedQuery

(No version information available, might be only in CVS)

SDO_DAS_Relational::executePreparedQuery — プリペアドステートメントとして渡された SQL クエリにプレースホルダ置換用の値を指定して実行し、 結果を正規化されたデータグラフ形式で返す

説明

*SDODataObject **SDO_DAS_Relational::executePreparedQuery** (PDO $database_handle , PDOStatement $prepared_statement , array $value_list [, array $column_specifier])*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クエリをリレーショナルデータベースに対して実行します。 実行には、渡された PDO データベースハンドルを使用します。 シンプルな **executeQuery()** と違う点は、 こちらはプリペアドステートメントと値のリストを受け取るというところです。 これを使用するほうがよい場面としては、次のふたつが考えられます。 まず、同じ文が引数だけを変えて何度も実行される場合。このような場合は、 最初に一度だけ文を準備しておくことでパフォーマンスを向上することができます。 もうひとつは、SQL 文にさまざまな値が指定される可能性があり、 それらの値が完全には信頼できないといった場合です。 後者の場合、それらの値を単純に連結して SQL を作成するのは危険です。 というのは、値の中に SQL で特別な意味を持つ文字が含まれているかもしれないからです。 このような、いわゆる SQL インジェクション攻撃から身を守るには、 プレースホルダ（あるいはパラメータマーカとも言い、'?' で表されます）を使用した SQL 文を準備しておくほうが安全です。 実際に使用する値のリストは、別の引数として指定します。 それ以外の点では、この関数は **executeQuery()** と同じです。つまり、 メタデータから作成されたモデルを使用し、結果セットを取得します。 そして結果をデータグラフとして返します。

パラメータ

PDO_database_handle

PDO 拡張モジュールを使用して作成します。 PDO データベースハンドルを作成する典型的な方法は、このようになります。

```
$dbh  =  new  PDO("mysql:dbname=COMPANYDB;host=localhost",DATABASE_USER,DATABASE_PASSWORD);
```

prepared_statement

データベースに対して実行するプリペアドステートメント。 PDO の **prepare()** メソッドによって評価されます。

value_list

SQL 文でプレースホルダを置き換える値の配列。 SQL 文にプレースホルダ（パラメータマーカ）がない場合は、 この引数には **NULL** あるいは空の配列を指定します。

column_specifier

リレーショナル DAS では、結果セットの各カラムを調べ、 それがどのテーブルのどのカラムからきたものなのかを知る必要があります。 データ自身からそれらの情報を取得できる場合もありますが、 そうでない場合もあります。取得できない場合にはカラム指定子が必要となります。 これは、カラムを識別するための配列です。 配列の各項目は、単純な テーブル名.カラム名 形式の文字列となります。

カラム指定子が必要となるのは、 データベースのメタデータ内に同じ名前のカラムが複数ある場合です。 例で使用しているデータベースでは、すべてのテーブルに id および name のふたつのカラムがあります。 リレーショナル DAS が PDO から結果セットを読み込む際には、 属性 PDO_FETCH_ASSOC を使用できます。これは、 結果セットのカラム名をリレーショナル DAS のカラム名に対応させますが、 同じ名前の重複を識別できません。そのため、 これは、結果セットでカラム名が重複する可能性がない場合にのみ動作します。

まとめます。そのカラムがどのテーブルのものかが判別できなくなる 可能性が少しでもある場合には、カラム指定子の配列を設定します。 省略できるのは、データベースのメタデータ内にあるすべてのカラム名が 一意な場合のみです。

例 で示したサンプルでは、すべてカラム指定子を設定しています。 インストール先の Scenarios ディレクトリ内に、カラム指定子を使用しない例がひとつあります。 これは employee テーブルだけを使用するものです。 ひとつのテーブルしか使用していないので、 カラム名が重複することがないわけです。

返り値

データグラフを返します。具体的には、特別な型のルートオブジェクトを返します。 このルートオブジェクトの下に、結果セットからのデータが格納されています。 ルートオブジェクトは複数の値を持つプロパティを持っており、 そのプロパティは、コンストラクタで指定した アプリケーションのルート型と同じ名前になります。 また、そのプロパティの内容は、 アプリケーションのルート型のオブジェクトとなります。

クエリがデータを返さなかった場合にも特別なルート型のオブジェクトが返されますが、 その中にあるアプリケーションルート型のプロパティの内容は空となります。

エラー / 例外

SDO_DAS_Relational::executePreparedQuery() は、データグラフを正常に作成できなかった場合に SDO_DAS_Relational_Exception をスローします。 これが起こる原因は、いろいろ考えられます。例えば、 すべてのオブジェクトの結果セットで主キーがなかった場合などです。 また、何らかの PDO 例外が発生した場合には、 それをキャッチして PDO の診断情報を SDO_DAS_Relational_Exception に格納し、それをスローします。

例

Example#1 executePreparedQuery() によるデータオブジェクトの取得

この例では、データベースからひとつのデータオブジェクトを取得します - あるいは、もし 'Acme' という名前の会社が複数あるのなら結果は複数かもしれません。 返されたそれぞれの会社について、プロパティ name および id の内容を表示します。

executePreparedQuery() の使用法についてのその他の例は、 sdo/DAS/Relational/Scenarios にあるサンプルコードを参照ください。

```
<?php
require_once  'SDO/DAS/Relational.php';
require_once  'company_metadata.inc.php';

/*************************************************************
 * メタデータから DAS を作成します
  *************************************************************/
$das  = new SDO_DAS_Relational  ($database_metadata,'company',$SDO_reference_metadata);

/*************************************************************
 * データベースとの接続を確立します
  *************************************************************/
$dbh  = new PDO(PDO_DSN,DATABASE_USER,DATABASE_PASSWORD);

/*************************************************************
 * クエリを発行し、company オブジェクトを取得します
 * プレースホルダを指定したプリペアドクエリを使用します
  *************************************************************/
$name  = 'Acme';
$pdo_stmt  = $dbh->prepare('select  name,  id  from  company  where  name=?');
$root  = $das->executePreparedQuery(
    $dbh,
    $pdo_stmt,
     array($name),
     array('company.name', 'company.id'));

/*************************************************************
```

```
 * name および id を表示します
 ************************************************************/
foreach ($root['company'] as $company) {
    echo "データベースから取得した会社の name は " .
    $company['name'] . " そして id は " . $company['id'] . "¥n";
}
?>
```

SDO_DAS_Relational::executeQuery

(No version information available, might be only in CVS)

SDO_DAS_Relational::executeQuery — SQL クエリをリレーショナルデータベースに対して実行し、 結果を正規化されたデータグラフ形式で返す

説明

SDODataObject **SDO_DAS_Relational::executeQuery** (PDO $database_handle , string $SQL_statement [, array $column_specifier])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

クエリをリレーショナルデータベースに対して実行します。 実行には、渡された PDO データベースハンドルを使用します。 メタデータから作成されたモデルを使用し、結果セットを取得します。 結果をデータグラフとして返します。

パラメータ

PDO_database_handle

PDO 拡張モジュールを使用して作成します。 PDO データベースハンドルを作成する典型的な方法は、このようになります。

$dbh = new PDO("mysql:dbname=COMPANYDB;host=localhost",DATABASE_USER,DATABASE_PASSWORD);

SQL_statement

データベースに対して実行する SQL 文。

column_specifier

リレーショナル DAS では、結果セットの各カラムを調べ、 それがどのテーブルのどのカラムからきたものなのかを知る必要があります。 データ自身からそれらの情報を取得できる場合もありますが、 そうでない場合もあります。取得できない場合にはカラム指定子が必要となります。 これは、カラムを識別するための配列です。 配列の各項目は、単純な テーブル名.カラム名 形式の文字列となります。

カラム指定子が必要となるのは、 データベースのメタデータ内に同じ名前のカラムが複数ある場合です。 例で使用しているデータベースでは、すべてのテーブルに id および name のふたつのカラムがあります。 リレーショナル DAS が PDO から結果セットを読み込む際には、 属性 PDO_FETCH_ASSOC を使用できます。これは、 結果セットのカラム名をリレーショナル DAS のカラム名に対応させますが、 同じ名前の重複を識別できません。そのため、 これは、結果セットでカラム名が重複する可能性がない場合にのみ動作します。

まとめます。そのカラムがどのテーブルのものかが判別できなくなる 可能性が少しでもある場合には、カラム指定子の配列を設定します。 省略できるのは、データベースのメタデータ内にあるすべてのカラム名が 一意な場合のみです。

例 で示したサンプルでは、すべてカラム指定子を設定しています。 インストール先の Scenarios ディレクトリ内に、カラム指定子を使用しない例がひとつあります。 これは employee テーブルだけを使用するものです。 ひとつのテーブルしか使用していないので、 カラム名が重複することがないわけです。

返り値

データグラフを返します。具体的には、特別な型のルートオブジェクトを返します。 このルートオブジェクトの下に、結果セットからのデータが格納されています。 ルートオブジェクトは複数の値を持つプロパティを持っており、 そのプロパティは、コンストラクタで指定した アプリケーションのルート型と同じ名前になります。 また、そのプロパティの内容は、 アプリケーションのルート型のオブジェクトとなります。

クエリがデータを返さなかった場合にも特別なルート型のオブジェクトが返されますが、 その中にあるアプリケーションルート型のプロパティの内容は空となります。

エラー / 例外

SDO_DAS_Relational::executeQuery() は、データグラフを正常に作成できなかった場合に SDO_DAS_Relational_Exception をスローします。これが起こる原因は、いろいろ考えられます。例えば、 すべてのオブジェクトの結果セットで主キーがなかった場合などです。 また、何らかの PDO 例外が発生した場合には、 それをキャッチして PDO の診断情報を SDO_DAS_Relational_Exception に格納し、それをスローします。

例

このメソッドをコールする例については、リレーショナル DAS の全般的な情報の中にある 例 の節を参照ください。

目次

セマフォ・共有メモリおよび IPC 関数(semaphore)

導入

このモジュールは、System V IPC 関連の関数へのラッパーを提供します。 セマフォ・共有メモリおよびプロセス間通信(IPC)がその中に含まれます。

セマフォは、マシーン上のリソースへの排他的アクセス機能や、 同時にあるリソースを使用することができるプロセスの数を制限するために 使用することができます。

このモジュールは、System V 共有メモリを使用した共有メモリ関数も 提供します。共有メモリは、グローバル変数へのアクセス手段を提供するために 使用することが可能です。別の httpd デーモンおよび (Perl, C, ... のような)他のプログラムさえ、グローバルデータ交換を 提供するこのデータにアクセスすることが可能です。 共有メモリは、同時アクセスに関して安全ではないということを覚えておいて ください。 同期をとるには、セマフォを使用してください。

Unix OS による共有メモリの制限

SHMMAX	共有メモリの最大サイズ。通常は 131072 バイト
SHMMIN	共有メモリの最小サイズ。通常は 1 バイト
SHMMNI	共有メモリセグメントの最大数。通常は 100
SHMSEG	プロセス毎の共有メモリの最大数。通常は 6

メッセージング関数は、他のプロセスと相互にメッセージを送受信する ために使用することができます。 これにより簡単で効率的なプロセス間のデータ交換が可能であり、 Unix ドメインソケットを用いる場合のような設定は不要です。

注意: この拡張モジュールは Windows 環境では利用できません。

要件

外部ライブラリを必要としません。

インストール手順

この関数はデフォルトでは有効になってはいません。System V セマフォの サポートを有効にするには、オプション --enable-sysvsem を指定して PHP を コンパイルする必要があります。System V 共有メモリのサポートを有効にするには、 オプション --enable-sysvshm を 指定して PHP をコンパイルする必要があります。System V メッセージを有効に するには、オプション --enable-sysvmsg を指定して PHP をコンパイル します。

実行時設定

php.ini の設定により動作が変化します。

セマフォ設定オプション

名前	デフォルト	変更の可否	変更履歴
sysvmsg.value	"42"	PHP_INI_ALL	
sysvmsg.string	"foobar"	PHP_INI_ALL	

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

System V メッセージ定数

定数	型	変更履歴
MSG_IPC_NOWAIT	integer	
MSG_EAGAIN	integer	5.2.0 以降
MSG_ENOMSG	integer	5.2.0 以降
MSG_NOERROR	integer	
MSG_EXCEPT	integer	

ftok

(PHP 4 >= 4.2.0, PHP 5)

ftok — パス名とプロジェクト ID を、System V IPC キーに変換する

説明

```
int ftok ( string $pathname , string $proj )
```

この関数は、pathname で表される既存のアクセス可能なファイルおよびプロジェクト ID (proj) を、shmop_open() やその他で使用する System V IPC キーに変換します。

パラメータ

pathname

アクセ宇可能なファイルへのパス。

proj

プロジェクト ID。一文字からなる文字列でなければなりません。

返り値

成功した場合には作成されたキーの値を、それ以外の場合には -1 を返します。

参考

- shmop_open()
- sem_get()

---

msg_get_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_get_queue — メッセージキューを作成またはそれにアタッチする

説明

resource **msg_get_queue** (int $key [, int $perms])

msg_get_queue() は、指定した key で System V メッセージキューにアクセスするために使用される ID を返します。 最初にコールされた際には、オプションの perms でメッセージキューを作成します。同じ key で 2 度目に **msg_get_queue()** がコールされると別の ID が返されますが、どちらの ID も同じメッセージキューを指します。

パラメータ

key

perms

キューのパーミッション。デフォルトは 0666 です。 メッセージキューがすでに存在する場合には、perms は無視されます。

返り値

ID を返します。これを使用して、System V メッセージキューにアクセスします。

参考

- msg_remove_queue()
- msg_receive()
- msg_send()
- msg_stat_queue()
- msg_set_queue()

---

msg_receive

(PHP 4 >= 4.3.0, PHP 5)

msg_receive — メッセージキューからメッセージを受信する

説明

bool **msg_receive** (resource $queue , int $desiredmsgtype , int &$msgtype , int $maxsize , mixed &$message [, bool $unserialize [, int $flags [, int &$errorcode]]])

msg_receive() は、指定した queue から指定した desiredmsgtype の最初のメッセージを受信します。

パラメータ

queue

desiredmsgtype

desiredmsgtype が 0 の場合、 キューの先頭にあるメッセージが返されます。desiredmsgtype が 0 より大きな値の場合、その型のメッセージのうち一番最初にあるものが返されます。 desiredmsgtype が 0 より小さな値の場合、 desiredmsgtype の絶対値と同じかそれより小さい型のメッセージのうち一番最初にあるものが返されます。 条件を満たすメッセージがない場合は、 条件を満たすメッセージがキューに投入されるまで待ち続けます。 パラメータ flags に **MSG_IPC_NOWAIT** を指定することで、 ブロックモードではなくすることが可能です。

msgtype

受信したメッセージの型がこのパラメータに保存されます。

maxsize

読み込むメッセージの最大サイズは *maxsize* で指定します。もしキューにあるメッセージのサイズがこれより大きい場合、 (以下で説明する *flags* が設定されていない限り) この関数は失敗します。

message

エラーが発生しなければ、 受信したメッセージは *message* に保存されます。

unserialize

unserialize のデフォルト値は **TRUE** です。 このパラメータが **TRUE** に設定されている場合、 メッセージはセッションモジュールと同様の方法でシリアライズされているものとみなされます。 メッセージは元の状態に復元されたうえでスクリプトに返されます。 これにより、配列やオブジェクト構造体のような複雑な形式のデータを他の PHP スクリプトから簡単に受信することが可能となります。 また、もし WDDX シリアライザを使用しているなら、あらゆる WDDX 互換のソースからデータを受け取ることが可能となります。

unserialize が **FALSE** の場合、 メッセージはバイナリセーフな文字列として返されます。

flags

オプションの *flags* により、低レベルの *msgrcv* システムコールにフラグを渡すことが可能です。デフォルト値は *0* ですが、以下の値のうちのいくつかを(値を足すかあるいは論理和ととることで) 指定することが可能です。

msg_receive のフラグの値

MSG_IPC_NOWAIT	*desiredmsgtype* を満たすメッセージが存在しない場合に、 待ち続けずにすぐに結果を返します。 関数は失敗し、**MSG_ENOMSG** に対応する整数値を返します。
MSG_EXCEPT	このフラグを正の *desiredmsgtype* と組み合わせて使用すると、この関数は *desiredmsgtype* 以外の型をもつ 最初のメッセージを受信するようになります。
MSG_NOERROR	このフラグを設定しておくと、メッセージが *maxsize* より大きい場合に そのメッセージを *maxsize* までに切り詰め、 エラーを返しません。

errorcode

エラーが発生した場合は、オプションの *errorcode* にシステムの *errno* 値が設定されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

処理が正常に完了すると、メッセージキューデータ構造体は以下のように更新されます。 *msg_lrpid* には呼び出し元のプロセス ID が 設定され、*msg_qnum* が 1 減少し、 *msg_rtime* が現在の時刻に設定されます。

参考

- msg_remove_queue()
- msg_send()
- msg_stat_queue()
- msg_set_queue()

---

msg_remove_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_remove_queue — メッセージキューを破棄する

説明

bool **msg_remove_queue** (*resource $queue*)

msg_remove_queue() は、*queue* で指定したメッセージキューを破棄します。この関数を使用するのは、 すべてのプロセスがメッセージキューの使用を終え、 キューが保持するシステムリソースを開放する必要が生じた場合のみです。

パラメータ

queue

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- msg_get_queue()
- msg_receive()
- msg_stat_queue()
- msg_set_queue()

---

msg_send

(PHP 4 >= 4.3.0, PHP 5)

msg_send — メッセージキューにメッセージを送信する

説明

bool **msg_send** (resource $queue , int $msgtype , mixed $message [, bool $serialize [, bool $blocking [, int &$errorcode]]])

msg_send() は、queue で指定したメッセージキューに対して msgtype で指定した型 (0 より大きい数値である必要があります) のメッセージ message を送信します。

パラメータ

queue

msgtype

message

serialize

オプションのパラメータ serialize は、 message を送信する方法を制御します。 serialize のデフォルト値は **TRUE** で、 この場合 message が送信される前に セッションモジュールと同じ方法でシリアライズされます。 これにより、配列やオブジェクトのような複雑な形式のデータを 他の PHP スクリプトに送信することが可能となります。 また、もし WDDX シリアライザを使用しているなら、あらゆる WDDX 互換クライアントに対して同じことが可能となります。

blocking

メッセージがキューに収まらないほど大きい場合は、他のプロセスが 現在キューにあるメッセージを読み込んでキューの空き容量が確保されるまで スクリプトの実行を待ち続けます。これをブロックモードといいます。 オプションのパラメータ blocking を **FALSE** に設定することでブロックモードではなくすることが可能で、 この場合、もしキューの空き容量よりも大きなメッセージを送信すると **msg_send()** はすぐに **FALSE** を返します。 また、オプションのパラメータ errorcode を **MSG_EAGAIN** に設定すると、 少し時間をおいてメッセージを再度送信しなければならないことが戻り値からわかります。

errorcode

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

処理が正常に完了すると、メッセージキューデータ構造体は以下のように更新されます。 msg_lspid には呼び出し元のプロセス ID が 設定され、msg_qnum が 1 増加し、 msg_stime が現在の時刻に設定されます。

参考

- msg_remove_queue()
- msg_receive()
- msg_stat_queue()
- msg_set_queue()

---

msg_set_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_set_queue — メッセージキューデータ構造体の情報を設定する

説明

bool **msg_set_queue** (resource $queue , array $data)

msg_set_queue() により、メッセージキューデータ構造体の msg_perm.uid、msg_perm.gid、msg_perm.mode および msg_qbytes フィールドを変更することが可能です。

データ構造体を変更するには、PHP の実行ユーザがキューの作成者あるいは (現在の msg_perm.xxx フィールドで指定されている)キューの所有者であるか、 あるいは root 権限を有している必要があります。msg_qbytes の値を システムで定義した制限をこえて設定するには、root 権限が必要です。

パラメータ

queue

data

設定したい値を、data 配列に設定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- msg_remove_queue()
- msg_receive()
- msg_stat_queue()
- msg_get_queue()

---

msg_stat_queue

(PHP 4 >= 4.3.0, PHP 5)

msg_stat_queue — メッセージキューデータ構造体の情報を返す

説明

array **msg_stat_queue** (resource $queue)

msg_stat_queue() は、queue で指定したメッセージキューのメタデータを返します。 これは、例えば受信したメッセージがどのプロセスから送信されたのかを調べる場合などに有用です。

パラメータ

queue

返り値

返り値は配列で、そのキーと値は以下のような意味をもちます。

msg_stat_queue の配列構造

msg_perm.uid	キューの所有者の uid 。
msg_perm.gid	キューの所有者の gid 。
msg_perm.mode	キューのファイルアクセスモード。
msg_stime	キューに対して最後にメッセージが送信された時刻。
msg_rtime	キューから最後にメッセージを受信した時刻。
msg_ctime	キューが最後に更新された時刻。
msg_qnum	キューにある読み込み待ちのメッセージの数。
msg_qbytes	送信されたメッセージのうち、まだ受信されていないものを保持する領域として、 キューで現在使用可能なバイト数
msg_lspid	最後にキューに対してメッセージを送信したプロセスの pid 。
msg_lrpid	最後にキューからメッセージを受信したプロセスの pid 。

参考

- msg_remove_queue()
- msg_receive()
- msg_get_queue()
- msg_set_queue()

---

sem_acquire

(PHP 4, PHP 5)

sem_acquire — セマフォを得る

説明

bool **sem_acquire** (resource $sem_identifier)

sem_acquire() は、(必要な場合) セマフォが確保できるまでブロックします。 既に確保されているセマフォを得ようとするプロセスは、 セマフォの獲得により max_aquire 値を超える場合、 永久にブロックされます。

リクエスト処理の後、プロセスにより獲得された全てのセマフォのうち、 明示的に開放されていないものが自動的に開放され、警告が表示されます。

パラメータ

sem_identifier

sem_identifier はセマフォのリソースで、 sem_get() によって得られます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- sem_get()
- sem_release()

---

sem_get

(PHP 4, PHP 5)

sem_get — セマフォ ID を得る

説明

resource **sem_get** (int $key [, int $max_acquire [, int $perm [, int $auto_release]]])

sem_get() は、 System V セマフォを指定したキーでアクセスするために使用可能な ID を返します。

同じキーで **sem_get()** を 2 度コールした場合、 別のセマフォ ID が返されます。 しかし、どちらの ID も同じそのセマフォをアクセスします。

パラメータ

key

max_acquire

同時にセマフォを得ることが可能なプロセス数を max_acquire (デフォルトは 1) にセットします。

perm

セマフォのパーミッション。デフォルトは 0666 です。 実際には、この値はプロセスが現在そのセマフォに付随している 唯一のプロセスであることがわかった場合にのみセットされます。

auto_release

リクエストの終了時に自動的にセマフォを開放するかどうかを指定します。

返り値

成功した場合に正のセマフォ ID、エラー時には **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	auto_release が追加されました。

参考

- sem_acquire()
- sem_release()
- ftok()

---

sem_release

(PHP 4, PHP 5)

sem_release — セマフォを解放する

説明

bool **sem_release** (resource $sem_identifier)

sem_release() は、そのセマフォが コール元のプロセスにより現在確保されている場合、解放します。 そうでない場合、警告が表示されます。

セマフォを解放した後、再び確保するには、sem_acquire() をコールします。

パラメータ

sem_identifier

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- sem_get()
- sem_acquire()

---

sem_remove

(PHP 4 >= 4.0.7, PHP 5)

sem_remove — セマフォを削除する

説明

bool **sem_remove** (resource $sem_identifier)

sem_remove() は、指定したセマフォを削除します。

セマフォを削除した後、そのセマフォにはもうアクセスできません。

パラメータ

sem_identifier

セマフォの ID。sem_get() により作成したものでなければなりません。 それ以外の場合は、警告を発生します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- sem_get()
- sem_release()
- sem_acquire()

---

shm_attach

(PHP 4, PHP 5)

shm_attach — 共有メモリセグメントを作成またはオープンする

説明

int **shm_attach** (int $key [, int $memsize [, int $perm]])

shm_attach() は ID を返します。 これは、指定されたキー key で System V 共有メモリにアクセスする際に使用することが可能です。 最初のコールの際に、サイズが memsize 、 オプションのパーミッション perm を指定した共有メモリセグメントを作成します。

同じ key で **shm_attach()** を 2 回コールした場合は 別の共有メモリ ID が返されますが、両方の ID は同じ共有メモリをアクセスします。 memsize および perm は無視されます。

パラメータ

key

memsize

メモリのサイズ。省略した場合のデフォルトは php.ini の sysvshm.init_mem、あるいは 10000 バイトとなります。

perm

オプションのパーミッション設定。デフォルトは 0666 です。

返り値

共有メモリセグメントの ID を返します。

参考

- **sem_detach()**
- ftok()

---

shm_detach

(PHP 4, PHP 5)

shm_detach — 共有メモリセグメントへの接続を閉じる

説明

bool **shm_detach** (int $shm_identifier)

shm_detach() は、 shm_attach() で作成され、 指定した shm_identifier を有する共有メモリへの接続を閉じます。 共有メモリは、まだ Unix システム上に存在しており、 データはまだ存在するということを覚えておいてください。

パラメータ

shm_identifier

返り値

shm_detach() は、常に **TRUE** を返します。

参考

- shm_attach()

- shm_remove()
- shm_remove_var()

---

shm_get_var

(PHP 4, PHP 5)

shm_get_var — 共有メモリから変数を返す

説明

mixed **shm_get_var** (int $shm_identifier , int $variable_key)

shm_get_var() は、 shm_identifier で指定した共有メモリセグメントから 変数 variable_key を読みこみます。 読み込んだ変数は、まだ共有メモリに存在します。

パラメータ

shm_identifier

共有メモリセグメント。shm_attach() から取得します。

variable_key

変数のキー。

返り値

指定したキーの変数を返します。

---

shm_put_var

(PHP 4, PHP 5)

shm_put_var — 共有メモリの変数を挿入または更新する

説明

bool **shm_put_var** (int $shm_identifier , int $variable_key , mixed $variable)

shm_put_var() は、 指定した variable_key を有する 変数 variable の挿入または更新を行います。

shm_identifier が有効な SysV 共有メモリではない場合や リクエストを処理するために充分な共有メモリが残っていない場合は (E_WARNING レベルの) 警告を発生させます。

パラメータ

shm_identifier

variable_key

variable

変数。全ての 変数型をサポートします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

shm_remove_var

(PHP 4, PHP 5)

shm_remove_var — 共有メモリから変数を削除する

説明

bool **shm_remove_var** (int $shm_identifier , int $variable_key)

指定したvariable_key を有する変数を共有メモリから削除し、占有するメモリを解放します。

パラメータ

shm_identifier

共有メモリの ID。

variable_key

変数のキー。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- shm_remove()

---

shm_remove

(PHP 4, PHP 5)

shm_remove — Unix システムから共有メモリを削除する

説明

bool **shm_remove** (int $shm_identifier)

shm_remove() は、共有メモリ shm_identifier を削除します。 全てのデータは破棄されます。

パラメータ

shm_identifier

共有メモリの ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- shm_remove_var()

目次

SESAM データベース関数

導入

SESAM/SQL-Server は、ドイツの Fujitsu Siemens Computers により 開発されたメインフレーム上のデータベースシステムです。この データベースは、ハイエンドのメインフレーム上でオペレーティングシステム BS2000/OSD を使用して動作します。

数多くの BS2000 システムにて実運用され、SESAM/SQL-Server は以下の 事項を実証しています。

- Java、Web ベースのクライアント/サーバ接続が簡単に可能。
- 99.99% 以上の確率で有効に動作する能力。
- 数万から数十万のユーザを管理する能力。

PHP スクリプトによるデータベース処理を可能にする PHP 3 SESAM インターフェースが利用可能です。

> **注意**: SESAMへのアクセスは、CVS 版の PHP 3 でのみ可能です。 PHP 4 は、SESAM データベースをサポートしません。

実行時設定

php.ini の設定により動作が変化します。

sesam_oml string

ロード可能な SESAM ドライバモジュールを含む BS2000 PLAM ライブラリの名前です。SESAM 関数を使用する際に必要です。 BS2000 PLAMライブラリは、ACCESS=READ,SHARE=YES に設定する必要が あります。これは、Apache サーバのユーザ ID により 読み込み可能とする必要があるためです。

sesam_configfile string

SESAM アプリケーション設定ファイルの名前です。SESAM 関数を 使用する際に必要です。BS2000 ファイルは、Apache のユーザ ID により読み込み可能である必要があります。

アプリケーション設定ファイルは、通常、次のような設定を 含んでいます (SESAM リファレンスマニュアルを参照ください)。

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog string

SESAM メッセージカタログファイルの名前。多くの場合、この ディレクティブは不要です。SESAM メッセージファイルがシステムの BS2000 メッセージファイルテーブルにインストールされていない 場合にのみこのディレクティブを設定することが可能です。

メッセージカタログは、ACCESS=READ,SHARE=YES に設定する必要が あります。これは、Apache サーバのユーザ ID から読み込み可能と しなければならないためです。

設定上の注意

PHP SESAM インターフェースは、スタンドアロン版をサポートしていません。 Apache モジュール版として組み込まれた場合のみ動作します。Apache PHP モジュールにおいて、SESAM インターフェース は Apache 用 ディレクティブにより設定されます。

SESAM 設定用ディレクティブ

ディレクティブ	意味
php3_sesam_oml	ロード可能な SESAM ドライバモジュールが含まれる BS2000 PLAM ライブラリの名前です。SESAM 関数を使用するために必要です。 例: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`

php3_sesam_configfile	SESAM アプリケーション設定ファイルの名前です。SESAM 関数を 使用する際に必要です。
	
	例:
	
	`php3_sesam_configfile $SESAM.SESAM.CONF.AW`
	
	通常、次のような設定が含まれます (SESAM リファレンス マニュアルを参照ください)。
	
	`CNF=B`
	`NAM=K`
	`NOTYPE`

php3_sesam_messagecatalog	SESAM メッセージカタログファイルの名前です。多くの場合、 このディレクティブは不要です。SESAM メッセージファイルがシステムの BS2000 メッセージファイルテーブルにインストールされていない 場合にのみ、このディレクティブを設定することが可能です。 例: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

PHP/SESAM インターフェースの設定に加えて、メインフレーム上の SESAM データベースサーバ自体を設定する必要があります。これは、 次のようになります。

- *SESAM データベースハンドラ (DBH) を開始*
- *SESAM データベースハンドラを指定して、データベースに接続*

PHP スクリプトとデータベースハンドラの間の接続を得るには、選択した SESAM 設定ファイルのパラメータ CNF および NAM が実行中のデータベースハンドラの ID に 一致している必要があります。

分散型データベースの場合、ホストおよびデータベース名を有する 分配テーブルを指定して SESAM/SQL-DCN エージェントを開始する必要が あります。

(POSIX サブシステムで実行されている) PHP と (POSIX サブシステムの外で 実行されている) データベースハンドラの間の通信は、SQLSCI という名前の 特別なドライバモジュールと通常メモリを使用するSESAM接続モジュールで 実現されます。通常メモリアクセスのため、そして、PHP は Web サーバの静的部分であるため、ODBC、JDBC、UTM 経由のリモートアクセスは 不要であり、データベースへのアクセスは非常に高速です。

スモールスタブローダ (SESMOD) のみが PHP にリンクされており、SESAM 接続モジュールは SESAM の OML PLAM ライブラリからロードされます。 設定の際、この PLAM ライブラリの名前と SESAM 設定ファイルを使用するためのファイルリンクを PHP に指定する必要があります (SESAM V3.0 において、SQLSCI は標準配布の SESAM ツールライブラリから入手可能です)。

SQL コマンドにおけるシングルクオートのクオートは、(シングルクオートの 前にバックスラッシュを付加するのではなく) シングルクオートを 2 つ重ねて使用するため、SESAM インターフェースを使用する全ての PHP スクリプトについて、PHP 設定ディレクティブで php3_magic_quotes_gpc および php3_magic_quotes_sybase を On にしておく方が賢明でしょう。

実行時の考慮

BS2000 モデルの制限のために、ドライバは、Apache サーバがその サーバの子プロセスをフォークした後にのみロード可能となります。 これは、各子プロセスの最初の SESAM リクエストを若干遅くしますが、 その後のアクセスへの応答は最高速度となります。

SESAM 用のメッセージカタログを明示的に定義した場合、そのカタログは ドライバがロードされる度（すなわち、最初の SESAM リクエスト時）にロードされます。BS2000 オペレーティングシステムはメッセージカタログの ロードに成功した後にメッセージを出力します。このメッセージは、Apache の error_log ファイルに送信されます。BS2000 は、現在 このメッセージの出力を停止することができません。このため、 ログを段々と埋めていきます。

SESAM OML PLAM ライブラリと SESAM 設定ファイルは、Web サーバを 実行しているユーザ ID から読み込み可能であることを確認してください。そうでない場合、サーバはドライバをロードすることができず、SESAM 関数をコールすることができません。また、Apache サーバを実行しているユーザ ID にデータベースへのアクセスが許可されている必要があります。 そうでない場合、SESAM データベースハンドラへの接続は失敗します。

カーソル型

SQL "select 型"クエリ用に確保された結果カーソルは、"sequential" または "scrollable" のどちらかとすることが可能です。"scrollable" カーソルで必要なメモリオーバヘッドはより大きいため、デフォルトは "sequential" です。

"scrollable" カーソルを使用した場合、カーソルは結果集合の中で自由に 移動可能です。各 "scrollable" クエリについて、スクロール型の グローバルなデフォルト値 (SESAM_SEEK_NEXT に初期化されます) があり、スクロールオフセットは、 sesam_seek_row() により一回設定されるか、 sesam_fetch_row() によりレコードを取得する度に 設定されるかのどちらかです。"scrollable" カーソルを使用して レコードを取得する際に、スクロール型およびスクロールオフセットの グローバルデフォルト値について次のような後処理が行われます。

スクロール後のカーソルに関する後処理

スクロール型	動作
SESAM_SEEK_NEXT	なし
SESAM_SEEK_PRIOR	なし
SESAM_SEEK_FIRST	スクロール型を SESAM_SEEK_NEXT に設定
SESAM_SEEK_LAST	スクロール型を SESAM_SEEK_PRIOR に設定
SESAM_SEEK_ABSOLUTE	内部オフセット値を自動的に増加させる
SESAM_SEEK_RELATIVE	なし (グローバルデフォルト値 offset を保持する。これにより、例えば、10 個前のレコードを取得すると いったことが可能になります)。

移植上の注意

PHP では配列の添字は (1 よりも) 0 から始まるのが普通なため、 いくつかの調整が SESAM インターフェースで行われています。 ある添字配列がネイティブの SESAM インターフェースで添字 1 から始まる 際には、PHP インターフェースでは最初の添字として 0 を使用します。 例えば、sesam_fetch_row() でカラムを取得する際に、 最初のカラムの添字は 0 であり、その後のカラム番号はカラム数 ($array["count"]) に達するまで (カラム数は含まず) 増えていきます。 SESAM アプリケーションを他の高級言語から PHP に移植する際には、この インターフェース上の変更に留意する必要があります。各 PHP sesam 関数の説明の適切な場所に添字が 0 から始まるという注意が含まれています。

セキュリティの考慮

SESAM データベースへのアクセスが可能な場合、Web サーバのユーザは、 可能な限り小さな権限のみを有している必要があります。多くの データベースでは許可する必要があるのは読み込み権限のみです。 使用する設定に応じて状況に見合ったアクセス権限を追加してください。 決してインターネットからの全てのユーザに全てのデータベースへの 完全な制御権を許可しないでください! データベースを管理する PHP スクリプトへのアクセス制限は、パスワード制限または SSL セキュリティにより制限してください。

他の SQL データベースからの移行

SQL には方言があるため 100% 互換ではありません。他のデータベース インターフェースから SQL アプリケーションを SESAM に移植する際には、いくつかの修正が必要になる可能性があります。次のような典型的な差異に 注意する必要があります。

- ベンダ固有のデータ型 いくつかのベンダ固有のデータ型は、標準 SQL のデータ型で 置換する必要があります (例えば、TEXT は VARCHAR(max. size) で置換可能です)。
- SQL 識別子と同じキーワード SESAM では (標準 SQL と同様に) このような ID は二重引用符で括る (もしくは名前を変える) 必要があります。
- データ型の表示長 SESAM データ型は表示長ではなく、ある精度を有しています。 int(4) (意図された使用法: '9999' までの整数) の代わりに、SESAM は単に 31 ビット長の int を要求します。また、SESAMで利用可能な日付時刻型は次のものだけです。 DATE、TIME(3)、TIMESTAMP(3)。
- ベンダ固有の SQL 型 unsigned、 zerofill、auto_increment 属性 unsigned と zerofill は サポートされません。auto_increment は自動です。 (SESAM に実装された自動インクリメントの利点を活かすために、 "... VALUES(0,...)" の代わりに "INSERT ... VALUES(*, ...)" を使用してください)。
- **int ... DEFAULT '0000'** 数値変数は、文字列定数で初期化するべきではありません。代わりに **DEFAULT 0** を使用してください。SQL データ型 datetime の変数を初期化する際、初期化文字列には、次のように 適当な型キーワードを前に付加する必要があります。 CREATE TABLE exmpl (xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL;);
- **$count = xxxx_num_rows();** いくつかのデータベースは、返される値が著しく不正確であるとしても、 クエリ結果のレコード数を推定または見積もろうとします。 SESAM はクエリ結果のレコード数を実際に取得する前に知ることは ありません。その数が実際に必要な場合には、**SELECT COUNT(...) WHERE ...** を試してみて ください。このクエリは、ヒット数を取得します。2 番目のクエリは (理想的には)結果を返します。
- **DROP TABLE thename;** SESAMでは、**DROP TABLE** においてテーブル名の 後にキーワード RESTRICT または CASCADE のどちらかを後に付加する必要が あります。RESTRICT を指定した場合、(VIEW のような) 依存するオブジェクトがある場合にエラーが返されます。 また、CASCADE を指定した場合、依存する オブジェクトは指定したテーブルから削除されます。

様々な SQL 型を使用する場合の注意

SESAM は現在 BLOB 型をサポートしていません。SESAM の将来の バージョンは、BLOBをサポートする予定です。

PHP インターフェースでは、SQL フィールドを取得する際に次の型変換が 自動的に適用されます。

SQL から PHP への型変換

SQL 型	PHP 型
SMALLINT, INTEGER	integer
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	float
DATE, TIME, TIMESTAMP	string
VARCHAR, CHARACTER	string

レコード全体を取得する際、結果は配列として返されます。空の フィールドには値が入っていないため、個々のフィールド自体の存在を調べる 必要があります（空のフィールドかどうか調べるには、 isset() または empty() を 使用してください)。この手法は、(空のフィールドの表現として型の文字列を使用するよりも）空のフィールドの見え方をユーザが 制御することが可能となります。

SESAM の "複数フィールド" 機能のサポート

SESAM 特有の"複数フィールド" 機能により、複数のフィールドの 配列からなるカラムを使用することが可能です。"複数フィールド" カラムは、次のように作成可能です。

Example#1 "複数フィールド" カラムを作成する

```
CREATE TABLE multi_field_test (
     pkey CHAR(20) PRIMARY KEY,
     multi(3) CHAR(12)
)
```

上のレコードに次のように代入することができます。

Example#2 "複数フィールド" カラムに代入する

```
INSERT INTO multi_field_test (pkey, multi(2..3) )
      VALUES ('Second', <'first_val', 'second_val'>)
```

(この場合のように）先頭の空のサブフィールドは無視され、代入された 値は詰められることに注意してください。このため、上記の例の結果は、multi(2..3) ではなく、multi(1..2) となります。

結果レコードを取得する際、"複数カラム" は "インラインの" 付加カラム のようにアクセスされます。上のにおいて、"pkey" は添字 0 を有し、3 つの "multi(1..3)" カラムは添字 1 から 3 でアクセス可能です。

参考

SESAM 固有の事項について詳細は、» the SESAM/SQL-Server ドキュメント(英語) または » SESAM/SQL-Server ドキュメント(ドイツ語) を参照ください。共にオンラインで参照可能です。もしくは、 適当なマニュアルを使用してください。

---

sesam_affected_rows

(No version information available, might be only in CVS)

sesam_affected_rows — 直近のクエリにより作用されたレコードの数を得る

説明

int **sesam_affected_rows** (string $result_id)

直近のクエリが変更した行の数を取得します。

sesam_affected_rows() 関数は、"即時型" SQL 命令 (INSERT、UPDATE、 DELETE のような更新処理)との組合せで使用された 場合にのみ有用な値を返します。これは、SESAM が "select型" クエリに 関して "作用されたレコード" に関する情報を返さないためです。

パラメータ

result_id

sesam_query() により返された有効な結果 ID。

返り値

result_id に関連するクエリにより作用された レコード数を返します。

例

Example#1 sesam_affected_rows() の例

```
<?php
$result = sesam_execimm("DELETE FROM PHONE WHERE LASTNAME = '" . strtoupper($name) . "'");
if (!$result) {
    /* ... エラー ... */
}
echo sesam_affected_rows($result).
    " 姓が " . $name . " のエントリが削除されました。\n";
?>
```

参考

- sesam_query()
- sesam_execimm()

---

sesam_commit

(No version information available, might be only in CVS)

sesam_commit — SESAM データベースへの待機中の更新処理をコミットする

説明

bool **sesam_commit** (void)

全ての待機中のデータベースへの更新処理をコミットします。

他のデータベースのように"自動コミット"機能がないため、事故により データが失われる可能性があることに注意してください。カレントの スクリプト実行終了時 (あるいは sesam_disconnect() をコールした際) にコミットされていないデータは、暗黙の sesam_rollback() コールにより破棄されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 SESAM データベースの更新をコミットする

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
     if (!sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
         die("insert 失敗");
     if (!sesam_commit())
         die("commit 失敗");
}
?>
```

参考

- sesam_rollback()

---

sesam_connect

(No version information available, might be only in CVS)

sesam_connect — SESAM データベース接続をオープンする

説明

bool **sesam_connect** (string $catalog , string $schema , string $user)

SESAM データベースハンドラ タスクへの接続を確立します。接続は、最初の起動時にのみ設定された SESAM OML PLAM ライブラリから実際にドライバがロードされるという 意味で常に"持続的"です。この後のコールではドライバは再利用され、 指定したカタログ、スキーマ、ユーザが直ちに使用されます。

パラメータ

catalog

データベースを作成する際、"catalog" 名は、 SESAM 設定ディレクティブ **//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)** で 指定します。

schema

"schema" には選択したデータベーススキーマを 指定します (SESAM ハンドブックを参照)。

user

"user" 引数には、この "catalog" / "schema" の組にアクセス可能なユーザの一人を 指定します。"user" は、システムのユーザ ID と HTTP ユーザ/パスワード保護から共に完全に独立していることに 注意してください。SESAM 設定にのみ使用されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 SESAM データベースへの接続

```
<?php
if (!sesam_connect ("mycatalog", "myschema", "otto")) {
    die("SESAM に接続できません");
}
?>
```

参考

- sesam_disconnect()

---

sesam_diagnostic

(No version information available, might be only in CVS)

sesam_diagnostic — 直近の SESAM コールに関するステータス情報を返す

説明

array **sesam_diagnostic** (void)

直近の SESAM コールについてのステータス情報を返します。

返り値

直近の SQL クエリ/命令/コマンドに関するステータスおよびエラーコードを 連想配列として返します。配列の要素は次のようになります。

sesam_diagnostic() により返されるステータス情報

要素	内容
$array["sqlstate"]	5 桁の SQL リターンコード (SQLSTATE の値の説明については、 SESAM マニュアルを参照ください)。
$array["rowcount"]	直近の update/insert/delete クエリで作用されたレコードの数 ("即時型"命令の後でのみ設定されます)。
$array["errmsg"]	"可読な" エラーメッセージ文字列 (エラーの後でのみ設定されます)。 only)
$array["errcol"]	以前のエラーのエラーカラム番号 (0 から始まり、未定義の場合は -1。エラーの後でのみ設定されます)。
$array["errlin"]	前のエラーのエラー行番号 (0 から始まり、未定義の場合は -1。エラーの後でのみ設定されます)。

例

次の例では、問題がある SQL 命令を含み、エラー位置を指す構文エラー (E SEW42AE ILLEGAL CHARACTER) が表示されます。

Example#1 エラー位置を付けて SESAM エラーメッセージを表示

```
<?php
// フォーマットされたエラーメッセージを出力する関数
// SQL 命令における構文エラーの場所を表示
function PrintReturncode($exec_str)
{
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        --$colspan;
    if ($err["errcol"] == -1)
        --$colspan;
    if ($err["rowcount"] == 0)
        --$colspan;
    echo "<table border=¥"1¥">¥n";
    echo "<tr><th colspan=¥"" . $colspan . "¥"><span class=¥"spanred¥">ERROR:</span> ".
        htmlspecialchars($err["errmsg"]) . "</th></tr>¥n";
    if ($err["errcol"] >= 0) {
        echo "<tr><td colspan=¥"" . $colspan . "¥"><pre>¥n";
        $errstmt = $exec_str . "¥n";
        for ($lin=0; $errstmt != ""; ++$lin) {
            if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
                if (!($i = strchr($errstmt, "¥n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                if ($line != "¥n")
                    echo htmlspecialchars($line);
            } else {
                if (! ($i = strchr ($errstmt, "¥n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen ($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                for ($col=0; $col < $err["errcol"]; ++$col) {
                    echo (substr($line, $col, 1) == "¥t") ? "¥t" : ".";
                }
                echo "<span class=¥"spanred¥">¥¥</span>¥n";
                echo "<span class=¥"normal¥">" . htmlspecialchars($line) . "</span>";
                for ($col=0; $col < $err["errcol"]; ++$col) {
                    echo (substr ($line, $col, 1) == "¥t") ? "¥t" : ".";
                }
                echo "<span class=¥"spanred¥">/</span>¥n";
            }
        }
        echo "</pre></td></tr>¥n";
    }
    echo "<tr>¥n";
    echo "  <td>sqlstate=" . $err["sqlstate"] . "</td>¥n";
    if ($err["errlin"] != -1)
        echo "  <td>errlin=" . $err["errlin"] . "</td>¥n";
    if ($err["errcol"] != -1)
        echo "  <td>errcol=" . $err["errcol"] . "</td>¥n";
    if ($err["rowcount"] != 0)
        echo "  <td>rowcount=" . $err["rowcount"] . "</td>¥n";
    echo "</tr>¥n";
    echo "</table>¥n";
}

if (!sesam_connect ("mycatalog", "phoneno", "otto"))
  die ("cannot connect");

$stmt = "SELECT * FROM phone¥n" .
```

```
        " WHERE@ LASTNAME='KRAEMER'¥n" .
        " ORDER BY FIRSTNAME";
if (!($result = sesam_query ($stmt)))
    PrintReturncode ($stmt);
?>
```

参考

- sesam_errormsg()

---

sesam_disconnect

(No version information available, might be only in CVS)

sesam_disconnect — SESAM 接続から切り離す

説明

bool **sesam_disconnect** (void)

(実際に接続を断にしたり、ドライバをアンロードすることなく) SESAM データベースへの論理リンクを閉じます。

オープンされた接続はスクリプトの実行終了時に閉じられるため、 この関数は通常不要です。コミットされていないデータは、暗黙のうちに sesam_rollback() が実行されるため、破棄されます。

sesam_disconnect() は、持続的なリンクを閉じず、 現在、定義されている "catalog" 、 "schema" 、"user" の 組を無効にするだけです。このため、 **sesam_disconnect()** の後でコールされた SESAM 関数は失敗します。

返り値

常に **TRUE** を返します。

例

Example#1 SESAM 接続を閉じる

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    /* ... クエリなどの処理 ... */
    sesam_disconnect();
}
?>
```

参考

- sesam_connect()

---

sesam_errormsg

(No version information available, might be only in CVS)

sesam_errormsg — 直近の SESAM コールのエラーメッセージを返す

説明

string **sesam_errormsg** (void)

直近の SESAM エラーに関連する SESAM エラーメッセージを取得します。

返り値

直近の SESAM エラーに関連する SESAM エラーメッセージを返します。 空の文字列の場合は、エラーが発生しなかったことを表します。

例

Example#1 sesam_errormsg() の例

```
<?php
if (!sesam_execimm($stmt)) {
  echo sesam_errormsg() . "<br />¥n";
}
?>
```

参考

- sesam_diagnostic()

---

sesam_execimm

(No version information available, might be only in CVS)

sesam_execimm — SQL 命令を直ちに実行する

説明

string **sesam_execimm** (string $query)

"直接" 文を実行します。

sesam_affected_rows() 関数により取得可能な affected_rows の値を設定します。

sesam_query() は、"即時型" および "select型" クエリの両方を処理することができることに注意してください。実行される 命令の型が事前に分かっている場合にのみ **sesam_execimm()** を使用してください。SELECT 型 クエリを使用しようとすると、**sesam_execimm()** は $err["sqlstate"] == "42SBW" を返します。

パラメータ

query

命令（すなわち、UPDATE、 INSERT、DELETEのような結果を返さない命令や入出力変数がない命令）を "直ちに"実行します。"select 型" クエリは、 **sesam_execimm()** で使用することはできません。

返り値

成功時にSESAM "結果 ID"、エラー時に **FALSE** を返します。

返される "結果ID" は、何かを取得する際に使用することはできず、 sesam_affected_rows() を実行する際に使用します。 この ID は、sesam_query() 関数と対称性を保つため にのみ返されます。

例

Example#1 sesam_execimm() Example

```
<?php
$stmt = "INSERT INTO mytable VALUES ('one', 'two')";
$result = sesam_execimm($stmt);
$err = sesam_diagnostic();
echo "sqlstate = " . $err["sqlstate"] . "¥n".
      "変更された行数 = " . $err["rowcount"] . " == " .
      sesam_affected_rows($result) . "¥n";
?>
```

参考

- sesam_query()
- sesam_affected_rows()

---

sesam_fetch_array

(No version information available, might be only in CVS)

sesam_fetch_array — 連想配列としてレコードを 1 件取得する

説明

array **sesam_fetch_array** (string $result_id [, int $whence [, int $offset]])

sesam_fetch_array() は、 sesam_fetch_row() の連想配列版です。データを 結果配列の数値添字に保存する代わりに、データをフィールド名をキーとして連想配列に保存します。

sesam_fetch_array() は、 指定した結果 ID が指す結果から 1 件分のレコードを取得します。 レコードは、連想配列として返されます。各結果カラムは、そのカラム（またはフィールド）名に等しい連想配列の要素に保存されます。カラム名は、 小文字に変換されます。

フィールド名がないカラム（例えば、数値演算の結果）および空の フィールドは、配列に保存されません。また、同じカラム名にが 2 つ以上ある場合、最後のカラムが優先されます。この場合、 sesam_fetch_row() をコールするかそのカラムへの エイリアスを作成してください。

SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2

特別な処理により "複数フィールド" カラム（もしくは同じカラム名を 有するカラム）を取得することが可能になります。"複数フィールド" の 各カラムに関して、添字名は、文字列 "(n)" を付加することにより 構築されます。ただし、n は複数フィールドのカラムの副添字であり、 1 から宣言済みの反復数までの範囲となります。クエリ構文で使用される 表記に一致させるために添字はゼロを基準にしていません。 次のように宣言されているあるカラムの場合、

CREATE TABLE ... (... MULTI(3) INT)

個々の "複数フィールド" カラムに関して連想添字は、それぞれ "multi(1)","multi(2)", "multi(3)" になります。

sesam_fetch_array() を続けてコールした場合、 結果集合の次の（スクロール属性に応じて前または n 番目の前/後）レコードまたはレコードがもうない場合に **FALSE** を返します。

パラメータ

result_id

sesam_query() が返す結果 ID。

whence

whence は "スクロール型" カーソルで 取得処理を行うためのオプションパラメータで、次のような定義済みの 定数を設定することが可能です。

"whence" パラメータで有効な値

値	定数	意味
0	SESAM_SEEK_NEXT	連続的に読み込む（取得後、内部デフォルト値は SESAM_SEEK_NEXT に設定されます）。
1	SESAM_SEEK_PRIOR	連続的に後向きに読み込む（取得後、内部デフォルト値は SESAM_SEEK_PRIOR に設定されます）。
2	SESAM_SEEK_FIRST	最初のレコードに移動する（取得後、デフォルト値は SESAM_SEEK_NEXT に設定されます）。
3	SESAM_SEEK_LAST	最後のレコードに移動する（取得後、デフォルト値は SESAM_SEEK_PRIOR に設定されます）。
4	SESAM_SEEK_ABSOLUTE	offset（0 が先頭。取得後、内部 デフォルト値は SESAM_SEEK_ABSOLUTE に設定されます。内部のオフセット値は、自動的に増加します）で指定した絶対レコード番号に移動します。
5	SESAM_SEEK_RELATIVE	カレントのスクロール位置に対して相対位置に移動します。ただし、 offset は正または負の値を指定可能です。

このパラメータは、"スクロール型" カーソルでのみ有効です。

"スクロール型" カーソルを使用している場合、カーソルは結果集合を 自由に移動可能です。whence パラメータが 省略された場合、スクロールの型には、グローバルなデフォルト値（SESAM_SEEK_NEXT に初期化されており、 sesam_seek_row() で設定可能です）が使用されます。 whence が指定された場合、その値は グローバルデフォルト値に置換されます。

offset

offset は、オプションのパラメータであり、 whence が SESAM_SEEK_RELATIVE または SESAM_SEEK_ABSOLUTE のどちらかである場合にのみ 評価されます（そして必要とされます）。このパラメータは、 "スクロール型" カーソルでのみ有効です。

返り値

取得したレコードに対応する配列またはレコードがもうない場合には **FALSE** を返します。

例

Example#1 SESAM 配列の取得

```
<?php
$result = sesam_query("SELECT * FROM phone¥n" .
                      "   WHERE LASTNAME='" . strtoupper($name) . "'¥n".
                      "   ORDER BY FIRSTNAME", 1);
if (!$result) {
    /* ... エラー ... */
}
// テーブルの表示
echo "<table border=¥"1¥">¥n";
while (($row = sesam_fetch_array($result)) && count($row) > 0) {
     echo "<tr>¥n";
     echo "<td>" . htmlspecialchars($row["firstname"]) . "</td>¥n";
     echo "<td>" . htmlspecialchars($row["lastname"]) . "</td>¥n";
     echo "<td>" . htmlspecialchars($row["phoneno"]) . "</td>¥n";
     echo "</tr>¥n";
}
echo "</table>¥n";
sesam_free_result($result);
?>
```

参考

- sesam_fetch_row()

---

sesam_fetch_result

(No version information available, might be only in CVS)

sesam_fetch_result — クエリ結果の全てあるいは一部を返す

説明

mixed **sesam_fetch_result** (string $result_id [, int $max_rows])

結果を取得します。オプションで、行数を max_rows に制限します。

パラメータ

result_id

sesam_query() が返す結果 ID。

max_rows

大きなクエリで使用されるメモリの最大値は、巨大なものになる可能性が あることに注意してください。結果が利用可能な全メモリを消費しない ことが確実である場合以外は、返されるレコード数の最大値を制限する ために max_rows パラメータを使用してください。

返り値

クエリ結果のエントリを種々の型の配列として返します。オプションで 最大レコード数を max_rows に制限することが 可能です。レコード番号およびカラム番号は共に 0 から始まることに 注意してください。

sesam_fetch_result() により返された種々の結果集合

配列要素	定数
int $arr["count"]	結果集合のカラム数 ("即時型"クエリの場合に 0)
int $arr["rows"]	結果集合のレコード数 (0 と max_rows の間)

配列要素	定数
bool $arr["truncated"]	レコード数が *max_rows* 以上の場合に **TRUE**、そうでない場合に **FALSE**。これが **TRUE** の場合でも、結果エントリがもうないために次の **sesam_fetch_result()** はレコードを返さない 可能性があります。
mixed $arr[col][row]	レコード (*row*) および カラム (*col*) にある全てのフィールドの 結果データ (整数のレコード番号 *row* は 0 から *$arr["rows"]-1* の間であり、 *col* は 0 から *$arr["count"]-1* の間です)。フィールドは 空である可能性があり、このため、PHP の isset() 関数を使用してフィールドが 存在するかどうか確認する必要があります。返されるフィールドの型は、 そのカラムの SQL 型宣言 (適用される変換については SESAM 概要参照) に依存します。 SESAM "複数フィールド" は "インライン" であり、カラムの番号と 同様に処理されます。

参考

- *sesam_fetch_array()*
- *sesam_fetch_row()*
- *sesam_query()*

---

sesam_fetch_row

(No version information available, might be only in CVS)

sesam_fetch_row — 1 件分のレコードを配列として取得する

説明

array ***sesam_fetch_row*** *(string $result_id [, int $whence [, int $offset]])*

指定した結果 ID が指す 結果から 1 件分のレコードのデータを取得します。レコードは、(0 から $array["count"]-1 を 添字とした) 配列として返されます。フィールドは空である可能性が あるため、PHP の isset() 関数を使用して フィールドの存在を確認する必要があります。返されるフィールドの型は、 そのカラムを宣言した SQL 型に依存します (適用される変換については、 SESAMの概要 を参照ください)。 SESAM "複数フィールド" は "インライン化" されており、連続する カラムのように処理されます。

*結果集合のカラム数が連想配列の要素 $array["count"] で返されます。 いくつかのカラムは空である可能性があるため、 **sesam_fetch_row()** により返された結果レコードに 対して count() 関数を使用することはできません。*

***sesam_fetch_row()** を連けてコールした場合、 結果集合の次の (スクロール属性に応じて前または n 番目の前/後) レコードまたはレコードがもうない場合に **FALSE** を返します。*

パラメータ

result_id

sesam_query() が返す結果 ID。

whence

whence は "スクロール型" カーソルで 取得処理を行うためのオプションパラメータで、次のような定義済みの 定数を設定することが可能です。

"whence" パラメータで有効な値

値	定数	意味
0	**SESAM_SEEK_NEXT**	連続的に読み込む (取得後、内部デフォルト値は *SESAM_SEEK_NEXT* に設定されます)。
1	**SESAM_SEEK_PRIOR**	連続的に後向きに読み込む (取得後、内部デフォルト値は *SESAM_SEEK_PRIOR* に設定されます)。
2	**SESAM_SEEK_FIRST**	最初のレコードに移動する (取得後、デフォルト値は *SESAM_SEEK_NEXT* に設定されます)。
3	**SESAM_SEEK_LAST**	最後のレコードに移動する (取得後、デフォルト値は *SESAM_SEEK_PRIOR* に設定されます)。
4	**SESAM_SEEK_ABSOLUTE**	*offset* (0 が先頭。取得後、内部 デフォルト値は *SESAM_SEEK_ABSOLUTE* に設定されます。内部のオフセット値は、自動的に増加します) で指定した絶対レコード番号に移動します。
5	**SESAM_SEEK_RELATIVE**	カレントのスクロール位置に対して相対位置に移動します。ただし、 *offset* は正または負の値を指定可能です。

このパラメータは、"スクロール型" カーソルでのみ有効です。

"スクロール型" カーソルを使用している場合、カーソルは結果集合を 自由に移動可能です。whence パラメータが 省略された場合、スクロールの型には、グローバルなデフォルト値 (SESAM_SEEK_NEXT に初期化されており、 sesam_seek_row() で設定可能です) が使用されます。 whence が指定された場合、その値は グローバルデフォルト値に置換されます。

offset

whence が SESAM_SEEK_RELATIVE または SESAM_SEEK_ABSOLUTE のどちらかである場合にのみ 評価されます (そして必要とされます)。このパラメータは、 "スクロール型" カーソルでのみ有効です。

返り値

*取得したレコードを含む配列、 またはもうレコードがない場合に **FALSE** を返します。*

例

Example#1 SESAM レコードを取得

```
<?php
$result = sesam_query("SELECT * FROM phone¥n" .
                      "   WHERE LASTNAME='" . strtoupper($name) . "'¥n" .
                      "   ORDER BY FIRSTNAME", 1);
if (!$result) {
    /* ... エラー ... */
}
// 逆順にテーブルを出力する
```

```
echo "<table  border=¥"1¥">¥n";
$row = sesam_fetch_row($result, SESAM_SEEK_LAST);
while (is_array($row)) {
     echo "<tr>¥n";
     for ($col = 0; $col < $row["count"]; ++$col) {
          echo "<td>" . htmlspecialchars($row[$col]) . "</td>¥n";
     }
     echo "</tr>¥n";
    // 暗黙のうちに SESAM_SEEK_PRIOR が使用される
    $row = sesam_fetch_row($result);
}
echo "</table>¥n";
sesam_free_result($result);
?>
```

参考

- sesam_fetch_array()
- sesam_fetch_result()

---

sesam_field_array

(No version information available, might be only in CVS)

sesam_field_array — 結果の個々のカラムに関するメタ情報を返す

説明

array **sesam_field_array** (string $result_id)

result_id に関連したクエリの後で、結果の 個々のカラムに関するメタ情報(カラム名、型、精度、...)を有する 連想/添字配列を返します。

パラメータ

result_id

sesam_query() が返す結果 ID。

返り値

sesam_field_array() が返す結果セット

配列要素	内容
int $arr["count"]	結果集合におけるカラム数の合計 (または"即時型"クエリの場合に 0)。SESAM "複数フィールド" はインライン化されており、対応する カラム番号と同様に処理されます。
string $arr[col]["name"]	カラム (*col*) のカラム名、ただし、 *col* は、0 から *$arr["count"]-1* の間です。返される値は、 (動的に計算されるカラムの場合) 空の文字列となる可能性があります。 SESAM "複数フィールド" は"インライン化"されており、カラム番号と 同様に同じカラム名で処理されます。
string $arr[col]["count"]	属性 "count" は、カラムが"複数フィールド"として宣言された 場合の繰り返し数を記述します。通常、"count"属性は 1 です。 しかし、"複数フィールド" カラムの最初のカラムは繰返し数を 有します ("複数フィールド" の 2 番目以降のカラムは 1 という "count"属性を有します)。この属性は、結果集合の中で "複数 フィールド" を検出するために使用可能です。"count" 属性の 使用例については、sesam_query() の説明に 示されている例を参照ください。
string $arr[col]["type"]	カラム (*col*) に関するデータの PHP 変数型、 ただし、*col* は 0 と *$arr["count"]-1* の間です。値は、結果の SQL 型に基づき以下のどれかとなります。 • integer • float • string SESAM "複数フィールド" は "インライン化" されており、 同じカラム数を有し、同じ PHP 型の場合と同様に処理されます。
string $arr[col]["sqltype"]	カラム (*col*) に関するカラムデータの SQL 変数型。ただし、*col* は 0 から *$arr["count"]-1* の間になります。返される 値は、結果の SQL 型を記述する次のどれかとなります。 • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" SESAM "複数フィールド" は "インライン化" されており、 同じカラム数、同じ PHP 型の場合と同様に処理されます。
string $arr[col]["length"]	カラム (*col*) のSQL 変数の SQL "length" 属性。 ただし、*col* は、0 から *$arr["count"]-1* の間です。"length" 属性は、 文字列変数の (最大) 長さを指定するために "CHARACTER" および "VARCHAR" SQL 型を指定して使用されます。 SESAM "複数フィールド" は "インライン化" されており、 同じカラム数、同じ PHP 型の場合と同様に処理されます。
string $arr[col]["precision"]	カラム (*col*) にある SQL 変数の "precision" 属性。ただし、*col* は、0 から *$arr["count"]-1* の間です。"precision" 属性は、数値および時間データ型で使用されます。 SESAM "複数フィールド" は "インライン化" されており、 同じカラム数、同じ PHP 型の場合と同様に処理されます。
string $arr[col]["scale"]	カラム (*col*) にある SQL 変数の "scale" 属性。 ただし、*col* は、0 から *$arr["count"]-1* の間です。"scale" 属性は、 数値データ型で使用されます。 SESAM "複数フィールド" は "インライン化" されており、 同じカラム数、同じ PHP 型の場合と同様に処理されます。

参考

- *sesam_query()*

---

sesam_field_name

(No version information available, might be only in CVS)

sesam_field_name — 結果集合のカラム名を返す

説明

*int **sesam_field_name** (string $result_id , int $index)*

フィールド名を取得します。

パラメータ

result_id

sesam_query() が返す結果 ID。

index

カラムの添字。SESAM のように 1 から始まらず、0 から始まります。

返り値

結果集合の中のフィールド名(すなわち、カラム名)またはエラーの際に **FALSE** を返します。

"即時型"クエリまたは動的なカラムの場合、空の文字列が返されます。

参考

- *sesam_field_array()* も参照ください。この関数は、 カラム名および型により簡単にアクセスするインターフェースを提供し、 "複数フィールド"の検出が可能になります。

---

sesam_free_result

(No version information available, might be only in CVS)

sesam_free_result — クエリに関するリソースを開放する

説明

bool ***sesam_free_result*** *(string $result_id)*

クエリのリソースを開放します。

パラメータ

result_id

sesam_query() が返す結果 ID。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

sesam_num_fields

(No version information available, might be only in CVS)

sesam_num_fields — 結果集合のフィールド/カラム数を返す

説明

int ***sesam_num_fields*** *(string $result_id)*

"select型"クエリで *sesam_query()* をコールした後、 この関数により結果のカラム数を得ることが可能です。

"即時型"命令の場合、値 0 が返されます。SESAM "複数フィールド" カラムは、 それぞれの次元毎に数えられます。すなわち、"複数フィールド" の 3 カラムはカラム 3 つとして数えられます。

パラメータ

result_id

sesam_query() が返す結果 ID。

返り値

result_id が指す結果セットの 全体のカラム（またはフィールド）の数を記述する整数を返します。 エラー時には **FALSE** を返します。

参考

- *sesam_query()*
- *sesam_field_array()* で "複数フィールド" カラムと標準カラムを区別します。

---

sesam_query

(No version information available, might be only in CVS)

sesam_query — SESAM SQL クエリを実行し、結果を準備する

説明

```
string sesam_query ( string $query [, bool $scrollable ] )
```

sesam_query() は、クエリをサーバ上の現在 アクティブなデータベースに送信します。この関数は、"即時" SQL 命令 および "select型" クエリの両方を実行可能です。"即時型" 命令を 実行した場合はカーソルは確保されず、この後に sesam_fetch_row() または sesam_fetch_result() をコールしても空の結果（結果の終端を示すカラム数 0）が返されます。

"スクロール型"カーソルを使用している場合、カーソルは結果集合上を 自由に移動可能です。各"スクロール型"クエリに関して、スクロール型の グローバルデフォルト値 （SESAM_SEEK_NEXT に 初期化されています） と sesam_seek_row() により 一度設定するか、sesam_fetch_row() を使用して レコードを取得する度に設定するスクロールオフセットがあります。

"即時型"命令の場合、作用されたレコードの数が保存されます。この数は、 sesam_affected_rows() 関数で取得可能です。

パラメータ

query

クエリ文。

scrollable

"select型" 命令の場合、 結果記述子および（オプションのパラメータ scrollable の設定によりスクロール型または 連続型の）カーソルが確保されます。scrollable が省略された場合、カーソルは連続型となります。

返り値

成功時に SESAM "結果 ID"、またはエラー時に **FALSE** を返します。 この結果 ID を、その他の関数で使用します。

例

Example#1 "phone" テーブルの全てのレコードを HTML テーブルとして表示する

```
<?php
if (!sesam_connect("phonedb", "demo", "otto"))
    die("接続できません");
$result = sesam_query("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die ($err["errmsg"]);
}
echo "<table border>¥n";
// 結果の上にカラム名をヘッダとして表示します
if ($cols = sesam_field_array($result)) {
    echo "<tr><th colspan=" . $cols["count"] . ">Result:</th></tr>¥n";
    echo "<tr>¥n";
    for ($col = 0; $col < $cols["count"]; ++$col) {
        $colattr = $cols[$col];
        /* SESAM の "Multiple Fields" について、列を連結します */
        if ($colattr["count"] > 1) {
            echo "<th colspan=¥"" . $colattr["count"] . "¥">" . $colattr["name"] .
                "(1.." . $colattr["count"] . ")</th>¥n";
            $col += $colattr["count"] - 1;
        } else
            echo "<th>" . $colattr["name"] . "</th>¥n";
    }
    echo "</tr>¥n";
}

do {
    // 最大 100 行までに分割して結果を取得します
    $ok = sesam_fetch_result($result, 100);
    for ($row=0; $row < $ok["rows"]; ++$row) {
        echo "  <tr>¥n";
        for ($col = 0; $col < $ok["cols"]; ++$col) {
            if (isset($ok[$col][$row])) {
                echo "<td>" . $ok[$col][$row] . "</td>¥n";
            } else {
                echo "<td>-empty-</td>¥n";
            }
        }
        echo "</tr>¥n";
    }
} while ($ok["truncated"]); //  データがなくなるまで続けます

echo "</table>¥n";
// 結果 id を開放します
sesam_free_result($result);
?>
```

参考

- sesam_fetch_row()
- sesam_fetch_result()

---

sesam_rollback

(No version information available, might be only in CVS)

sesam_rollback — SESAM データベースに対する待機中の更新を破棄する

説明

```
bool sesam_rollback ( void )
```

sesam_rollback() は、データベースへの待機中の 更新を破棄します。結果カーソルと結果記述子も変更されます。

スクリプトの終了時に、sesam_disconnect() 関数 から暗黙のうちに **sesam_rollback()** が実行され、 データベースの待機中の全ての変更は破棄されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 SESAM データベースへの更新を破棄する

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm ("INSERT INTO othertable VALUES (*, 'Another Test', 1)")) {
        sesam_commit();
    } else {
        sesam_rollback();
    }
}
?>
```

参考

- sesam_commit()

---

sesam_seek_row

(No version information available, might be only in CVS)

sesam_seek_row — 連続的に取得する際にスクロール可能なカーソルモードに設定する

説明

bool **sesam_seek_row** (string $result_id , int $whence [, int $offset])

スクロール可能なカーソルを、それ以降のフェッチのために設定します。

パラメータ

result_id

result_id は有効な結果 ID です (select 型のクエリで、かつ、sesam_query() を コールした際に"スクロール可能な"カーソルが要求された場合のみ)。

whence

whence は、この後の"スクロール型"カーソルの 取得処理で使用されるスクロールの型を指定する、スクロール型の グローバルなデフォルト値を設定し、次のような定義済みの定数を 設定します。

"whence" パラメータの有効な値

値	定数	意味
0	SESAM_SEEK_NEXT	連続的に読み込む
1	SESAM_SEEK_PRIOR	連続的に後向きに読む
2	SESAM_SEEK_FIRST	最初のレコードに移動する (取得後、デフォルト値は SESAM_SEEK_NEXT に設定されます)
3	SESAM_SEEK_LAST	最後のレコードに移動する (取得後、デフォルト値は SESAM_SEEK_PRIOR に設定されます)
4	SESAM_SEEK_ABSOLUTE	offset (0 が先頭。取得後、 内部デフォルト値は SESAM_SEEK_ABSOLUTE に設定されます。内部のオフセット値は、自動的に増加します) で指定した絶対レコード番号に移動します。
5	SESAM_SEEK_RELATIVE	カレントのスクロール位置に対して相対位置に移動。ただし、 offset は正または負の値を指定可能です (これは、以降の取得の際のデフォルトの "offset" も設定します)。

offset

オプションのパラメータであり、 whence が SESAM_SEEK_RELATIVE または SESAM_SEEK_ABSOLUTE のどちらかである場合にのみ 評価されます (また必要とされます)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

sesam_settransaction

(No version information available, might be only in CVS)

sesam_settransaction — SESAM トランザクションパラメータを設定する

説明

bool **sesam_settransaction** (int $isolation_level , int $read_only)

(SESAM 設定ファイル で設定される) トランザクションパラメータ "isolation level" と "read-only" のデフォルト値を上書きします。これは、連続するクエリを 最適化し、データベースの統一性を保証するためです。上書きされた値は、 次のトランザクションでのみ使用されます。 詳細

な説明は SESAM ハンドブックを参照ください。

この関数は、トランザクションを開始する前にのみコールすることが可能です。 、既にトランザクションが開始されている場合には、コールすることができません。

sesam_settransaction() により設定された値は、 SESAM 設定ファイルで 指定されたデフォルト設定を上書きします。

パラメータ

isolation_level

isolation_level パラメータの有効な値

値	定数	意味
1	SESAM_TXISOL_READ_UNCOMMITTED	Read Uncommitted
2	SESAM_TXISOL_READ_COMMITTED	Read Committed
3	SESAM_TXISOL_REPEATABLE_READ	Repeatable Read
4	SESAM_TXISOL_SERIALIZABLE	Serializable

read_only

read_only パラメータの有効な値

値	定数	意味
0	SESAM_TXREAD_READWRITE	Read/Write
1	SESAM_TXREAD_READONLY	Read-Only

返り値

settransaction() 処理が成功して、 有効な場合に **TRUE**、その他の場合に **FALSE** を返します。

例

Example#1 SESAM トランザクションパラメータの設定

```
<?php
sesam_settransaction  (SESAM_TXISOL_REPEATABLE_READ,
                       SESAM_TXREAD_READONLY);
?>
```

目次

PostgreSQL セッション保存ハンドラ

導入

注意: この拡張モジュールは Windows 環境では利用できません。

このモジュールは、» PostgreSQL をストレージとして使用する session モジュール用にセッション保存ハンドラを提供するものです。 ユーザレベルの セッション保存関数 - session_set_save_handler() を使用することも可能ですが、 このモジュールは C で書かれており、PHP で書いた保存ハンドラより約 2 倍 高速です。

PgSQL セッション保存ハンドラは あらゆる規模の Web サイトを対象に 設計されており、いくつかの進んだ機能を持っています。

- セッションテーブルを自動的に作成します。
- セッションテーブルの vacuum を自動的に行います。
- 効率的なガベージコレクションを行います。

- 複数の PostgreSQL サーバをサポートします。
- データベースサーバの自動フェイルオーバー（スイッチング）を行います。
- 複数の PostgreSQL サーバを稼動させている場合、自動的に ロードバランシングが行われます。
- UPDATE のショートカット機能を有します。

要件

すくなくとも PHP >= 4.3.0 、およびデータベースサーバとして PostgreSQL >=7.2.0 が必要です。また、PostgreSQL 7.2.0 以降の libpq （とそのヘッダファイル） および » libmm （とそのヘッダファイル） が必要です。

インストール手順

簡単なインストール手順

- tar.gz アーカイブを php4/ext に展開します （最新の正式リリースは、SourceForge の » PHP Form Extension Project にあります）。
- 新しいディレクトリを session_pgsql のように呼ぶと すると、それを session_pgsql という名前にする 必要があります（PHP モジュールとして構築する場合を除きます）。
- PHP 4 では **./buildconf** を実行します。
- **configure** --with-session-pgsql （およびその他のオプション）を実行します。
- **make; make install**

実行時設定

php.ini の設定により動作が変化します。

PostgreSQL セッション保存ハンドラは、まだ開発中です。設定の詳細に ついては、配布物に含まれる README ファイルを参照して下さい。

テーブル定義

セッションテーブルの定義

```
CREATE TABLE php_session (
   sess_id              text,
   sess_name            text,
   sess_data            text,
   sess_created         integer,
   sess_modified        integer,
   sess_expire          integer,
    sess_addr_created   text,
    sess_addr_modified  text,
   sess_counter         integer,
   sess_error           integer,
   sess_warning         integer,
   sess_notice          integer,
    sess_err_message    text,
   sess_custom          text
);

CREATE INDEX php_session_idx ON php_session USING BTREE (sess_id);
```

警告

HASH 形式の INDEX を使用する際は、 サーバの負荷が非常に高くなった際にデッドロックの 問題が発生することがあります。通常の操作ではデッドロックが起こることは まずありませんが、起こることもありえます。インデックスには HASH 形式を使用しないでください。

すべてのフィールドが定義されている限り、セッションテーブルを 変更することも可能です。

アプリケーション変数テーブルの定義

```
CREATE TABLE php_app_vars (
   app_modified         integer,
   app_name             text,
   app_vars             text
);
```

連絡先

現時点で、この拡張モジュールの開発者は、開発を継続する時間の余裕 はありません。将来的により多くの機能を実装する予定です。

コメント、バグ修整、拡張、またはこのモジュールの開発を援助したい 場合には » yohgaki@php.net までメールをください。いかなる援助も歓迎します。

---

session_pgsql_add_error

(No version information available, might be only in CVS)

session_pgsql_add_error — エラーカウントを加算し、直近のエラーメッセージを設定する

説明

bool **session_pgsql_add_error** (int $error_level [, string $error_message])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

error_level

error_message

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *session_pgsql_get_error()*

---

session_pgsql_get_error

(No version information available, might be only in CVS)

session_pgsql_get_error — エラーの数および直近のエラーメッセージを返す

説明

array ***session_pgsql_get_error*** *([bool $with_error_message])*

エラーの数を取得し、またオプションでエラーメッセージも取得します。

パラメータ

with_error_message

TRUE *に設定すると、各エラーのエラーメッセージも返されます。*

返り値

エラーの数を配列で返します。

参考

- *session_pgsql_add_error()*

---

session_pgsql_get_field

(No version information available, might be only in CVS)

session_pgsql_get_field — カスタムフィールドの値を取得する

説明

string ***session_pgsql_get_field*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- *session_pgsql_set_field()*

---

session_pgsql_reset

(No version information available, might be only in CVS)

session_pgsql_reset — セッションデータベースサーバとの接続をリセットする

説明

bool ***session_pgsql_reset*** *(void)*

セッションデータベースサーバとの接続をリセットします。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

---

session_pgsql_set_field

(No version information available, might be only in CVS)

session_pgsql_set_field — カスタムフィールドの値を設定する

説明

bool **session_pgsql_set_field** (string $value)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

value

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- session_pgsql_get_field()

---

session_pgsql_status

(No version information available, might be only in CVS)

session_pgsql_status — 現在の保存ハンドラの状態を得る

説明

array **session_pgsql_status** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

セッション処理関数(session)

導入

PHPのセッションサポート機能は、複数回のアクセスを通じて特定のデー タを保持する手段を実現するものです。これにより、よりカスタマイズさ れたアプリケーションを構築し、自分の Web サイトのアピール度を増加 させることが可能となります。

Web サイトの訪問者にはセッションIDというセッションIDと呼ばれるユニークなIDが割りつけられ ます。このIDは、ユーザー側にクッキーとして保存するか、または、URL に埋め込みます。

セッションサポート機能により、任意の数の変数をリクエスト間で受けわたせる ようになります。来訪者がサイトにアクセスした際、 PHP は特定のセッションIDがリクエストとともに送信されているかどうかを（ session.auto_startが1の場合は）自動的に、または （session_start() により明示的な、あるいは session_register() により暗黙の）要求を受けて確認します。 このIDが送信されている場合には、以前保存された変数が再現されます。

警告

session.auto_startをonとした場合、 オブジェクトをセッション変数に代入することができなくなります。 これは、セッションにおいてオブジェクトを再現するためには、 セッション開始前にクラス定義がロードされている必要があるためです。

全ての登録された変数は、リクエストが終了した後に、シリアル化されます。 未定義の登録変数は、未定義としてマーク付けされます。 これらの変数は、後でユーザーが定義しない限り、以降のアクセスにおいて セッションモジュールにより定義されません。

警告

いくつかのデータ型はシリアライズできませんので、 セッションにストアされません。これは resource 型の変数、 もしくは循環参照しているオブジェクト（例えば、 自分自身への参照を他のオブジェクトに渡しているオブジェクト）を含みます。

> **注意**: セッション処理機能は、PHP 4.0.0 でサポートされました。
>
> **注意**: セッションを処理している時、 session_register()関数を使用するか スーパーグローバル配列$_SESSIONへ新しいキーを 追加することにより変数が登録されるまで、セッションのレコードは作 成されないことに注意してください。これは、セッションが session_start()関数により開始されている場合で も真です。

セッションとセキュリティ

外部リンク: » Session fixation

セッションモジュールは、セッションに保存した情報を見ることができる のがそのセッションを作成したユーザーだけであることを保証することが できません。セッションの完全性を積極的に守るには、そのセッションに 紐づく値に応じた追加措置が必要です。

セッションに運ばれるデータの重要性を評価し、必要な保護策を講じてください。 この対策は、通常は何らかの犠牲を伴うもので、ユーザの利便性を損なうことになります。 例えば、単純なソーシャルエンジニアリングからユーザを守るためには session.use_only_cookies を有効にする必要があります。 この場合、ユーザ側でクッキーが常に有効となっていなければならず、 有効でない場合はセッションが動作しなくなります。

存在するセッションIDが第三者に洩れる手順は何種類かあります。 洩れたセッションIDにより、第三者が特定のIDに関連する全てのリソー スにアクセスできるようになります。まず、セッションIDがURLにより伝 送される場合です。外部サイトにリンクを張っている場合、外部サイト のreferrerログにセッションIDを含むURLが保存される可能性があります。 第二に、よりアクティブな攻撃者がネットワークのトラフィックをモニ ターしている可能性があります。セッションIDが暗号化されていない場 合、セッションIDはネットワーク上を平文テキストで伝送されます。 解決策はサーバ上にSSLを実装し、ユーザにSSLを必ず使用させることです。

要件

外部ライブラリを必要としません。

> **注意**: オプションで、Ralf S. Engelschallにより開発されたセッションの保 存用の共有メモリ(mm)を使用することも可能です。 » mmをダウンロードし、インストールす ることができます。このオプションは、Windowsプラットフォームでは 利用できません。mm用セッション保存モジュールは同一セッションのロッ クに問題があるため、同時アクセスを保証することはできません。 ファイルにセッションを保存するためには、 (Solaris/LinuxまたはBSD上の/dev/md)ファイルシステムに 共有メモリを使用するためにより適当でしょう。 セッションのデータはメモリに保存されます。そのため、Web サーバを 再起動するとデータが削除されます。

インストール手順

PHPのセッションサポートはデフォルトで有効となっています セッションサポートを有効にしてPHPを構築したくない場合には、 configureにオプション--disable-sessionを指定する必要があります。 セッション記憶領域として共有メモリ(mm)を使用するには、PHPのconfigure に --with-mm[=DIR]を指定します。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

> **注意**: デフォルトでは、特定のセッションのすべての情報は session.save_path INIオプションで指定されたディレクトリに生成される ファイルに保存されます。 結びついている情報に関わらずセッション毎にひとつのファイルが生成されます。 セッションが開始され（ファイルが生成される）しかし何の情報も そのファイルに書き込まれない場合もあります（サイズがゼロのファイルが残る）。 この振る舞いはファイルシステムを使っていることによる副作用であり、 カスタムセッションハンドラ（例：データベースを使ったもの）を 使う場合には何の情報も持たないセッションについて追跡しないように することは可能です。

実行時設定

php.ini の設定により動作が変化します。

セッションの設定オプション

名前	デフォルト	変更の可否	変更履歴
session.save_path	""	PHP_INI_ALL	
session.name	"PHPSESSID"	PHP_INI_ALL	
session.save_handler	"files"	PHP_INI_ALL	
session.auto_start	"0"	PHP_INI_ALL	
session.gc_probability	"1"	PHP_INI_ALL	
session.gc_divisor	"100"	PHP_INI_ALL	PHP 4.3.2 から利用可能
session.gc_maxlifetime	"1440"	PHP_INI_ALL	
session.serialize_handler	"php"	PHP_INI_ALL	
session.cookie_lifetime	"0"	PHP_INI_ALL	
session.cookie_path	"/"	PHP_INI_ALL	
session.cookie_domain	""	PHP_INI_ALL	
session.cookie_secure	""	PHP_INI_ALL	PHP 4.0.4 から利用可能
session.cookie_httponly	""	PHP_INI_ALL	PHP 5.2.0 から使用可能
session.use_cookies	"1"	PHP_INI_ALL	
session.use_only_cookies	"1"	PHP_INI_ALL	PHP 4.3.0 から利用可能
session.referer_check	""	PHP_INI_ALL	
session.entropy_file	""	PHP_INI_ALL	
session.entropy_length	"0"	PHP_INI_ALL	
session.cache_limiter	"nocache"	PHP_INI_ALL	
session.cache_expire	"180"	PHP_INI_ALL	
session.use_trans_sid	"0"	PHP_INI_ALL	PHP_INI_ALL は PHP <= 4.2.3、PHP_INI_PERDIR は PHP < 5 から。PHP 4.0.3 から利用可能。
session.bug_compat_42	"1"	PHP_INI_ALL	PHP 4.3.0 から利用可能。PHP 6.0.0 で削除。

名前	デフォルト	変更の可否	変更履歴
session.bug_compat_warn	"1"	PHP_INI_ALL	PHP 4.3.0 から利用可能。PHP 6.0.0 で削除。
session.hash_function	"0"	PHP_INI_ALL	PHP 5.0.0 から利用可能
session.hash_bits_per_character	"4"	PHP_INI_ALL	PHP 5.0.0 から利用可能
url_rewriter.tags	"a=href,area=href,frame=src,form=,fieldset="	PHP_INI_ALL	PHP 4.0.4 から利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

セッション管理システムは、php.iniファイルに記述可能な多くの設定オ プションをサポートします。以下に概要を示します。

session.save_handler string
session.save_handler は、 セッションに関連するデータの保存および取得時に使用するハンドラを定義します。 デフォルトは、files です。各拡張モジュールで、 独自の save_handler を使用できることに注意しましょう。 インストール環境単位で登録されているハンドラを取得するには phpinfo() を使用します。 session_set_save_handler() も参照してください。
session.save_path string
session.save_path は、保存ハンドラに渡される 引数を定義します。デフォルトのファイルハンドラを選択した場合、 ファイルが作成される場所のパスになります。 session_save_path() も参照ください。

オプションの引数としてN（数値）を指定できます。 これはセッションファイルを分散して保存する際に ディレクトリ階層レベルを決定します。 例えば、'5;/tmp'とすると /tmp/4/b/1/e/3/sess_4b1e384ad74619bd212e236e52a5a174If という位置にセッションファイルを生成します。 Nを使用するには、これらすべてのディレクトリが 事前に作成されている必要があります。 そのためのシェルスクリプトがext/sessionに mod_files.sh というファイル名であります。 また、0以上のNが指定されている場合には自動ガーベッジコレクション が機能しないことに注意してください。詳細は php.ini を見てください。 また、Nを指定する場合は、 session.save_pathを"quotes"で囲う必要があります。 なぜならセパレータ(;) は php.ini ではコメントとしても利用されているからです。

警告

この設定を/tmp（デフォルト)のようにどこか らでも読み込み可能なディレクトリのままにしている場合、サーバ上 の他のユーザがこのディレクトリのファイルのリストを取得すること により、セッションをハイジャックをすることが可能となります。

> **注意**: PHP 4.3.6 以前では、WindowsユーザがPHPのsession関数を使用するためには、 この変数を変更する必要があります。c:/temp のような有効なパスを指定するようにしてください。

session.name string
session.name はセッション名を指定し、 クッキー名として使用されます。 アルファベット文字のみで指定する必要があります。 デフォルトは、PHPSESSID です。 session_name()も参照してください。
session.auto_start boolean
session.auto_start はリクエスト開始時に セッションモジュールがセッションを自動的に開始するかどうかを 指定します。デフォルトは、0(無効)です。
session.serialize_handler string
session.serialize_handler は、シリアル化または シリアル化データを復元するために使用されるハンドラの名前を定義します。 現在、(php あるいは php_binary という名前の) PHP 内部フォーマットおよび (wddx という名前の) WDDX がサポートされています。WDDX は、PHP がWDDX サポート を有効にしてコンパイルされている場合のみ使用可能です。デフォルトは、 php です。
session.gc_probability integer
session.gc_probabilityと session.gc_divisorの組み合わせでgc （ガーベッジコレクション）ルーチンの始動を制御します。 デフォルトは、1です。 詳細はsession.gc_divisor をご覧ください
session.gc_divisor integer
session.gc_divisorと session.gc_probabilityの組み合わせで すべてのセッションの初期化過程でgc（ガーベッジコネクション）プロセス も始動する確率を制御します。確率は gc_probability/gc_divisor で計算されます。例えば、1/100は各リクエスト毎に1%の確率でGCプロセスが 始動します。 session.gc_divisorのデフォルトは100です。
session.gc_maxlifetime integer
session.gc_maxlifetime は、データが 'ごみ' とみなされ、消去されるまでの秒数を指定します。 ガベージコレクション（ごみの収集）は、セッションの開始時に行われます。

> **注意**: 異なる値を session.gc_maxlifetime に指定している 別々のスクリプトがセッションデータの保存場所を共有している場合、一番小さい設定値に達した時点でデータが消去されます。このような場合には、 お互いに session.save_path を使用します。

> **注意**: デフォルトのファイルに基づくセッションハンドラを使用している場 合、使用するファイルシステムは、アクセス時間(atime)を記録できる 必要があります。Windows FATはこれができないため、 FATファイルシステムまたはatimeの記録ができない他のファイルシス テムで問題を発生した場合は、セッションのガベージコレクト処理を 行う他の手段を用意する必要があります。 PHP4.2.3以降、atimeの代わりにmtime（更新時刻）が使用されます。 このため、atimeが利用できないファイルシステムでの問題は無くなりました。

session.referer_check string
session.referer_check には、HTTP Referer に おいて確認を行う文字列を指定します。Refererがクライアントにより 送信されており、かつ、指定した文字列が見付からない場合、埋め込 まれたセッションIDは無効となります。デフォルトは空の文字列です。
session.entropy_file string
session.entropy_file は、 セッションIDを作成する際の別のエントロピソースとして使用する 外部リソースへのパスを指定します。 例としては、多くの UNIX で利用可能な /dev/random または /dev/urandom があげられます。
session.entropy_length integer
session.entropy_length は、前記のファイルから 読みこむバイト数を指定します。デフォルトは、0 (無効)です。
session.use_cookies boolean
session.use_cookiesによりクライアント側にセッ ションIDを保存する際にクッキーを使用するかどうかを指定します。デ フォルトは1 (有効)です。
session.use_only_cookies boolean
session.use_only_cookies は、 このモジュールがクライアント側へのセッション ID の保存に Cookie のみ を使用することを指定します。 この設定を有効にすることにより、セッション ID を URL に埋め込む攻撃を防ぐことができます。この設定は、 PHP 4.3.0 で追加されました。 PHP 6.0 以降で、デフォルトは 1 (有効) となります。
session.cookie_lifetime integer
session.cookie_lifetime は、 ブラウザに送信するクッキーの有効期間を秒単位で指定します。 0 を指定すると "ブラウザを閉じるまで" という意味になります。 デフォルトは、0 です。 session_get_cookie_params() および session_set_cookie_params() も参照してください。

> **注意**: 有効期限のタイムスタンプは、サーバの時刻に基づいて決まります。 クライアントのブラウザの時刻がこれと同じであるとは限りません。

session.cookie_path string
session.cookie_pathによりsession_cookieで設 定するパスを指定します。デフォルトは/です。 session_get_cookie_params()および session_set_cookie_params()も参照してください。
session.cookie_domain string
session.cookie_domain により session_cookie で 指定するドメインを指定します。デフォルトでは指定されません。 この場合は、クッキーの仕様によって、クッキーを作成したサーバの ホスト名が指定されます。 session_get_cookie_params() および session_set_cookie_params() も参照ください。
session.cookie_secure boolean
session.cookie_secureは、 セキュアな接続を通じてのみCookieを送信できるかどうかを指定します。 デフォルトは、offです。 この設定は、PHP 4.0.4で追加されました。 session_get_cookie_params()および session_set_cookie_params()も参照してください。

session.cookie_httponly boolean
クッキーに対して、HTTP を通してのみアクセスできるようにします。 つまり、JavaScript のようなスクリプト言語からはアクセスできなくなるということです。 この設定を使用すると、XSS 攻撃によって ID を盗まれる危険性を減らせます (が、すべてのブラウザがこの設定をサポートしているというわけではありません)。
session.cache_limiter string
session.cache_limiterにより セッションページにおけるキャッシュ制御の方法 (none/nocache/private/private_no_expire/public) を指定します。デフォルトは、nocacheです。 session_cache_limiter()も参照してください。
session.cache_expire integer
session.cache_expireによりキャッシュされた セッションページの有効期間を分単位で指定します。 このオプションは、nocacheリミッタに関しては効果がありません。 デフォルトは、180です。 session_cache_expire()も参照してください。
session.use_trans_sid boolean
session.use_trans_sidは、透過的なセッション IDの付加をするかどうかを指定します。 デフォルトは、0(無効)です。

> **注意**: PHP 4.1.2より前のバージョンでは、このオプションは --enable-trans-sidにより コンパイル時に有効とされていました。 PHP 4.2.0以降、trans-sid機能は常にコンパイルされます。 URLに基づくセッション管理は、Cookieに基づくセッション管理と比べてセキュリティリスクが大きくなります。例えば、ユーザは、emailに より友人にアクティブなセッションIDを含むURLを送信する可能性があ り、また、ユーザは自分のブックマークにセッションIDを含むURLを保 存し、常に同じセッションIDで使用するサイトにアクセスする可能性 があります。

session.bug_compat_42 boolean
PHP バージョンが 4.2.3 とそれ以前には、たとえ register_globals が無効の場合でもグローバルスコープでセッション変数の初期化を 許してしまうドキュメント化されていない特徴/バグがあります。 この機能を使用している場合で session.bug_compat_warn も有効にしている場合、 PHP 4.3.0 とそれ以降のバージョンでは警告が発されます。 この特徴/バグは、 このディレクティブを無効にすることで無効にすることが可能です。
session.bug_compat_warn boolean
PHP バージョンが 4.2.3 とそれ以前には、たとえ register_globals が無効の場合でもグローバルスコープでセッション変数の初期化を 許してしまうドキュメント化されていない特徴/バグがあります。 この機能を session.bug_compat_42 と session.bug_compat_warn を有効にして使用している場合、 PHP 4.3.0 とそれ以降のバージョンでは警告が発されます。
session.hash_function mixed
session.hash_function によりセッション ID を生成するために使用されるハッシュアルゴリズムを指定することが可能です。 '0' は MD5 (128 ビット) で、'1' は SHA-1 (160 ビット) を意味します。

PHP 6.0.0 以降では、hash 拡張モジュール の任意のアルゴリズムが (この拡張モジュールが使用可能な場合に) 指定できるようになります。たとえば sha512 や whirlpool などです。 サポートされているアルゴリズムの一覧は、 hash_algos() 関数で取得できます。

> **注意**: このディレクティブは PHP 5 で導入されました。

session.hash_bits_per_character integer
session.hash_bits_per_character により バイナリのハッシュデータを何らかの可読なデータに変換する際、 それぞれの文字に何ビットストアさせるかを定義することが可能です。 指定可能な値は、'4' (0-9, a-f)、'5' (0-9, a-v) そして '6' (0-9, a-z, A-Z, "-", ",") です。

> **注意**: このディレクティブは PHP 5 で導入されました。

url_rewriter.tags string
url_rewriter.tagsは、透過的なセッションIDの 付加機能が有効となった場合に、セッションIDを含めるために書き換 えられるHTMLタグを指定します。デフォルトは、 a=href,area=href,frame=src,input=src,form=fakeentry,fieldset= です。

> **注意**: HTML/XHTML strict に適合させたい場合には form エントリは削除し、 formフィールドの前後に<fieldset> タグを使ってください。

track_varsおよび register_globals 設定はセッション変数の保存および回復方法に影響を与えます。

> **注意**: PHP 4.0.3以降、track_vars は常 にonとなっています。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

SID (string)
"name=ID"形式でセッション名とセッションIDを格納している定数。 セッションIDがセッションクッキーに適切にセットされている場合には空文字列が入る。

例

> **注意**: PHP 4.1.0以降、$_SESSIONは、 $_POST, $_GET, $_REQUEST等のようにグローバル変数として利用可 能です。 $HTTP_SESSION_VARSと異なり、 $_SESSIONは常にグローバルです。そこで、 global は$_SESSIONの場合は不要です。 このドキュメントでは、$_SESSION をあらゆる場 所で使用していることに注意してください。もし前者を使用したい場合に は、$_SESSIONを $HTTP_SESSION_VARSで置き換えることができます。 また、$_SESSIONを使用する前に session_start() を用いてセッションを開始して おく必要があることに注意してください。
> 連想配列$_SESSIONのキーは、PHPの通常の変数名 と同じ制限があります。すなわち、数字で始まることはできず、文字ま たはアンダースコアで始まる必要があります。 詳細については、本マニュアルの 変数の節を参照して下さ い。

register_globals が無効の場合、グローバル連想配列$_SESSIONのメ ンバーのみがセッション変数として登録されます。 回復されたセッション変数は、配列$_SESSIONでの み利用可能です。

セキュリティとコードの可読性のために$_SESSION (またはPHP 4.0.6以前は$HTTP_SESSION_VARS)の使用 が推奨されます。$_SESSIONの場合、session_register(), session_unregister(), session_is_registered()は不要です。ユーザは、 通常の変数と同様にセッション変数にアクセス可能 です。

Example#1 $_SESSIONで変数を登録

```
<?php
session_start();
// PHP 4.0.6以前の場合は$HTTP_SESSION_VARSを使用してください
if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
} else {
    $_SESSION['count']++;
}
?>
```

Example#2 register_globals が無効な場合に、$_SESSIONに登録されている変 数の登録を解除する

```
<?php
session_start();
// PHP 4.0.6とそれ以前では$HTTP_SESSION_VARSを使用してください
unset($_SESSION['count']);
?>
```

警告

unset($_SESSION)によって 全ての$_SESSIONを初期化してはいけません。 $_SESSIONスーパーグローバル変数を用いた セッション変数の登録ができなくなってしまうからです。

警告

セッション変数において参照を使用することはできません。 他の変数への参照の再現する方法がないからです。

register_globals が有効な場合、全てのグローバル変数はセッション変数として登録するこ とが可能で、セッション変数は対応するグローバル変数として回復されま す。PHPは、どのグローバル変数がセッション変数として登録されるのか を知る必要があるため、ユーザは、変数を session_register()関数で登録する必要がありま す。しかし、$_SESSIONの場合は、エントリを設定 するだけでこれを行う必要はありません。

警告

$_SESSIONを使用し、 register_globals を無効とする場合、自分のスクリプトをPHP 4.2より以前のバージョン で動作させたい場合は、 session_register()、 session_is_registered()、 session_unregister()を使用しないでください。 PHP 4.3.0 以降ではこれらの関数を使用することができます。 セキュリティ面と性能面の双方よりregister_globals を無効とすることが推奨されています。

register_globals が有効な場合、グローバル変数と$_SESSIONの エントリは、前のセッションインスタンスで登録されたセッション変数 の同じ値を参照することになります。 しかし、変数が $_SESSION で登録された場合、 グローバル変数が使用可能となるのは次のリクエスト以降です。

PHP 4.2.3とそれ以前のバージョンのみに関係する問題があります。 session_register()により新しいセッショ ン変数を登録する場合、グローバルスコープのエントリと $_SESSIONのエントリは、次の session_start()まで同じ値へのリファレンスとは なりません。 すなわち、グローバル変数への修正は、$_SESSION のエントリには反映されません。PHP 4.3.0 では修正されています。

セッションIDの受渡し

セッションIDの通知を行うためには次の二つの方法があります。

- Cookie
- URLパラメータ

sessionモジュールは、両方の方法をサポートします。 Cookieは最適ですが、(クライアントがCookieを受け入れない可能性が あるため)信頼性がなく、これに依存することができません。2番目の方 法は、セッションIDを直接URLに埋め込みます。

PHPには、透過的にリンクを変換する機能を有しています。 PHP 4.2.0 以降を使用していない場合、PHP構築時にこの機能を有効にして おく必要があります。UNIX環境では、 --enable-trans-sidをconfigureに指定して ください。この構築オプションと実行時オプション session.use_trans_sidが有効な場合、 相対URIは自動的にセッションIDを含むように変換されます。

> **注意**: arg_separator.output php.ini ディレクティブにより、引数セパレータをカスタマイズする ことができます。XHTMLに完全準拠するためには、ここに & を指定してください。

もしくは、セッションが開始している場合に定義されている定数 SID を使用することもできます。クライアントが適当な セッションクッキーを送信しなかった場合、この定数は session_name=session_id の形式となります。 他方、送信された場合には、この定数は空の文字列に展開されます。このため、この定数を無条件に URL に埋め込むことができます。

次の例は、変数の登録法および SID を用いて他のページに正しくリンク する方法のデモです。

Example#3 単一のユーザーに関するヒット数を数える

```
<?php

session_start();

if (empty($_SESSION['count'])) {
   $_SESSION['count'] = 1;
} else {
   $_SESSION['count']++;
}
?>

<p>
こんにちは、あなたがこのページに来たのは <?php echo $_SESSION['count']; ?> 回目ですね。
</p>

<p>
続けるには、<a href="nextpage.php?<?php echo htmlspecialchars(SID); ?>">ここをクリック</A>
してください。
</p>
```

XSSに関係する攻撃を防止するためにSIDを出力する際に、 htmlspecialchars()を使用します。

PHPをコンパイルする際に --enable-trans-sid を使用した場合、 上の例のように SID を出力する必要はありません。

> **注意**: 相対URLでないURLは外部サイトを指していると仮定され、SIDが追加 されません。これは、SIDを外部のサーバに開示することは セキュリティ 上のリスクとなる可能性があるためです。

カスタムセッションハンドラ

セッション情報をデータベースに保存する機能か他の保存法を実装する には、一連のユーザレベルの保存関数を作成し、 session_set_save_handler()を使用する必要があり ます。

---

session_cache_expire

(PHP 4 >= 4.2.0, PHP 5)

session_cache_expire — 現在のキャッシュの有効期限を返す

説明

int **session_cache_expire** ([int $new_cache_expire])

session_cache_expire() は現在の session.cache_expire の設定を返します。

リクエストがあった時点でキャッシュの有効期限は session.cache_limiter で設定されたデフォルト値にリセットされます。 そのため、すべてのリクエストにおいて (そしてsession_start() をコールする前に) **session_cache_expire()** をコールする必要があります。

パラメータ

new_cache_expire

new_cache_expire が指定された場合、 現在のキャッシュの有効期限は、 new_cache_expire で置換されます。

> **注意**: session.cache_limiterが nocache以外の値にセットされている場合にのみ new_cache_expire が有効となります。

返り値

session.cache_expire の現在の設定を返します。 返り値は分単位で、デフォルトは 180 です。

例

Example#1 session_cache_expire() の例

```
<?php

/* set the cache limiter to 'private' */

session_cache_limiter('private');
$cache_limiter = session_cache_limiter();

/* set the cache expire to 30 minutes */
session_cache_expire(30);
$cache_expire = session_cache_expire();

/* start the session */

session_start();

echo "The cache limiter is now set to $cache_limiter<br />";
echo "The cached session pages expire after $cache_expire minutes";
?>
```

参考

- session.cache_expire
- session.cache_limiter
- session_cache_limiter()

---

session_cache_limiter

(PHP 4 >= 4.0.3, PHP 5)

session_cache_limiter — 現在のキャッシュリミッタを取得または設定する

説明

string **session_cache_limiter** ([string $cache_limiter])

session_cache_limiter() は、 現在のキャッシュリミッタの名前を返します。

キャッシュリミッタは、クライアントに送信されるキャッシュ制御用の HTTPヘッダを制御します。これらのヘッダは、ページの内容をクライアントやプロキシがキャッシュ する規則を定義します。例えば、キャッシュリミッタを nocacheに設定した場合、クライアント/プロキシのキャッ シュは無効になります。しかし、publicの場合は、 キャッシュを許可します。privateと設定すること も可能で、この場合、プロキシがキャッシュすることは許可しませんが クライアントがキャッシュすることは許可されます。

privateモードにおいて、Expireヘッダがクライア ントに送信されます。これは、Mozillaのようないくつかのブラウザを混 乱させます。これを避けるには、 private_no_expireモードを使用してください。 このモードでは、Expireヘッダはクライアントに送信されません。

キャッシュリミッタは、リクエスト開始時に session.cache_limiterに保存されたデフォルト値 にリセットされます。つまり、各リクエスト毎に (アウトプットバッファ が無効な場合は、session_start()がコールされる 前に) **session_cache_limiter()**をコールする必要 があります。

パラメータ

cache_limiter

cache_limiter が指定された場合、現在のキャッシュリミッタは新しい値に変更されます。

返り値

現在のキャッシュリミッタの名前を返します。

変更履歴

バージョン	説明
4.2.0	private_no_expire が追加されました。

例

Example#1 session_cache_limiter() の例

```
<?php

/* キャッシュリミッタを'private'に設定する */

session_cache_limiter('private');
$cache_limiter = session_cache_limiter();

echo "The cache limiter is now set to $cache_limiter<br />";
?>
```

参考

- session.cache_limiter

---

session_commit

(PHP 4 >= 4.4.0, PHP 5)

session_commit — session_write_close() のエイリアス

説明

この関数は次の関数のエイリアスです。 session_write_close().

---

session_decode

(PHP 4, PHP 5)

session_decode — 文字列からセッションデータをデコードする

説明

bool **session_decode** (string $data)

session_decode() は、 data のセッションデータをデコードし、 セッションに保存する変数を設定します。

パラメータ

data

エンコードされたデータ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- session_encode()

---

session_destroy

(PHP 4, PHP 5)

session_destroy — セッションに登録されたデータを全て破棄する

説明

bool **session_destroy** (void)

session_destroy()は、現在のセッションに 関連づけられた全てのデータを破棄します。この関数は、 セッションに関するグローバル変数を破棄しません。 また、セッションクッキーを破棄しません。

ユーザーがログアウトするときのように、セッションを切断するには、 セッション ID の割り当ても解除する必要があります。セッション ID の受け渡しに クッキーが使用されている場合（デフォルト）には、セッションクッキーも 削除されなければなりません。 そのためには setcookie() が利用できます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 $_SESSIONでセッションを破棄する

```
<?php
// セッションの初期化
//  session_name("something")を使用している場合は特にこれを忘れないように!
session_start();

// セッション変数を全て解除する
$_SESSION = array();

// セッションを切断するにはセッションクッキーも削除する。
// Note: セッション情報だけでなくセッションを破壊する。
if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time()-42000, '/');
}

// 最終的に、セッションを破壊する
session_destroy();
?>
```

注意

> ***注意***: $_SESSION を使っていない古いコードでのみ session_unset() を使用するようにしましょう。

参考

- unset()
- setcookie()

---

session_encode

(PHP 4, PHP 5)

session_encode — 現在のセッションデータを文字列としてエンコードする

説明

string **session_encode** (void)

session_encode() は、 現在のセッションの内容をエンコードした文字列を返します.

返り値

現在のセッションの内容をエンコードしたものを返します。

参考

- session_decode()

---

session_get_cookie_params

(PHP 4, PHP 5)

session_get_cookie_params — セッションクッキーのパラメータを得る

説明

array **session_get_cookie_params** (void)

セッションクッキーのパラメータを取得します。

返り値

現在のセッションクッキーの情報を配列として返します。 この配列には次のような項目が含まれています。

- "lifetime" - クッキーの生存期間(lifetime)
- "path" - 情報が保存されている場所のパス
- "domain" - クッキーのドメイン
- "secure" - クッキーはセキュアな接続でのみ送信されます。
- "httponly" - クッキーは HTTP を通してのみアクセス可能となります。

変更履歴

バージョン	説明
5.2.0	返される配列に、"httponly" エントリが追加されました。
4.0.4	返される配列に、"secure" エントリが追加されました。

参考

- session.cookie_lifetime
- session.cookie_path
- session.cookie_domain
- session.cookie_secure
- session.cookie_httponly
- session.cookie_lifetime
- session_set_cookie_params()

---

session_id

(PHP 4, PHP 5)

session_id — カレントのセッション ID を取得または設定する

説明

string **session_id** ([string $id])

session_id() はカレントのセッション ID を取得 または設定するために使用されます。

カレントの名前とセッション ID を、URL に追加可能な文字列として取得する ために定数 SID も使用することが可能です。 セッションハンドリング も参照してください。

パラメータ

id

id が指定された場合、カレントの セッション ID を置換します。その際、 session_start() より前にコールされている必要があります。 セッションハンドラによっては、セッション ID として使用できる文字に 制限がある場合があります。例えば、ファイルによるセッションハンドラは セッション ID として使える文字は a-z, A-Z, 0-9 に限られます!

> **注意**: セッション保持にクッキーを使用している場合、 **session_id()** において id 引数を指定すると、 カレントのセッション ID が セットされるものと まったく同一であるかどうかに関わらず、 session_start() が呼び出される際に常に新しいクッキーが送信されます。

返り値

session_id() はカレントのセッションのセッション ID、 もしくはカレントセッションが存在しない（カレントのセッション ID が存在しない）場合は空文字列 ("") を返します。

参考

- session_regenerate_id()
- session_start()
- session_set_save_handler()
- session.save_handler

---

session_is_registered

(PHP 4, PHP 5)

session_is_registered — 変数がセッションに登録されているかどうかを調べる

説明

bool **session_is_registered** (string $name)

グローバル変数がセッションに登録されているかどうかを調べます。

パラメータ

name

変数名。

返り値

session_is_registered() は、 name という名前のグローバル変数が現在のセッションに登録されている場合に **TRUE**、それ以外の場合に **FALSE** を返します。

注意

> **注意**: $_SESSION （または PHP 4.0.6 以前の場合は $HTTP_SESSION_VARS） が使用されている場合、ある変数が $_SESSION に登録されているかを確認するために isset() を使用してください。

警告

$_SESSION （もしくは $HTTP_SESSION_VARS） を使用している場合、 session_register(), **session_is_registered()**, session_unregister() を使用しないでください。

---

session_module_name

(PHP 4, PHP 5)

session_module_name — 現在のセッションモジュールを取得または設定する

説明

string **session_module_name** ([string $module])

session_module_name()は、 現在のセッションモジュールの名前を返します。

パラメータ

module

module が指定された場合、 そのモジュールを代わりに使用します。

返り値

現在のセッションモジュールの名前を返します。

---

session_name

(PHP 4, PHP 5)

session_name — 現在のセッション名を取得または設定する

説明

string **session_name** ([string $name])

session_name() は、現在のセッション名を返します。

セッション名は、リクエストが開始された際に セッション名に保存された session.name の デフォルト値にリセットされます。よって、各リクエスト毎に(そして session_start() または session_register() をコールする前に) **session_name()** をコールする必要があります。

パラメータ

name

セッション名は、クッキーおよび URL のセッション ID を参照します。 セッション名は英数字のみで構成されている必要があり、また、 短く、その内容が分かるようなものである必要があります (これは、クッキー警告を有効にしているユーザ用です)。

警告

セッション名は数字だけで構成することはできません。 少なくとも文字がひとつ以上現れる必要があります。そうでない場合、 新規セッション ID が毎回生成されます。

返り値

現在のセッションの名前を返します。

例

Example#1 session_name() の例

```
<?php

/* セッション名をWebsiteIDに設定する */

$previous_name = session_name("WebsiteID");

echo "前回のセッション名は、$previous_name です。<br />";
?>
```

参考

- 設定ディレクティブ session.name

---

session_regenerate_id

(PHP 4 >= 4.3.2, PHP 5)

session_regenerate_id — 現在のセッションIDを新しく生成したものと置き換える

説明

bool **session_regenerate_id** ([bool $delete_old_session])

session_regenerate_id() は現在のセッションIDを 新しいものと置き換えます。その際、現在のセッション情報は維持されます。

パラメータ

delete_old_session

*関連付けられた古いセッションを削除するかどうか。 デフォルトは **FALSE** です。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

変更履歴

バージョン	説明
4.3.3	これ以降、セッション保持にクッキーを使用している場合、 session_regenerate_id() を使用することにより 新しいセッション ID を持つセッションクッキーも発行します。
5.1.0	*delete_old_session* パラメータが追加されました。

例

Example#1 A session_regenerate_id() の使用例

```
<?php
session_start();

$old_sessionid = session_id();

session_regenerate_id();

$new_sessionid = session_id();

echo "古いセッション: $old_sessionid<br />";
echo "新しいセッション: $new_sessionid<br />";

print_r($_SESSION);
?>
```

参考

- session_id()
- session_start()
- session_name()

---

session_register

(PHP 4, PHP 5)

session_register — 現在のセッションに1つ以上の変数を登録する

説明

*bool **session_register** (mixed $name [, mixed $...])*

***session_register()** の引数の数は可変であり、 各引数は変数名を保持する文字列または変数名からなる配列 とすることが可能です。 各変数名が処理される毎に、**session_register()** は、その変数名のグローバル変数を現在のセッションに登録します。*

配列 $_SESSION または $HTTP_SESSION_VARS (PHP < 4.1.0) の適当なメンバに設定をするだけでもセッション変数を作成することが可能です。

```
<?php
// session_register() の使用は推奨されません。
$barney = "A big purple dinosaur.";
session_register("barney");

// PHP 4.1.0以降では$_SESSIONの使用が推奨されます。
$_SESSION["zim"] = "An invader from another planet.";

// 古い手法としては $HTTP_SESSION_VARS があります。
$HTTP_SESSION_VARS["spongebob"] = "He's got square pants.";
?>
```

この関数をコールする前に session_start() をコールしていない場合、暗黙のうちに引数を付けずに session_start() がコールされます。 $_SESSION を使う場合はこの動作とは違い、 使用前に必ず session_start() をコールする必要があります。

パラメータ

name

変数名を含む文字列、 あるいは変数名や配列を含む配列。

...

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

注意

警告

register_globals を考慮することなくスクリプトを書きたい場合には、 $_SESSION 配列を代わりに使用する必要があります。 $_SESSION のエントリは自動的に登録されます。 スクリプトで **session_register()** を使用している場合には、それは register_globals を不可としている環境下では動作しません。

> **注意**: **register_globals: 重要な注意**PHP 4.2.0 以降、PHP ディレクティブ register_globals のデフォルト値は off となっています。また、このディレクティブは PHP 6.0.0 で完全に削除される予定です。 PHP コミュニティは、ユーザがこのディレクティブの設定に依存せず、 superglobals のような他の手段を使用することを推奨します。

警告

この関数は、global変数を登録します。 セッション変数を関数の内部で登録したい場合、 **global** キーワードあるいは $GLOBALS[] 配列を用いてその変数をグローバルとするか、下記のように session 配列を使用してください。

警告

$_SESSION (あるいは $HTTP_SESSION_VARS) を使用する場合は **session_register()**、 session_is_registered() および session_unregister() を使用しないでください。

> **注意**: セッションにリソース変数を登録することは現在できません。例えば、 データベースへの接続を生成し、接続IDをセッション変数として登録し、 セッションが回復された時点で、接続が有効であることを期待することはできません。 リソースを返すPHP関数は、その関数定義に resource型の返り値を有することが示されている。 リソースを返す関数のリストは、付録 リソース型の一覧 で見ることができます。
> $_SESSION (あるいは PHP 4.0.6 以前なら $HTTP_SESSION_VARS) を使用する場合は、値を $_SESSION に代入します。 たとえば $_SESSION['var'] = 'ABC'; などのようにします。

参考

- session_is_registered()
- session_unregister()
- $_SESSION

---

session_save_path

(PHP 4, PHP 5)

session_save_path — 現在のセッションデータ保存パスを取得または設定する

説明

string **session_save_path** ([string $path])

session_save_path() は、 現在のセッションデータ保存パスを返します。

パラメータ

path

セッションデータのパス。これを指定すると、 データを保存するパスが変更されます。ただしそのためには session_start() がコールされる前に **session_save_path()** がコールされている必要があります。

> **注意**: いくつかのオペレーションシステムでは、多くの小さなファイルを効率的に 処理するファイルシステム上にパスを指定することが望ましいです。 例えば、Linux では ext2fs よりも reiserfs の方が性能面でより 優れています。

返り値

現在のデータ保存先ディレクトリのパスを返します。

参考

- 設定ディレクティブ session.save_path

---

session_set_cookie_params

(PHP 4, PHP 5)

session_set_cookie_params — セッションクッキーパラメータを設定する

説明

void **session_set_cookie_params** (int $lifetime [, string $path [, string $domain [, bool $secure [, bool $httponly]]]])

ファイル php.ini で定義されたクッキーパラメータを設定します。 この関数の効果が持続するのは、スクリプトの実行が終了するまでです。 したがって、リクエスト毎や session_start() がコールされる前に **session_set_cookie_params()** をコールする必要があります。

パラメータ

lifetime

path

domain

secure

httponly

返り値

値を返しません。

変更履歴

バージョン	説明
5.2.0	*httponly* パラメータが追加されました。
4.0.4	*secure* パラメータが追加されました。

参考

- *session.cookie_lifetime*
- *session.cookie_domain*
- *session.cookie_secure*
- *session.cookie_httponly*
- *session_get_cookie_params()*

---

session_set_save_handler

(PHP 4, PHP 5)

session_set_save_handler — ユーザ定義のセッション保存関数を設定する

説明

bool ***session_set_save_handler*** *(callback $open , callback $close , callback $read , callback $write , callback $destroy , callback $gc)*

session_set_save_handler() は、セッションに 関連するデータを保存および取得するために使用されるユーザ定義の セッション保存関数を設定します。この関数は、セッションデータを ローカルデータベースに保存する場合のように PHP セッションにより 提供されるもの以外の保存方法を使用したい場合に有用です。

パラメータ

open

close

read

save ハンドラが期待通りに動作するように *read* 関数は常に文字列を返す 必要があります。データがない場合には、空の文字列を返してください。他のハンドラからの返り値は、論理式、すなわち成功した場合に ***TRUE***、 失敗した場合に ***FALSE*** を返します。

write

> ***注意***: *"write"* ハンドラは、出力ストリームが閉じてから実行されます。 したがって、*"write"* ハンドラ内でデバッグ出力を行っても、 それはブラウザに表示されません。 デバッグ出力が必要なら、それをファイルに書き出すようにしましょう。

destroy

gc

返り値

成功した場合に ***TRUE*** を、失敗した場合に ***FALSE*** を返します。

例

Example#1 session_set_save_handler() の例

以下の例では、デフォルトの保存 *files* に似た ファイルベースのセッション保存を行います。この例は、PHP がサポート する任意のデータベースエンジンを用いてデータベースへの保存を行う ように容易に拡張可能です。

```
<?php
function open($save_path, $session_name)
{
   global $sess_save_path;

  $sess_save_path = $save_path;
   return(true);
}

function close()
{
   return(true);
}

function read($id)
{
   global $sess_save_path;
```

```
  $sess_file = "$sess_save_path/sess_$id";
   return (string) @file_get_contents($sess_file);
}

function write($id, $sess_data)
{
  global $sess_save_path;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    $return = fwrite($fp, $sess_data);
    fclose($fp);
    return $return;
  } else {
    return(false);
  }

}

function destroy($id)
{
  global $sess_save_path;

  $sess_file = "$sess_save_path/sess_$id";
   return(@unlink($sess_file));
}

function gc($maxlifetime)
{
  global $sess_save_path;

   foreach (glob("$sess_save_path/sess_*") as $filename) {
    if (filemtime($filename) + $maxlifetime < time()) {
      @unlink($filename);
    }
  }
  return true;
}

session_set_save_handler("open", "close", "read", "write", "destroy", "gc");

session_start();

// セッションを通常通り使用します。

?>
```

注意

警告

PHP 5.0.5 以降、write ハンドラおよび close ハンドラはオブジェクトが破棄されたあとにコールされます。 そのため、セッション内でデストラクタを使用可能ですが、 ハンドラ内ではオブジェクトを使用できません。

この「ニワトリが先かタマゴが先か」の問題を解決するために、 デストラクタから session_write_close() を コールすることが可能です。

警告

SAPI の種類によっては、スクリプトの終了時にセッションを閉じると 現在の作業ディレクトリが変わってしまうことがあります。これを防ぐには、事前に session_write_close() でセッションを閉じます。

参考

- 設定ディレクティブ session.save_handler

---

session_start

(PHP 4, PHP 5)

session_start — セッションデータを初期化する

説明

bool **session_start** (void)

session_start() は、セッションを作成します。 もしくは、リクエスト上で GET, POST またはクッキーにより渡されたセッション ID に基づき現在のセッションを復帰します。

名前付きのセッションを使用したい場合、 **session_start()** の前に session_name() をコールする必要があります。

session_start() は、 trans-sid が有効の場合に URL 書換え用の内部出力ハンドラを登録します。ユーザが ob_start() と共に ob_gzhandler または類似のものを使用している場合、 出力ハンドラの順番は正しく出力を行うために重要です。例えば、 セッション開始時にユーザは ob_gzhandler を登録する必要があります。

返り値

この関数は常に **TRUE** を返します。

変更履歴

バージョ	説明

ン

4.3.3 セッションが既に開始されている状態で session_start() をコールすると E_NOTICE レベルのエラーを発生させます。 またその場合二度目のセッションスタートは単に無視されます。

例

Example#1 セッションの例 page1.php

```
<?php
// page1.php

session_start();

echo 'Welcome to page #1';

$_SESSION['favcolor'] = 'green';
$_SESSION['animal']   = 'cat';
$_SESSION['time']     = time();

// セッションクッキーが有効なら動作します
echo '<br /><a href="page2.php">page 2</a>';

// あるいは必要に応じてセッション ID を渡します
echo '<br /><a href="page2.php?' . SID . '">page 2</a>';
?>
```

page1.php を表示した後なら、 page2.php はセッション上の情報を含んでいるはずです。 セッションについてのリファレンス を読むと、 セッションIDの伝達 に関する情報が得られます。 例えば、SID とは何かといったことです。

Example#2 セッションの例: page2.php

```
<?php
// page2.php

session_start();

echo 'Welcome to page #2<br />';

echo $_SESSION['favcolor']; // green
echo $_SESSION['animal'];   // cat
echo date('Y m d H:i:s', $_SESSION['time']);

// page1.php と同様に、ここで SID を使うこともできます
echo '<br /><a href="page1.php">page 1</a>';
?>
```

注意

> **注意**: クッキーに基づくセッションを使用している場合、ブラウザに何か出力を行う前に session_start() をコールする必要があります。
>
> **注意**: ob_gzhandler() よりも zlib.output_compression の使用が推奨されています。

参考

- $_SESSION
- 設定ディレクティブ session.auto_start
- session_id()

---

session_unregister

(PHP 4, PHP 5)

session_unregister — 現在のセッションから変数の登録を削除する

説明

bool **session_unregister** (string $name)

session_unregister() は現在のセッションから name という名前のグローバル変数の登録を削除します。

パラメータ

name

変数名。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

注意

> **注意**: $_SESSION (または PHP 4.0.6 以前の場合は $HTTP_SESSION_VARS) が使用されている場合、セッション変数の登録を削除する際に unset()を 使用してください。 $_SESSION 自身を unset() しないでください。 さもないと、$_SESSION スーパーグローバルの特殊な機能が無効化されてしまいます。

警告

この関数は、name に対応するグローバル変数の登録を削除しません。 セッションとしてその変数が保存されるのを妨げるだけです。 対応するグローバル変数を削除するには、 unset() をコールする必要があります。

警告

$_SESSION (あるいは $HTTP_SESSION_VARS) を使用する場合は、 session_register()、 session_is_registered() および **session_unregister()** を使用しないでください。

---

session_unset

(PHP 4, PHP 5)

session_unset — 全てのセッション変数を開放する

説明

void **session_unset** (void)

関数 **session_unset()** は現在登録されている全てのセッション変数を開放します。

返り値

値を返しません。

注意

> **注意**: $_SESSION (または PHP 4.0.6 以前の場合は $HTTP_SESSION_VARS) が使用されている場合、セッション変数の登録を削除するために unset() すなわち、unset ($_SESSION['varname']); を使用してください。

警告

$_SESSION スーパーグローバルを使用した セッション変数の登録が不可能になってしまうため、 unset($_SESSION) を使って $_SESSION を完全に unset しないでください。

---

session_write_close

(PHP 4 >= 4.0.4, PHP 5)

session_write_close — セッションデータを書き込んでセッションを終了する

説明

void **session_write_close** (void)

現在のセッションを終了し、セッションデータを書き込みます。

セッションデータは、**session_write_close()** をコールしなくても、スクリプト終了時に保存されます。しかし、 セッションデータは、同時書き込みを防ぐためにロックされるため、 ある時点であるセッションの処理ができるスクリプトは、1つだけです。 セッションでフレームセットを使用する場合、 このロックのためにフレームがひとつずつロードされるような経験をするでしょう。 セッションへの全ての変更が行われるとすぐにセッションを終了することにより、 全てのフレームのロードに要する時間を減らすことができます。

返り値

値を返しません。

目次

- session_unset — 全てのセッション変数を開放する
- session_write_close — セッションデータを書き込んでセッションを終了する

共有メモリ関数(shmop)

導入

shmop は、共有メモリセグメントを PHP から簡単に読み書きまたは作成、 削除することを可能にする一連の関数です。

> **注意**: Windows 2000 より前のバージョンの Windows では共有メモリをサポートしていません。 Windows では、PHP が Apache や IIS などの web サーバモジュールとして稼動している場合にのみ shmop が動作します (CLI および CGI では動作しません)。
>
> **注意**: PHP 4.0.3 では、以下の関数に接頭辞 shmop ではなく shm が付いていました。

要件

外部ライブラリを必要としません。

インストール手順

shmop を使用するには、--enable-shmopパラメータを configure に 指定して PHP をコンパイルする必要があります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

定義済み定数

定数は定義されていません。

例

Example#1 共有メモリ操作の概要

```
<?php

// システムID 0xff3を有する 100 バイトの共有メモリブロックを作成する
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if (!$shm_id) {
    echo "共有メモリセグメントを作成できませんでした。\n";
}

// 共有メモリのブロック長を得る
$shm_size = shmop_size($shm_id);
echo "SHM ブロックサイズ: ".$shm_size. " が作成されました。\n";

// 共有メモリにテスト用の文字列を書き込んでみる
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if ($shm_bytes_written != strlen("my shared memory block")) {
    echo "データ全体を書き込めませんでした。\n";
}

// その文字列を再び読み込んでみる
$my_string = shmop_read($shm_id, 0, $shm_size);
if (!$my_string) {
    echo "共有メモリブロックから読み込めません。\n";
}
echo "共有メモリ内のデータは次のようになります: ".$my_string."¥n";

// ブロックを削除し、共有メモリセグメントを閉じる
if (!shmop_delete($shm_id)) {
    echo "共有メモリブロックに削除用のマークを付けることができません。";
}
shmop_close($shm_id);

?>
```

---

shmop_close

(PHP 4 >= 4.0.4, PHP 5)

shmop_close — 共有メモリブロックを閉じる

説明

void **shmop_close** (int $shmid)

shmop_close() は共有メモリブロックを閉じるために 使用されます。

パラメータ

shmid

shmop_open() が作成した共有メモリブロックの識別子。

返り値

値を返しません。

例

Example#1 共有メモリブロックを閉じる

```
<?php
shmop_close($shm_id);
?>
```

この例は、$shm_id を ID とする共有メモリブロックを閉じます。

参考

- shmop_open()

---

shmop_delete

(PHP 4 >= 4.0.4, PHP 5)

shmop_delete — 共有メモリブロックを削除する

説明

bool **shmop_delete** (int $shmid)

shmop_delete() は共有メモリブロックを削除するために 使用されます。

パラメータ

shmid

shmop_open() が作成した共有メモリブロックの識別子。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 共有メモリブロックの削除

```
<?php
shmop_delete($shm_id);
?>
```

この例は、$shm_id を ID とする共有メモリブロックを 削除します。

---

shmop_open

(PHP 4 >= 4.0.4, PHP 5)

shmop_open — 共有メモリブロックを作成またはオープンする

説明

int **shmop_open** (int $key , string $flags , int $mode , int $size)

shmop_open() は共有メモリブロックを作成または オープンします。

パラメータ

key

共有メモリブロックのシステム ID であり、10 進数または 16 進数で指定することが可能です。

flags

フラグに設定できる内容は、次のとおりです。

- "a" アクセス用(shmat に SHM_RDONLY を設定する) 既存の共有メモリセグメントを読み込み専用でオープンする必要がある場合に このフラグを使用してください。
- "c" 作成用(IPC_CREATE を設定する) 新規に共有メモリセグメントが必要な場合にこのフラグを使用してください。 もし同じキーのセグメントがすでに存在する場合、それを読み書きモードで オープンしようと試みます。
- "w" 読み込み & 書き込みアクセス用 共有メモリセグメントの読み込みや書き込みの必要がある場合にこのフラグを 使用してください。たいていの場合はこのフラグを使用します。
- "n" 新規メモリセグメントの作成用(IPC_CREAT|IPC_EXCL を設定する) 新規に共有メモリセグメントが必要で、もし同じフラグのセグメントが 存在するときには失敗させたい場合にこのフラグを使用してください。 セキュリティを確保するために、このフラグは有用です。これを使用する ことで、条件の競合による問題を避けることが可能です。

mode

共有メモリセグメントに設定したい 許可属性で、ファイルに関する許可属性と同様なものです。許可属性は、 例えば 0644 のような 8 進数形式で渡す必要があります。

size

作成したい共有メモリブロックの大きさをバイト数で指定します。

注意: 注意: 既存のメモリセグメントをオープンする場合には、3 番目および 4 番目の引数には 0 を指定する必要があります。成功時に

返り値

成功した場合は、**shmop_open()** は 作成した共有メモリセグメントにアクセスするために使用する ID を返します。 失敗した場合に **FALSE** を返します。

例

Example#1 共有メモリブロックを新規に作成する

```
<?php
$shm_key  = ftok(__FILE__, 't');
$shm_id  = shmop_open($shm_key, "c", 0644, 100);
?>
```

この例は、ftok() が返すシステム ID の共有メモリブロックをオープンします。

参考

- shmop_close()
- shmop_delete()

---

shmop_read

(PHP 4 >= 4.0.4, PHP 5)

shmop_read — 共有メモリブロックからデータを読み込む

説明

string **shmop_read** (int $shmid , int $start , int $count)

shmop_read() は共有メモリブロックから文字列を 読み込みます。

パラメータ

shmid

shmop_open() が作成した共有メモリブロックの識別子。

start

読み込みを開始する位置。

count

読み込むバイト数。

返り値

データ、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 共有メモリブロックを読み込む

```
<?php
$shm_data  = shmop_read($shm_id, 0, 50);
?>
```

この例は共有メモリブロックから 50 バイトを読み込んで、 $shm_data の中のデータに置くものです。

参考

- shmop_write()

---

shmop_size

(PHP 4 >= 4.0.4, PHP 5)

shmop_size — 共有メモリブロックの大きさを得る

説明

int **shmop_size** (int $shmid)

shmop_size() は共有メモリブロックの大きさを 得るために使用されます。

パラメータ

shmid

shmop_open() が作成した共有メモリブロックの識別子。

返り値

共有メモリブロックが占有するバイト数を表す整数を返します。

例

Example#1 共有メモリブロックの大きさを得る

```
<?php
$shm_size  = shmop_size($shm_id);
?>
```

この例は、$shm_id を ID とする共有メモリブロックの 大きさを $shm_size に代入します。

---

shmop_write

(PHP 4 >= 4.0.4, PHP 5)

shmop_write — 共有メモリブロックにデータを書き込む

説明

int **shmop_write** (int $shmid , string $data , int $offset)

shmop_write() は共有メモリブロックに文字列を 書き込みます。

パラメータ

shmid

shmop_open() が作成した共有メモリブロックの識別子。

data

共有メモリブロックに書き込む文字列。

offset

共有メモリセグメント内でデータを書き込み始める位置。

返り値

書き込んだ data の大きさ、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 共有メモリブロックに書き込む

```
<?php
$shm_bytes_written  = shmop_write($shm_id, $my_string, 0);
?>
```

この例は、共有メモリブロックに $my_string 内の データを書き込みます。$shm_bytes_written には、 書き込んだバイト数が代入されます。

参考

- shmop_read()

目次

SimpleXML関数

導入

SimpleXML拡張モジュールは、 XMLをオブジェクトにとても簡単かつ容易に変換するための機能を 提供します。変換後のオブジェクトでは、 通常の

プロパティセレクタや配列反復子を用いて処理を行うことが 可能です。

要件

SimpleXML 拡張モジュールは PHP 5 が必要になります。

インストール手順

SimpleXML 拡張モジュールは、デフォルトで利用可能です。 この機能を無効にするには、 --disable-simplexml コンフィギュアオプションを使用してください。

例

このリファレンスの多くの例ではXML文字列を必要とします。各例で この文字列をくり返す代わりに、あるファイルにこの文字列を保存して、 各例で読みこむことにします。この読みこまれるファイルは、以下の例 に関するセクションで使用されます。 もしくは、XMLドキュメントを作成し、simplexml_load_file() により読みこむことも 可能です。

Example#1 XML文字列を設定するインクルードファイル example.php

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<movies>
 <movie>
  <title>PHP: Behind the Parser</title>
  <characters>
   <character>
    <name>Ms. Coder</name>
    <actor>Onlivia Actora</actor>
   </character>
   <character>
    <name>Mr. Coder</name>
    <actor>El Act&#211;r</actor>
   </character>
  </characters>
  <plot>
   So, this language. It's like, a programming language. Or is it a
   scripting language? All is revealed in this thrilling horror spoof
   of a documentary.
  </plot>
  <great-lines>
   <line>PHP solves all my web problems</line>
  </great-lines>
  <rating type="thumbs">7</rating>
  <rating type="stars">5</rating>
 </movie>
</movies>
XML;
?>
```

SimpleXMLの容易さが最も明確に現われるのは、 簡単なXMLドキュメントから文字列または数字を展開する時です。

Example#2 <plot> を取得する

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

echo $xml->movie[0]->plot; // "So this language. It's like..."
?>
```

XML ドキュメント内の要素のうち、PHP の命名規約で許可されていない文字（たとえばハイフンなど）を含む名前のものにアクセスするには、 要素名を括弧とアポストロフィで囲みます。

Example#3 <line> を取得する

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

echo $xml->movie->{'great-lines'}->line; // "PHP solves all my web problems"
?>
```

Example#4 SimpleXMLでユニークでない要素にアクセスする

単一の親要素の子要素としてある要素のインスタンスが複数存在する時、 通常の反復処理を適用することができます。

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

/* 個々の <movie> ノードに対して、<plot> を分割して表示します */
foreach ($xml->movie as $movie) {
    echo $movie->plot, '<br />';
}

?>
```

Example#5 属性を使用する

ここまでは、要素の名前と値を読む方法のみを扱って来ました。 SimpleXMLは要素の属性にアクセスすることも可能です。 要素の属性にアクセスする方法は、配列 の要素に アクセスするのと全く同じです。

```
<?php
```

```
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

/* 最初の映画の <rating> ノードにアクセスします。
 * また、その評価も出力します。*/
foreach ($xml->movie[0]->rating as $rating) {
    switch((string) $rating['type']) { //  要素のインデックスとして、属性を取得します
    case 'thumbs':
        echo $rating, ' thumbs up';
        break;
    case 'stars':
        echo $rating, ' stars';
        break;
    }
}
?>
```

Example#6 要素および属性をテキストと比較する

要素または属性を文字列と比較する、もしくは、文字列を引数とする関数に 渡すには、(string) により文字列にキャストする 必要があります。さもないと、PHPはこの要素をオブジェクトとして扱います。

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

if ((string) $xml->movie->title == 'PHP: Behind the Parser') {
    print 'My favorite movie.';
}

htmlentities((string) $xml->movie->title);
?>
```

Example#7 XPath の使用

SimpleXML は、XPath を標準でサポートしています。 <character> 要素を全て見つけるには、 以下のようにします。

```
<?php
include 'example.php';
$xml = new SimpleXMLElement($xmlstr);

foreach ($xml->xpath('//character') as $character) {
    echo $character->name, 'played by ', $character->actor, '<br />';
}
?>
```

'//' はワイルドカードとして動作します。絶対パスを指定するには、 スラッシュを一つだけにします。

Example#8 値を設定する

SimpleXMLの中のデータは、定数とすることができません。 オブジェクトは、その全ての要素について変更が可能です。

```
<?php
include 'example.php';
$xml = new SimpleXMLElement($xmlstr);

$xml->movie[0]->characters->character[0]->name = 'Miss Coder';

echo $xml->asXML();
?>
```

上のコードは、元のXMLドキュメントと全く同じXMLドキュメントを新規に 出力しますが、新しいXMLファイルでは、Ms. Coder が Miss Coder に変更されているところが異なります。

Example#9 要素と属性を追加する

PHP 5.1.3 以降では、SimpleXML を使用して簡単に子要素および属性を追加することができます。

```
<?php
include 'example.php';
$xml = new SimpleXMLElement($xmlstr);

$character = $xml->movie[0]->characters->addChild('character');
$character->addChild('name', 'Mr. Parser');
$character->addChild('actor', 'John Doe');

$rating = $xml->movie[0]->addChild('rating', 'PG');
$rating->addAttribute('type', 'mpaa');

echo $xml->asXML();
?>
```

上のコードは、元と同じオブジェクトを出力しますが、 そこに新しいキャラクターと評価が追加されています。

Example#10 DOMとの相互運用性

PHPは、SimpleXML形式とDOM形式の間でXMLノードを変換する機構を有しています。 この例では、DOM要素をSimpleXMLに変換することができます。

```
<?php
$dom = new domDocument;
$dom->loadXML('<books><book><title>blah</title></book></books>');
if (!$dom) {
    echo 'Error while parsing the document';
    exit;
}

$s = simplexml_import_dom($dom);
```

```
echo $s->book[0]->title;
?>
```

SimpleXMLElement->addAttribute()

(PHP 5 >= 5.1.3)

SimpleXMLElement->addAttribute() — SimpleXML 要素に属性を追加する

説明

SimpleXMLElement
void **addAttribute** (string $name , string $value [, string $namespace])

SimpleXML 要素に属性を追加します。

パラメータ

name

追加する属性の名前。

value

属性の値。

namespace

指定されている場合は、その属性が所属する名前空間。

返り値

値を返しません。

例

Example#1 SimpleXML 要素への属性と子要素の追加

```
<?php

include 'example.php';

$sxe = new SimpleXMLElement($xmlstr);
$sxe->addAttribute('type', 'documentary');

$movie = $sxe->addChild('movie');
$movie->addChild('title', 'PHP2: More Parser Stories');
$movie->addChild('plot', 'This is all about the people who make it work.');

$characters = $movie->addChild('characters');
$character = $characters->addChild('character');
$character->addChild('name', 'Mr. Parser');
$character->addChild('actor', 'John Doe');

$rating = $movie->addChild('rating', '5');
$rating->addAttribute('type', 'stars');

echo $sxe->asXML();

?>
```

参考

- SimpleXMLElement->addChild()

SimpleXMLElement->addChild()

(PHP 5 >= 5.1.3)

SimpleXMLElement->addChild() — XML ノードに子要素を追加する

説明

SimpleXMLElement
SimpleXMLElement **addChild** (string $name [, string $value [, string $namespace]])

ノードに子要素を追加し、子要素の SimpleXMLElement を返します。

パラメータ

name

追加する子要素の名前。

value

指定されている場合は、子要素の値。

namespace

指定されている場合は、その子要素が所属する名前空間。

返り値

addChild メソッドは、 XML ノードに追加した子要素を表す SimpleXMLElement オブジェクトを返します。

例

Example#1 SimpleXML 要素への属性と子要素の追加

```
<?php

include 'example.php';

$sxe = new SimpleXMLElement($xmlstr);
$sxe->addAttribute('type', 'documentary');

$movie = $sxe->addChild('movie');
$movie->addChild('title', 'PHP2: More Parser Stories');
$movie->addChild('plot', 'This is all about the people who make it work.');

$characters = $movie->addChild('characters');
$character  = $characters->addChild('character');
$character->addChild('name', 'Mr. Parser');
$character->addChild('actor', 'John Doe');

$rating = $movie->addChild('rating', '5');
$rating->addAttribute('type', 'stars');

echo $sxe->asXML();

?>
```

参考

- SimpleXMLElement->addAttribute()

SimpleXMLElement->asXML()

(PHP 5 >= 5.0.1)

SimpleXMLElement->asXML() — SimpleXML 要素に基づき整形式の XML 文字列を返す

説明

SimpleXMLElement
mixed **asXML** *([string $filename])*

asXML メソッドは、親オブジェクトのデータを XML version 1.0 形式にフォーマットします。

パラメータ

filename

指定した場合、データを返すかわりにファイルにデータを書き込みます。

返り値

filename が指定されていない場合、この関数は 成功時に string 、エラー時に FALSE を返します。 パラメータが指定されていた場合は、ファイルが正常に書き込めたときに TRUE 、そうでないときに FALSE を返します。

例

Example#1 XML を取得する

```
<?php
$string = <<<XML
<a>
 <b>
  <c>text</c>
  <c>stuff</c>
 </b>
 <d>
  <c>code</c>
 </d>
</a>
XML;

$xml = new SimpleXMLElement($string);

echo $xml->asXML(); // <?xml ... <a><b><c>text</c><c>stuff</c> ...

?>
```

asXML は Xpath の結果にも適用できます:

Example#2 Xpath の結果に asXML() を使用する

```
<?php
```

```
// 上の XML の例から続く

/* <a><b><c>を探す */
$result = $xml->xpath('/a/b/c');

while(list( , $node) = each($result)) {
    echo $node->asXML(); // <c>text</c> と <c>stuff</c>
}
?>
```

SimpleXMLElement->attributes()

(PHP 5 >= 5.0.1)

SimpleXMLElement->attributes() — 要素の属性を定義する

説明

SimpleXMLElement
SimpleXMLElement **attributes** ([string $ns [, bool $is_prefix]])

この関数は、XMLタグの中で定義された属性とその値を取得します。

> **注意**: SimpleXML では、ほとんどのメソッドに反復処理を追加するための手順が定義されています。これらは、var_dump() やオブジェクトを評価する他の手段で見ることはできません。

パラメータ

ns

オプションで指定する、取得した属性の名前空間。

is_prefix

デフォルトは **FALSE**。

返り値

例

Example#1 XML文字列を解釈する

```
<?php
$string = <<<XML
<a xmlns:b>
  <foo name="one" game="lonely">1</foo>
</a>
XML;

$xml = simplexml_load_string($string);
foreach($xml->foo[0]->attributes() as $a => $b) {
    echo $a,'="',$b,"¥"¥n";
}
?>
```

上の例の出力は以下となります。

```
name="one"
game="lonely"
```

SimpleXMLElement->children()

(PHP 5 >= 5.0.1)

SimpleXMLElement->children() — 指定したノードの子ノードを見付ける

説明

SimpleXMLElement
SimpleXMLElement **children** ([string $ns [, bool $is_prefix]])

このメソッドは、指定した要素のメンバーである子を見つけます。 結果は、通常の反復子により取得できます。

> **注意**: SimpleXML では、ほとんどのメソッドに反復処理を追加するための手順が定義されています。これらは、var_dump() やオブジェクトを評価する他の手段で見ることはできません。

パラメータ

ns

is_prefix

デフォルトは **FALSE**。

返り値

変更履歴

バージョン	説明
5.2.0	オプションのパラメータ *is_prefix* が追加されました。

例

Example#1 children() 疑似配列を走査する

```
<?php
$xml = new SimpleXMLElement(
'<person>
  <child role="son">
    <child role="daughter"/>
  </child>
  <child role="daughter">
    <child role="son">
      <child role="son"/>
    </child>
  </child>
</person>');

foreach ($xml->children() as $second_gen) {
    echo ' The person begot a ' . $second_gen['role'];

    foreach ($second_gen->children() as $third_gen) {
        echo ' who begot a ' . $third_gen['role'] . ';';

        foreach ($third_gen->children() as $fourth_gen) {
            echo ' and that ' . $third_gen['role'] .
                ' begot a ' . $fourth_gen['role'];
        }
    }
}
?>
```

上の例の出力は以下となります。

```
The person begot a son who begot a daughter; The person
begot a daughter who begot a son; and that son begot a son
```

SimpleXMLElement->__construct()

(No version information available, might be only in CVS)

SimpleXMLElement->__construct() — 新しい *SimpleXMLElement* オブジェクトを作成する

説明

SimpleXMLElement
__construct (*string $data [, int $options [, bool $data_is_url [, string $ns [, bool $is_prefix]]]]*)

新しい *SimpleXMLElement* オブジェクトを作成します。

パラメータ

data

整形式 *XML* 文字列。 もし *data_is_url* が **TRUE** の場合には、*XML* ドキュメントへのパスあるいは *URL*。

options

オプションで、*追加の Libxml パラメータ*を指定するために使用します。

data_is_url

デフォルトでは *data_is_url* は **FALSE** です。 *data* が、文字列データではなく *XML* ドキュメントへのパスあるいは *URL* である場合に **TRUE** を使用します。

ns

is_prefix

返り値

data を表す *SimpleXMLElement* オブジェクトを返します。

エラー / 例外

XML データ内でエラーが見つかるたびに **E_WARNING** エラーメッセージが発生します。そしてエラーが見つかった場合は例外をスローします。

例

Example#1 SimpleXMLElement オブジェクトの作成

```
<?php
```

```
include 'example.php';

$sxe = new SimpleXMLElement($xmlstr);
echo $sxe->movie[0]->title;

?>
```

Example#2 URL からの SimpleXMLElement オブジェクトの作成

```
<?php

$sxe = new SimpleXMLElement('http://example.org/document.xml', NULL, TRUE);
echo $sxe->asXML();

?>
```

参考

- simplexml_load_string
- simplexml_load_file

SimpleXMLElement->getDocNamespaces()

(PHP 5 >= 5.1.2)

SimpleXMLElement->getDocNamespaces() — ドキュメントで宣言されている名前空間を返す

説明

SimpleXMLElement
array **getDocNamespaces** ([bool $recursive])

ドキュメントで宣言されている名前空間を返します。

パラメータ

recursive

指定されている場合は、親ノードおよび子ノードで宣言されている全ての名前空間を返します。 されていない場合は、ルートノードで宣言されている名前空間のみを返します。

返り値

getDocNamespaces メソッドは、 名前空間名および関連付けられた URI を配列で返します。

例

Example#1 ドキュメントの名前空間の取得

```
<?php

$xml = <<<XML
<?xml version="1.0" standalone="yes"?>
<people xmlns:p="http://example.org/ns">
      <p:person id="1">John Doe</p:person>
      <p:person id="2">Susie Q. Public</p:person>
</people>
XML;

$sxe = new SimpleXMLElement($xml);

$namespaces = $sxe->getDocNamespaces();
var_dump($namespaces);

?>
```

Example#2 複数の名前空間の使用

```
<?php

$xml = <<<XML
<?xml version="1.0" standalone="yes"?>
<people xmlns:p="http://example.org/ns" xmlns:t="http://example.org/test">
      <p:person t:id="1">John Doe</p:person>
       <p:person t:id="2" a:addr="123 Street" xmlns:a="http://example.org/addr">
          Susie Q. Public
     </p:person>
</people>
XML;

$sxe = new SimpleXMLElement($xml);

$namespaces = $sxe->getDocNamespaces(TRUE);
var_dump($namespaces);

?>
```

参考

- SimpleXMLElement->getNamespaces()

- SimpleXMLElement->registerXPathNamespace()

SimpleXMLElement->getName()

(PHP 5 >= 5.1.3)

SimpleXMLElement->getName() — XML 要素の名前を取得する

説明

SimpleXMLElement
string **getName** (void)

XML 要素の名前を取得します。

返り値

getName メソッドは、 SimpleXMLElement オブジェクトが参照している XML タグの名前を string で返します。

例

Example#1 XML 要素の名前の取得

```
<?php

$sxe = new SimpleXMLElement($xmlstr);

echo $sxe->getName() . "¥n";

foreach ($sxe->children() as $child)
{
    echo $child->getName() . "¥n";
}

?>
```

SimpleXMLElement->getNamespaces()

(PHP 5 >= 5.1.2)

SimpleXMLElement->getNamespaces() — ドキュメントで使用している名前空間を返す

説明

SimpleXMLElement
array **getNamespaces** ([bool $recursive])

ドキュメントで使用している名前空間を返します。

パラメータ

recursive

指定されている場合は、親ノードおよび子ノードで使用している全ての名前空間を返します。 されていない場合は、ルートノードで使用している名前空間のみを返します。

返り値

getNamespaces メソッドは、 名前空間名および関連付けられた URI を配列で返します。

例

Example#1 ドキュメントで使用している名前空間の取得

```
<?php

$xml = <<<XML
<?xml version="1.0" standalone="yes"?>
<people xmlns:p="http://example.org/ns" xmlns:t="http://example.org/test">
      <p:person id="1">John Doe</p:person>
      <p:person id="2">Susie Q. Public</p:person>
</people>
XML;

$sxe = new SimpleXMLElement($xml);

$namespaces = $sxe->getNamespaces(true);
var_dump($namespaces);

?>
```

上の例の出力は以下となります。

```
array(1) {
   ["p"]=>
    string(21) "http://example.org/ns"
}
```

参考

- SimpleXMLElement->getDocNamespaces()
- SimpleXMLElement->registerXPathNamespace()

SimpleXMLElement->registerXPathNamespace()

(PHP 5 >= 5.2.0)

SimpleXMLElement->registerXPathNamespace() — 次の XPath クエリ用の prefix/ns コンテキストを作成する

説明

SimpleXMLElement
bool **registerXPathNamespace** (string $prefix , string $ns)

次の XPath クエリ用の prefix/ns コンテキストを作成します。特にこれが有用なのは、 XML ドキュメントの提供者が名前空間プレフィックスを変更したような場合です。 registerXPathNamespace はプレフィックスを作成して名前空間に関連付け、 そのプレフィックスで名前空間のノードにアクセスできるようにします。 提供者側がプレフィックスを変更したとしても、コードを書き換える必要はありません。

パラメータ

prefix

ns で指定した名前空間への XPath クエリで使用する、 名前空間プレフィックス。

ns

XPath クエリで使用する名前空間。 これは XML ドキュメントで使用している名前空間と一致していなければなりません。 一致していない場合、 prefix を使用した XPath クエリは何も結果を返しません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 XPath クエリで使用する名前空間プレフィックスの設定

```
<?php

$xml = <<<EOD
<book xmlns:chap="http://example.org/chapter-title">
      <title>My Book</title>
      <chapter id="1">
            <chap:title>Chapter 1</chap:title>
            <para>Donec velit. Nullam eget tellus vitae tortor gravida scelerisque.
                 In orci lorem, cursus imperdiet, ultricies non, hendrerit et, orci.
                 Nulla facilisi. Nullam velit nisl, laoreet id, condimentum ut,
                ultricies id, mauris.</para>
      </chapter>
      <chapter id="2">
            <chap:title>Chapter 2</chap:title>
            <para>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin
                 gravida. Phasellus tincidunt massa vel urna. Proin adipiscing quam
                vitae odio. Sed dictum. Ut tincidunt lorem ac lorem. Duis eros
                 tellus, pharetra id, faucibus eu, dapibus dictum, odio.</para>
      </chapter>
</book>
EOD;

$sxe = new SimpleXMLElement($xml);

$sxe->registerXPathNamespace('c', 'http://example.org/chapter-title');
$result = $sxe->xpath('//c:title');

foreach ($result as $title) {
  echo $title . "¥n";
}

?>
```

上の例の XML ドキュメントでは、プレフィックス chap で名前空間を指定していることを確認しておきましょう。仮に、このドキュメント（あるいはよく似た別のドキュメント）が以前に同じ名前空間に対してプレフィックス c を使用していたとしましょう。プレフィックスが変わった時点で、これまでの XPath クエリは正しい値を返さないようになります。 そしてクエリに対して何らかの変更が必要となります。
registerXPathNamespace を使用すると、 仮に名前空間プレフィックスが変更された場合でもクエリの変更する必要がなくなります。

参考

- SimpleXMLElement->getDocNamespaces()
- SimpleXMLElement->getNamespaces()

SimpleXMLElement->xpath()

(PHP 5 >= 5.2.0)

SimpleXMLElement->xpath() — XML データに Xpath クエリを実行する

説明

SimpleXMLElement
array **xpath** (string $path)

xpathメソッドは、 XPath path にマッチする SimpleXML ノードを検索します。

パラメータ

path

XPath パス。

返り値

SimpleXMLElement オブジェクトの配列を返します。 エラーが発生した場合は **FALSE** を返します。

例

Example#1 Xpath

```
<?php
$string = <<<XML
<a>
  <b>
    <c>text</c>
    <c>stuff</c>
  </b>
  <d>
    <c>code</c>
  </d>
</a>
XML;

$xml = new SimpleXMLElement($string);

/* <a><b><c> を探します */
$result = $xml->xpath('/a/b/c');

while(list( , $node) = each($result)) {
     echo '/a/b/c:  ',$node,"¥n";
}

/* 相対パスでも動作します... */
$result = $xml->xpath('b/c');

while(list( , $node) = each($result)) {
     echo 'b/c:  ',$node,"¥n";
}
?>
```

上の例の出力は以下となります。

```
/a/b/c:  text
/a/b/c:  stuff
b/c:  text
b/c:  stuff
```

これら二つの結果は同じであることに注意してください。

---

simplexml_import_dom

(PHP 5)

simplexml_import_dom — DOM ノードから SimpleXMLElement オブジェクトを取得する

説明

SimpleXMLElement **simplexml_import_dom** (DOMNode $node [, string $class_name])

この関数は、DOM ドキュメントのノードを引数とし、 SimpleXML ノードを作成します。この新しいオブジェクトは、 この後、通常の SimpleXML 要素として使用できます。

パラメータ

node

DOM 要素ノード。

class_name

このオプションパラメータを使用すると、 simplexml_load_string() は指定したクラスのオブジェクトを返します。このクラスは SimpleXMLElement を継承していなければなりません。

返り値

SimpleXMLElement、あるいは失敗した場合に FALSE を返します。

例

Example#1 DOM のインポート

```
<?php
$dom = new DOMDocument;
$dom->loadXML('<books><book><title>blah</title></book></books>');
if (!$dom) {
    echo 'ドキュメントのパース時にエラーが発生しました';
     exit;
}

$s = simplexml_import_dom($dom);

echo $s->book[0]->title; // blah
?>
```

参考

- dom_import_simplexml()

---

simplexml_load_file

(PHP 5)

simplexml_load_file — XMLファイルをパースし、オブジェクトに代入する

説明

object **simplexml_load_file** (string $filename [, string $class_name [, int $options [, string $ns [, bool $is_prefix]]]])

指定したファイルの中の整形式 XML ドキュメントをオブジェクトに変換します。

パラメータ

filename

XML ファイルへのパス。

> **注意**: Libxml 2 は URI をエスケープしませんので、例えば URI パラメータ a に b&c を渡したい場合、simplexml_load_file(rawurlencode('http://example.com/?a=' . urlencode('b&c'))) をしてコールする必要があります。PHP 5.1.0 以降では、これをする必要は ありません。PHP が自動的に行います。

class_name

simplexml_load_file() が指定されたクラスのオブジェクトを返すようにするために、 このオプションのパラメータを使用します。 このクラスは、SimpleXMLElement クラスを継承していなければなりません。

options

PHP 5.1.0 と Libxml 2.6.0 から、追加の Libxml パラメータ を指定するために options を使用することもできます。

ns

is_prefix

返り値

SimpleXMLElement クラスのオブジェクトを返します。 XML ドキュメント内のデータをプロパティに含みます。 エラー時には **FALSE** を返します。

例

Example#1 XMLドキュメントをパースする

```
<?php
// The file test.xml contains an XML document with a root element
// and at least an element /[root]/title.

if (file_exists('test.xml')) {
    $xml = simplexml_load_file('test.xml');

    print_r($xml);
} else {
    exit('Failed to open test.xml.');
}
?>
```

このスクリプトは成功時に以下のように出力します。

```
SimpleXMLElement Object
(
   [title] => Example Title
   ...
```

```
)
```

この時点で、$xml->title としたり、 他の全ての要素にアクセスすることができます。

参考

- simplexml_load_string
- SimpleXMLElement->__construct()

---

simplexml_load_string

(PHP 5)

simplexml_load_string — XML 文字列をオブジェクトに代入する

説明

```
object simplexml_load_string ( string $data [, string $class_name [, int $options [, string $ns [, bool $is_prefix ]]]]
)
```

整形式 XML 文字列をオブジェクトとして返します。

パラメータ

data

整形式 XML 文字列。

class_name

このオプションのパラメータを使用して、 simplexml_load_file() が指定されたクラスのオブジェクトを返すようにします。 このクラスは、SimpleXMLElement クラスを継承していなければなりません。

options

PHP 5.1.0 と Libxml 2.6.0 から、追加の Libxml パラメータ を指定するために options を使用することもできます。

ns

is_prefix

返り値

SimpleXMLElement クラスのオブジェクトを返します。 XML ドキュメント内のデータをプロパティに含みます。 エラー時には FALSE を返します。

例

Example#1 XML 文字列をパースする

```
<?php
$string = <<<XML
<?xml version='1.0'?>
<document>
  <title>Forty What?</title>
  <from>Joe</from>
  <to>Jane</to>
  <body>
   I know that's the answer -- but what's the question?
  </body>
</document>
XML;

$xml = simplexml_load_string($string);

var_dump($xml);
?>
```

上の例の出力は以下となります。

```
SimpleXMLElement Object
(
   [title] => Forty What?
   [from] => Joe
   [to] => Jane
   [body] =>
    I know that's the answer -- but what's the question?
)
```

この時点で、$xml->body のようにアクセスすることができます。

参考

- simplexml_load_file
- SimpleXMLElement->__construct()

目次

SNMP 関数

導入

要件

Unix 上で SNMP 関数を使用するためには、» NET-SNMP パッケージをインストールする必要があります。 Windows 上ではこれらの関数は NT 上でのみ利用可能であり、 Win95/98 では利用できません。

インストール手順

重要: UCD SNMP パッケージを使用するには、コンパイル前に NO_ZEROLENGTH_COMMUNITY を 1 に定義する必要があります。UCD SNMP のコンフィグレーションを行った後で config.h あるいは acconfig.h を編集し、NO_ZEROLENGTH_COMMUNITY を探してください。 #define の行のコメントを外してください。 これにより、次のようになるはずです。

```
#define  NO_ZEROLENGTH_COMMUNITY  1
```

それから、--with-snmp[=DIR] をつけて PHP をコンパイルします。

SNMP コマンドと組み合わせた場合に奇妙なセグメンテーションフォールトが 生じる場合は、上の指示に従っていないと思われます。UCD SNMP を再コンパイル したくない場合は、この不具合に対処するため、PHP を --enable-ucd-snmp-hack スイッチをつけてコンパイルしてください。

Windows 版は、SNMP サポート用ファイルが mibs ディレクトリにあります。このディレクトリを DRIVE:¥usr¥mibs に移動する必要が あります。DRIVE は、PHP がインストールされているドライブに置き換えなければ なりません。例: c:¥usr¥mibs

実行時設定

設定ディレクティブは定義されていません。

リソース型

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

SNMP_OID_OUTPUT_FULL (integer)
5.2.0 以降
SNMP_OID_OUTPUT_NUMERIC (integer)
5.2.0 以降
SNMP_VALUE_LIBRARY (integer)
SNMP_VALUE_PLAIN (integer)
SNMP_VALUE_OBJECT (integer)
SNMP_BIT_STR (integer)
SNMP_OCTET_STR (integer)
SNMP_OPAQUE (integer)
SNMP_NULL (integer)
SNMP_OBJECT_ID (integer)
SNMP_IPADDRESS (integer)
SNMP_COUNTER (integer)
SNMP_UNSIGNED (integer)
SNMP_TIMETICKS (integer)
SNMP_UINTEGER (integer)
SNMP_INTEGER (integer)
SNMP_COUNTER64 (integer)

---

snmp_get_quick_print

(PHP 4, PHP 5)

snmp_get_quick_print — UCD ライブラリの quick_print の現在の設定値を取得する

説明

bool **snmp_get_quick_print** (void)

UCD ライブラリに保持された quick_print の現在の値を返します。 デフォルトでは、quick_print はオフです。

返り値

quick_print がオンの場合に **TRUE**、それ以外の場合に **FALSE** を返します。

例

Example#1 snmp_get_quick_print() の例

```
<?php
$quickprint = snmp_get_quick_print();
?>
```

参考

- quick_print の役割に関する詳細な説明は、 snmp_set_quick_print() を参照ください。

---

snmp_get_valueretrieval

(PHP 4 >= 4.3.3, PHP 5)

snmp_get_valueretrieval — SNMP の値が返される方法を返す

説明

int **snmp_get_valueretrieval** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- snmp_set_valueretrieval()

---

snmp_read_mib

(PHP 5)

snmp_read_mib — アクティブな MIB ツリーの中に MIB ファイルを読み込んでパースする

説明

bool **snmp_read_mib** (string $filename)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

snmp_set_enum_print

(PHP 4 >= 4.3.0, PHP 5)

snmp_set_enum_print — すべての enum を、実際の整数値ではなく enum 値とともに返す

説明

void **snmp_set_enum_print** (int $enum_print)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意

注意: **snmp_set_enum_print()** は、 UCB SNMP ライブラリを使用している場合のみ利用可能です。 この関数は、Windows SNMP ライブラリを使用している場合は利用できません。

---

snmp_set_oid_numeric_print

(PHP 4 >= 4.3.0, PHP 5)

snmp_set_oid_numeric_print — 指定したオブジェクト内の全てのオブジェクトを、対応するオブジェクト ID を含めて返す

説明

void ***snmp_set_oid_numeric_print*** *(int $oid_numeric_print)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

変更履歴

バージョン	説明
5.2.0	PHP 5.2.0 以降では、 この関数は次の関数のエイリアスです。 snmp_set_oid_output_format()

参考

- *snmp_set_oid_output_format()*

---

snmp_set_oid_output_format

(PHP 5 >= 5.2.0)

snmp_set_oid_output_format — OID の出力形式を設定する

説明

void ***snmp_set_oid_output_format*** *(int $oid_format)*

snmp_set_oid_output_format() *は、 出力形式を完全か数値のいずれかに設定します。*

パラメータ

oid_format

完全な出力がほしい場合に ***SNMP_OID_OUTPUT_FULL***、 *それ以外の場合に* ***SNMP_OID_OUTPUT_NUMERIC*** *を指定します。*

返り値

値を返しません。

注意

> ***注意***: ***snmp_set_oid_output_format()*** *は、 UCB SNMP ライブラリを使用している場合のみ利用可能です。 この関数は、Windows SNMP ライブラリを使用している場合は利用できません。*

---

snmp_set_quick_print

(PHP 4, PHP 5)

snmp_set_quick_print — UCB SNMP ライブラリで quick_print の値を設定する

説明

void ***snmp_set_quick_print*** *(bool $quick_print)*

UCB SNMP ライブラリで quick_print の値を設定します。この値を (1) に設定した場合、SNMP ライブラリは、'簡潔に表示された (quick printed)' 値を返します。 これは、値のみが出力されることを意味します。 quick_print が有効でない場合(デフォルト)、UCD SNMP ライブラリは、(IP アドレスまたは OID のような)その値の種類を含む、より詳細な情報を 出力します。 加えて、quick_print が有効でない場合、ライブラリは 3 文字以下の全ての 文字列について 16 進数値も出力します。

デフォルトで、UCD SNMP ライブラリは冗長な値を返します。 quick_print は、値のみを返すために用いられます。

現在、文字列は引用符で括られて返されますが、 この動作は将来のリリースでは修正される予定です。

パラメータ

quick_print

返り値

値を返しません。

例

quick_print の設定は、返される情報を吟味してから表示する場合に 使用されることが多いです。

Example#1 snmp_set_quick_print() の使用例

```
<?php
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a¥n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a¥n";
```

```
?>
```

上の例の出力は、たとえば以下のようになります。

```
'Timeticks: (0) 0:00:00.00'
'0:00:00.00'
```

参考

- snmp_get_quick_print()

---

snmp_set_valueretrieval

(PHP 4 >= 4.3.3, PHP 5)

snmp_set_valueretrieval — SNMP の値が返される方法を設定する

説明

void **snmp_set_valueretrieval** (int $method)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- snmp_get_valueretrieval()

---

snmpget

(PHP 4, PHP 5)

snmpget — SNMP オブジェクトを取得する

説明

string **snmpget** (string $hostname , string $community , string $object_id [, int $timeout [, int $retries]])

snmpget() は object_id で指定した SNMP オブジェクトの値を読みとるために使用されます。

パラメータ

hostname

SNMP エージェント。

community

リードコミュニティ。

object_id

SNMP オブジェクト。

timeout

retries

返り値

成功した場合に SNMP オブジェクトの値、失敗した場合に **FALSE** を返します。

例

Example#1 snmpget() の使用例

```
<?php
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0");
?>
```

参考

- snmpset()

---

snmpgetnext

(PHP 5)

snmpgetnext — SNMP オブジェクトを取得する

説明

string **snmpgetnext** (string $host , string $community , string $object_id [, int $timeout [, int $retries]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

snmprealwalk

(PHP 4, PHP 5)

snmprealwalk — 指定したオブジェクトに関するオブジェクト ID を含むすべてのオブジェクトを返す

説明

array **snmprealwalk** (string $host , string $community , string $object_id [, int $timeout [, int $retries]])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

snmpset

(PHP 4, PHP 5)

snmpset — SNMP オブジェクトを設定する

説明

bool **snmpset** (string $hostname , string $community , string $object_id , string $type , mixed $value [, int $timeout [, int $retries]])

snmpset() 関数は、 object_id で指定した SNMP オブジェクトの 値を設定するために使用します。

パラメータ

hostname

SNMP エージェント。

community

リードコミュニティ。

object_id

SNMP オブジェクト。

type

value

timeout

retries

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- snmpget()

---

snmpwalk

(PHP 4, PHP 5)

snmpwalk — エージェントから全ての SNMP オブジェクトを取得する

説明

array **snmpwalk** (string $hostname , string $community , string $object_id [, int $timeout [, int $retries]])

snmpwalk() 関数は、hostname で指定した SNMP エージェントから全ての値を読みとるのに使用します。

パラメータ

hostname

SNMP エージェント。

community

リードコミュニティ。

object_id

NULL の場合は、*object_id* が SNMP オブジェクトツリーのルートとして解釈され、 ツリーの配下のすべてのオブジェクトを配列として返します。

object_id を指定した場合は、 その *object_id* の配下のすべての SNMP オブジェクトを返します。

timeout

retries

返り値

object_id() からの SNMP オブジェクトの値の配列をルートとして返します。 エラーの場合に **FALSE** を返します。

例

Example#1 snmpwalk() の例

```
<?php
$a = snmpwalk("127.0.0.1", "public", "");

foreach ($a as $val) {
    echo "$val¥n";
}

?>
```

上記の関数コールは、ローカルホスト上で稼働する SNMP エージェントからすべての SNMP オブジェクトを返します。 ループ処理により値を一つずつとりだすことができます。

参考

- snmpwalkoid()

---

snmpwalkoid

(PHP 4, PHP 5)

snmpwalkoid — ネットワークエンティティに関する情報ツリーを検索する

説明

array **snmpwalkoid** (string $hostname , string $community , string $object_id [, int $timeout [, int $retries]])

snmpwalkoid() 関数は、hostname で指定した SNMP エージェントから すべてのオブジェクト ID とその値を読みこむために使用します。

snmpwalkoid() および snmpwalk() は、歴史的経緯により残されているものです。 どちらも、下位互換のために提供されています。 代わりに snmprealwalk() を使用してください。

パラメータ

hostname

SNMP エージェント。

community

リードコミュニティ。

object_id

NULL の場合は、*object_id* が SNMP オブジェクトツリーのルートとして解釈され、 ツリーの配下のすべてのオブジェクトを配列として返します。

object_id を指定した場合は、 その *object_id* の配下のすべての SNMP オブジェクトを返します。

timeout

retries

返り値

object_id() からの SNMP オブジェクトの値の配列をルートとして返します。 エラーの場合に **FALSE** を返します。

例

Example#1 snmpwalkoid() の例

```
<?php
$a = snmpwalkoid("127.0.0.1", "public", "");
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br />¥n";
}
?>
```

上記の関数コールは、ローカルホスト上で稼働する SNMP エージェントからすべての SNMP オブジェクトを返します。 ループ処理により値を一つずつとりだすことができます。

参考

- snmpwalk()

目次

SOAP関数

導入

SOAP拡張モジュールは、 SOAPサーバーおよびクライアントを書くために使用することができます。 本拡張モジュールは、» SOAP 1.1、 » SOAP 1.2 および » WSDL 1.1 規約のサブセットをサポートします。

要件

本拡張モジュールは、 » GNOME xml library を必要とします。 このライブラリをダウンロード／インストールしてください。 libxml-2.5.4 以上が必要です。

インストール手順

本拡張モジュールは、 --enable-soapを指定して PHPのconfigureが行われた場合のみ利用可能です。

実行時設定

php.ini の設定により動作が変化します。

SOAP 設定オプション

名前	デフォルト	変更の可否	変更履歴
soap.wsdl_cache_enabled	"1"	PHP_INI_ALL	PHP 5.0.0 から利用可能
soap.wsdl_cache_dir	"/tmp"	PHP_INI_ALL	PHP 5.0.0 から利用可能
soap.wsdl_cache_ttl	"86400"	PHP_INI_ALL	PHP 5.0.0 から利用可能
soap.wsdl_cache_limit	"5"	PHP_INI_ALL	PHP 5.1.5 から利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

soap.wsdl_cache_enabled boolean

WSDLキャッシュ機能有効または無効にします。

soap.wsdl_cache_dir string

SOAP 拡張モジュールがキャッシュファイルを置くディレクトリの名前を 設定します。

soap.wsdl_cache_ttl int

キャッシュされたファイルが元のファイルの代わりに使用される秒数（有効期間）を設定します。

soap.wsdl_cache_limit integer

wsdl ファイルのキャッシュに使用するメモリの最大数を設定します。

定義済みクラス

SoapClient

コンストラクタ

- SoapClient->__construct() - constructs a new SoapClient object

メソッド

- SoapClient->__call() - SOAP 関数をコールする（推奨されません）
- SoapClient->__doRequest() - SOAP リクエストを実行する
- SoapClient->__getFunctions() - SOAP 関数の一覧を返す
- SoapClient->__getLastRequest() - 直近の SOAP リクエストを返す
- SoapClient->__getLastRequestHeaders() - 直近の SOAP リクエストヘッダを返す
- SoapClient->__getLastResponse() - 直近の SOAP レスポンスを返す
- SoapClient->__getLastResponseHeaders() - 直近の SOAP レスポンスヘッダを返す
- SoapClient->__getTypes() - SOAP 型の一覧を返す
- SoapClient->__setCookie() - SOAP リクエストと共に送信されるクッキーを設定する
- SoapClient->__soapCall() - SOAP 関数をコールする

SoapFault

コンストラクタ

- SoapFault->__construct() - 新規 SoapFault オブジェクトを生成する

SoapHeader

SoapHeader は、SOAP ヘッダを渡すもしくは返すための特別な低レベルクラスです。 このクラスは単純なデータホルダーで、 コンストラクタ以外の特別なメソッドを持ちません。このクラスは SOAP ヘッダを渡すための SoapClient->__soapCall() 中、もしくは SOAP レスポンスにおけるヘッダを返すための SOAP ヘッダハンドラ中で使用されます。

コンストラクタ

- SoapHeader->__construct() - 新規 SoapHeader オブジェクトを生成する

SoapParam

SoapParam は、非 WSDL モードにおけるパラメータの名前付け、 もしくは値を返すための特別な低レベルクラスです。 このクラスは単純なデータホルダーで、 コンストラクタ以外の特別なメソッドを持ちません。

コンストラクタ

- SoapParam->__construct() - 新規 SoapParam オブジェクトを生成する

SoapServer

コンストラクタ

- SoapServer->__construct() - 新規 SoapServer オブジェクトを生成する

メソッド

- SoapServer->addFunction() - SOAP リクエストによって処理される単一もしくはいくつかの関数を追加する
- SoapServer->fault() -
- SoapServer->getFunctions() - 定義されている関数の一覧を返す
- SoapServer->handle() - SOAP リクエストを処理する
- SoapServer->setClass() - SOAP リクエストを処理するクラスを設定する
- SoapServer->setPersistence() - SoapServer の持続モードを設定する

SoapVar

SoapVar は、非 WSDL モードにおけるパラメータのエンコード、 もしくは値を返すための特別な低レベルクラスです。 このクラスは単純なデータホルダーで、 コンストラクタ以外の特別なメソッドを持ちません。 このクラスは SOAP リクエストもしくはレスポンスにおける型プロパティを 設定したい場合に有効です。

コンストラクタ

- SoapVar->__construct() - 新規 SoapVar オブジェクトを生成する

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

SOAP_1_1 (integer)
SOAP_1_2 (integer)
SOAP_PERSISTENCE_SESSION (integer)
SOAP_PERSISTENCE_REQUEST (integer)
SOAP_FUNCTIONS_ALL (integer)
SOAP_ENCODED (integer)
SOAP_LITERAL (integer)
SOAP_RPC (integer)
SOAP_DOCUMENT (integer)
SOAP_ACTOR_NEXT (integer)
SOAP_ACTOR_NONE (integer)
SOAP_ACTOR_UNLIMATERECEIVER (integer)
SOAP_COMPRESSION_ACCEPT (integer)
SOAP_COMPRESSION_GZIP (integer)
SOAP_COMPRESSION_DEFLATE (integer)
SOAP_WAIT_ONE_WAY_CALLS (integer)
PHP 5.1.0 で追加されました。
UNKNOWN_TYPE (integer)
XSD_STRING (integer)
XSD_BOOLEAN (integer)
XSD_DECIMAL (integer)
XSD_FLOAT (integer)
XSD_DOUBLE (integer)
XSD_DURATION (integer)
XSD_DATETIME (integer)
XSD_TIME (integer)
XSD_DATE (integer)
XSD_GYEARMONTH (integer)
XSD_GYEAR (integer)
XSD_GMONTHDAY (integer)
XSD_GDAY (integer)
XSD_GMONTH (integer)
XSD_HEXBINARY (integer)
XSD_BASE64BINARY (integer)
XSD_ANYURI (integer)
XSD_ANYXML (integer)
PHP 5.1.0 で追加されました。
XSD_QNAME (integer)
XSD_NOTATION (integer)
XSD_NORMALIZEDSTRING (integer)
XSD_TOKEN (integer)
XSD_LANGUAGE (integer)
XSD_NMTOKEN (integer)
XSD_NAME (integer)
XSD_NCNAME (integer)
XSD_ID (integer)
XSD_IDREF (integer)
XSD_IDREFS (integer)
XSD_ENTITY (integer)
XSD_ENTITIES (integer)
XSD_INTEGER (integer)
XSD_NONPOSITIVEINTEGER (integer)
XSD_NEGATIVEINTEGER (integer)
XSD_LONG (integer)
XSD_INT (integer)
XSD_SHORT (integer)
XSD_BYTE (integer)
XSD_NONNEGATIVEINTEGER (integer)
XSD_UNSIGNEDLONG (integer)
XSD_UNSIGNEDINT (integer)
XSD_UNSIGNEDSHORT (integer)
XSD_UNSIGNEDBYTE (integer)
XSD_POSITIVEINTEGER (integer)
XSD_NMTOKENS (integer)
XSD_ANYTYPE (integer)
SOAP_ENC_OBJECT (integer)
SOAP_ENC_ARRAY (integer)
XSD_1999_TIMEINSTANT (integer)
XSD_NAMESPACE (string)
XSD_1999_NAMESPACE (string)

---

is_soap_fault

(PHP 5)

is_soap_fault — SOAP コールが失敗したかどうかを調べる

説明

bool **is_soap_fault** (mixed $obj)

この関数は、SOAP コールが失敗したかどうかを調べたいが、例外を使用したくない 場合に有用です。 この関数を使用するには、オプション exceptions に ゼロまたは **FALSE** を指定して SoapClient オブジェクトを作成する必要があります。 この場合、SOAP メソッドは、特別な SoapFault オブジェクトを返します。 このオブジェクトには、フォルトの詳細 (faultcode, faultstring, faultactor および faultdetails) が含まれています。

exceptions が設定されていない場合、 SOAPコールは、エラー時に例外を投げます。 **is_soap_fault()** は指定したパラメータ SoapFault オブジェクトであるかどうかを調べます。

パラメータ

obj

検査するオブジェクト

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 is_soap_fault() の例

```
<?php
$client = new SoapClient("some.wsdl", array('exceptions' => 0));
$result = $client->SomeFunction();
if (is_soap_fault($result)) {
    trigger_error("SOAP Fault: (faultcode: {$result->faultcode}, faultstring: {$result->faultstring})", E_USER_ERROR);
}
?>
```

Example#2 SOAP の標準的なエラーレポートメソッドは例外となる

```
<?php
try {
    $client = new SoapClient("some.wsdl");
    $result = $client->SomeFunction(/* ... */);
} catch (SoapFault $fault) {
    trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring: {$fault->faultstring})", E_USER_ERROR);
}
?>
```

参考

- SoapClient->__construct()
- SoapFault->__construct()

SoapClient->__call()

(PHP 5 >= 5.0.1)

SoapClient->__call() — SOAP 関数をコールする (推奨されません)

説明

SoapClient
mixed **__call** (string $function_name , array $arguments [, array $options [, array $input_headers [, array $output_headers]]])

このメソッドは推奨されません。代わりに SoapClient->__soapCall() を使用してください。

SoapClient->__construct()

(No version information available, might be only in CVS)

SoapClient->__construct() — SoapClient コンストラクタ

説明

SoapClient
__construct (mixed $wsdl [, array $options])

このコンストラクタは、WSDL モードもしくは 非 WSDL モードで SoapClient オブジェクトを生成します。

パラメータ

wsdl

WSDL ファイルの URI もしくは 非 WSDL モードの場合 **NULL**

> **注意**: 開発中は、WSDL のキャッシュを php.ini の soap.wsdl_cache_ttl で無効にしておくとよいでしょう。 そうしないと、WSDL を変更しても soap.wsdl_cache_ttl で設定した時間が経過するまで それが反映されなくなります。

options

オプションの配列。もし WSDL モードで動作させる場合、 このパラメータはオプションです。非 WSDL モードで動作させる場合、 location と uri オプションを指定する必要があります。ここで、 location はリクエストを行う URL、 uri は SOAP サービスのターゲット名前空間です。

style および use オプション は非 WSDL モードでのみ動作します。 WSDL モードでは、これらは WSDL ファイルで指定されます。

soap_version オプションは、 SOAP 1.1 または SOAP 1.2 クライアントのどちらを使用するかを指定します。

HTTP 認証用として、login および password オプションが使用可能です。 プロキシサーバ経由で HTTP 接続を確立する場合は、 proxy_host, proxy_port, proxy_login および proxy_password の各オプションを使用してください。

compression オプションにより、 HTTP SOAP リクエストやレスポンスの圧縮を行うことができます。

encoding オプションは内部的な文字エンコーディングを定義します。 このオプションは SOAP リクエストのエンコーディング (常に utf-8) を変更しませんが、その中の文字列を変換します。

classmap オプションは WSDL 型を PHP クラスにマッピングするために使用可能です。 このオプションには、キーとしてWSDL 型、値として PHP クラスの名前を持つ配列を指定する必要があります。

boolean のオプション trace を設定すると、 SoapClient->__getLastRequest、 SoapClient->__getLastRequestHeaders、 SoapClient->__getLastResponse および SoapClient->__getLastResponseHeaders といったメソッドが使用できるようになります。

exceptions オプションは boolean 値で、soap のエラー時に SoapFault 型の例外をスローさせるかどうかを設定します。

connection_timeout オプションは、SOAP サービスに接続する際のタイムアウト秒数を指定します。 これを使用しても、レスポンスが遅いサービスのタイムアウトを定義することはできません。 サービスのコールが完了するまでの待ち時間を制限するには、設定項目 default_socket_timeout を使用します。

typemap オプションは、型マッピングの配列です。 この配列のキーは type_name、 type_ns (名前空間 URI)、from_xml (引数として文字列をひとつ受け取るコールバック) そして to_xml (引数としてオブジェクトをひとつ受け取るコールバック) です。

その他には stream_context や features、 cache_wsdl そして user_agent といったオプションがあります。

例

Example#1 SoapClient の例

```
<?php

$client = new SoapClient("some.wsdl");

$client = new SoapClient("some.wsdl", array('soap_version'    => SOAP_1_2));

$client = new SoapClient("some.wsdl", array('login'           => "some_name",
                                            'password'        => "some_password"));

$client = new SoapClient("some.wsdl", array('proxy_host'      => "localhost",
                                            'proxy_port'      => 8080));

$client = new SoapClient("some.wsdl", array('proxy_host'      => "localhost",
                                            'proxy_port'      => 8080,
                                            'proxy_login'     => "some_name",
                                            'proxy_password'  => "some_password"));

$client = new SoapClient("some.wsdl", array('local_cert'      => "cert_key.pem"));

$client = new SoapClient(null, array('location'  => "http://localhost/soap.php",
                                     'uri'       => "http://test-uri/"));

$client = new SoapClient(null, array('location'  => "http://localhost/soap.php",
                                     'uri'       => "http://test-uri/",
                                     'style'     => SOAP_DOCUMENT,
                                     'use'       => SOAP_LITERAL));

$client = new SoapClient("some.wsdl",
   array('compression'  => SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP));

$server = new SoapClient("some.wsdl", array('encoding'=>'ISO-8859-1'));

class MyBook {
     public $title;
     public $author;
}

$server = new SoapClient("books.wsdl", array('classmap' => array('book' => "MyBook")));

?>
```

SoapClient->__doRequest()

(PHP 5 >= 5.0.1)

SoapClient->__doRequest() — SOAP リクエストを実行する

説明

SoapClient
string **__doRequest** (string $request , string $location , string $action , int $version [, int $one_way])

HTTP 上で SOAP リクエストを実行します。

異なるトランスポート層や追加の XML を処理する、もしくは他の目的のために サブクラスでこのメソッドをオーバーライドする事ができます。

パラメータ

request

XML SOAP リクエスト

location

リクエスト先の URL

action

SOAP アクション

version

SOAP バージョン

one_way

返り値

XML SOAP レスポンス

変更履歴

バージョン	説明
5.1.3	パラメータ *one_way* が追加されました。

例

Example#1 いくつかの例

```
<?php
function Add($x,$y) {
   return $x+$y;
}

class LocalSoapClient extends SoapClient {

   function __construct($wsdl, $options) {
     parent::__construct($wsdl, $options);
     $this->server = new SoapServer($wsdl, $options);
     $this->server->addFunction('Add');
   }

   function __doRequest($request, $location, $action, $version) {
     ob_start();
     $this->server->handle($request);
     $response = ob_get_contents();
     ob_end_clean();
      return $response;
   }

}

$x = new LocalSoapClient(NULL,array('location'=>'test://',
                                    'uri'=>'http://testuri.org'));
var_dump($x->Add(3,4));
?>
```

SoapClient->__getFunctions()

(PHP 5 >= 5.0.1)

SoapClient->__getFunctions() — SOAP 関数の一覧を返す

説明

```
SoapClient
array __getFunctions ( void )
```

SOAP 関数の一覧を返します。

> **注意**: この関数は、WSDL モードでのみ動作します。

返り値

SOAP 関数の一覧

例

Example#1 SoapClient->__getFunctions() の例

```
<?php
$client = new SoapClient('some.wsdl');
var_dump($client->__getFunctions());
?>
```

参考

- SoapClient->__construct()

SoapClient->__getLastRequest()

(PHP 5 >= 5.0.1)

SoapClient->__getLastRequest() — 直近の SOAP リクエストを返す

説明

```
SoapClient
string __getLastRequest ( void )
```

> **注意**: このメソッドは、オプション *trace* を指定して *SoapClient* が作成されている場合のみ使用可能です。

返り値

直近の SOAP リクエスト

例

Example#1 SoapClient->__getLastRequest() の例

```
<?php
$client = SoapClient("some.wsdl", array('trace' => 1));
$result = $client->SomeFunction();
echo "REQUEST:¥n" . $client->__getLastRequest() . "¥n";
?>
```

参考

- SoapClient->__construct()
- SoapClient->__getLastRequestHeaders()
- SoapClient->__getLastResponse()
- SoapClient->__getLastResponseHeaders()

SoapClient->__getLastRequestHeaders()

(PHP 5 >= 5.0.1)

SoapClient->__getLastRequestHeaders() — 直近の SOAP リクエストヘッダを返す

説明

SoapClient
string **__getLastRequestHeaders** (void)

注意: このメソッドは、オプション trace を指定して SoapClient が作成されている場合のみ使用可能です。

返り値

直近の SOAP リクエストヘッダ

参考

- SoapClient->__construct()
- SoapClient->__getLastRequest()
- SoapClient->__getLastResponse()
- SoapClient->__getLastResponseHeaders()

SoapClient->__getLastResponse()

(PHP 5 >= 5.0.1)

SoapClient->__getLastResponse() — 直近の SOAP レスポンスを返す

説明

SoapClient
string **__getLastResponse** (void)

注意: このメソッドは、オプション trace を指定して SoapClient が作成されている場合のみ使用可能です。

返り値

直近の SOAP レスポンス

例

Example#1 SoapClient->__getLastResponse() の例

```
<?php
$client = SoapClient("some.wsdl", array('trace' => 1));
$result = $client->SomeFunction();
echo "RESPONSE:¥n" . $client->__getLastResponse() . "¥n";
?>
```

参考

- SoapClient->__construct()
- SoapClient->__getLastResponseHeaders()
- SoapClient->__getLastRequest()
- SoapClient->__getLastRequestHeaders()

SoapClient->__getLastResponseHeaders()

(PHP 5 >= 5.0.1)

SoapClient->__getLastResponseHeaders() — 直近の SOAP レスポンスヘッダを返す

説明

SoapClient
string **__getLastResponseHeaders** (void)

> **注意**: このメソッドは、オプション trace を指定して SoapClient が作成されている場合のみ使用可能です。

返り値

直近の SOAP レスポンスヘッダ

参考

- SoapClient->__construct()
- SoapClient->__getLastResponse()
- SoapClient->__getLastRequest()
- SoapClient->__getLastRequestHeaders()

SoapClient->__getTypes()

(PHP 5 >= 5.0.1)

SoapClient->__getTypes() — SOAP 型の一覧を返す

説明

SoapClient
array **__getTypes** (void)

Tこの関数は、WSDL モードでのみ動作します。

返り値

SOAP 型の一覧

例

Example#1 SoapClient->__getTypes() の例

```
<?php
$client = new SoapClient("some.wsdl");
var_dump($client->__getTypes());
?>
```

参考

- SoapClient->__construct()

SoapClient->__setCookie()

(PHP 5 >= 5.0.4)

SoapClient->__setCookie() — SOAP リクエストと共に送信されるクッキーを設定する

説明

SoapClient
void **__setCookie** (string $name [, string $value])

SOAP リクエストと共に送信されるクッキーを定義します。

> **注意**: このメソッドをコールすることで、その後の全ての SoapClient メソッドコール に影響します。

パラメータ

name

クッキーの名前

value

クッキーの値。指定されない場合、クッキーは削除されます。

返り値

値を返しません。

SoapClient->__soapCall()

(No version information available, might be only in CVS)

SoapClient->__soapCall() — SOAP 関数をコールする

説明

SoapClient
mixed **__soapCall** (string $function_name , array $arguments [, array $options [, mixed $input_headers [, array &$output_headers]]])

本メソッドは、SOAP コールを行う低レベル API 関数です。 通常、WSDL モードでは、SOAP 関数を SoapClient のメソッドとして簡単にコールすることができます。 本メソッドは、soapaction が不明な場合や、 uri がデフォルトと異なっていたり、 SOAP ヘッダを送受信したい場合に、非 WSDL モードを使用する際に有用です。

エラーの場合、SOAP 関数のコールは、PHP 例外または、例外が無効な場合に SoapFault オブジェクトが返されます。 この関数コールが失敗したかどうかを調べるには、 SoapFault 例外をキャッチするか、 is_soap_fault() 関数を指定して結果を調べてください。

返り値

SOAP 関数は、一つまたは複数の値を返す可能性があります。 SOAP 関数によって返される値が 1 つだけの場合、__soapCall の返す値は単純な値 (例えば、整数型、文字列など) になります。 複数の値が返される場合、__soapCall は出力パラメータの名前を連想配列として返します。

例

Example#1 SoapClient->__soapCall() の例

```
<?php

$client = new SoapClient("some.wsdl");
$client->SomeFunction($a, $b, $c);

$client->__soapCall("SomeFunction", array($a, $b, $c));
$client->__soapCall("SomeFunction", array($a, $b, $c), NULL,
                    new SoapHeader(), $output_headers);


$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$client->SomeFunction($a, $b, $c);
$client->__soapCall("SomeFunction", array($a, $b, $c));
$client->__soapCall("SomeFunction", array($a, $b, $c),
                     array('soapaction' => 'some_action',
                           'uri'        => 'some_uri'));
?>
```

参考

- SoapClient->__construct()
- SoapParam->__construct()
- SoapVar->__construct()
- SoapHeader->__construct()
- SoapFault->__construct()
- is_soap_fault()

SoapFault->__construct()

(No version information available, might be only in CVS)

SoapFault->__construct() — SoapFault コンストラクタ

説明

SoapFault
__construct (string $faultcode , string $faultstring [, string $faultactor [, mixed $detail [, string $faultname [, SoapHeader $headerfault]]]])

このクラスは、PHP ハンドラから SOAP フォールトレスポンスを送信した場合に有用です。 faultcode , faultstring , faultactor および details は SOAP フォールトの標準的な要素です。

パラメータ

faultcode

SoapFault のエラーコード

faultstring

SoapFault のエラーメッセージ

faultactor

エラーの原因となったアクターを識別する文字列

detail

faultname

WSDL からの厳密なフォールトエンコーディングを取得するために利用可能

headerfault

レスポンスヘッダにおいて SOAP ハンドラがエラーの報告処理を行っている間に利用可能 Can be used during SOAP header handling to report an error in the response header.

例

Example#1 いくつかの例

```
<?php
function test($x)
{
     return new SoapFault("Server", "Some error message");
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

SOAP フォールトを投げるために PHP の例外機構を使用することができます。

Example#2 いくつかの例

```
<?php
function test($x)
{
     throw new SoapFault("Server", "Some error message");
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

参考

- SoapClient->__construct()
- SoapClient->__soapCall()
- SoapVar->__construct()
- SoapParam->__construct()
- SoapFault->__construct()
- is_soap_fault()

SoapHeader->__construct()

(No version information available, might be only in CVS)

SoapHeader->__construct() — SoapHeader コンストラクタ

説明

SoapHeader
__construct (*string $namespace , string $name [, mixed $data [, bool $mustUnderstand [, mixed $actor]]]*)

新規 SoapHeader オブジェクトを生成します。

パラメータ

namespace

SOAP ヘッダ要素の名前空間

name

SOAP ヘッダ要素の名前

data

SOAP ヘッダの内容。PHP の値もしくは SoapVar オブジェクトです。

mustUnderstand

SOAP ヘッダ要素の mustUnderstand 属性の値

actor

SOAP ヘッダ要素の actor 属性の値

例

Example#1 いくつかの例

```
<?php
$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
```

```
                                        'uri'       => "http://test-uri/"));
$client->__soapCall("echoVoid", null, null,
                new SoapHeader('http://soapinterop.org/echoheader/',
                               'echoMeStringRequest',
                               'hello  world'));
?>
```

参考

- SoapClient->__soapCall()
- SoapVar->__construct()
- SoapParam->__construct()

SoapParam->__construct()

(No version information available, might be only in CVS)

SoapParam->__construct() — SoapParam コンストラクタ

説明

SoapParam
__construct (mixed $data , string $name)

新規 SoapParam オブジェクトを生成します。

パラメータ

data

渡すもしくは返すデータ。 PHP の値としてこのパラメータを直接渡すことができますが、この場合、 paramN という名前が付けられますので SOAP サービスがこれを理解しないかも知れません。

name

パラメータの名前

例

Example#1 いくつかの例

```
<?php
$client = new SoapClient(null,array('location'  => "http://localhost/soap.php",
                                    'uri'       => "http://test-uri/"));
$client->SomeFunction(new SoapParam($a, "a"),
                      new SoapParam($b, "b"),
                      new SoapParam($c, "c"));
?>
```

参考

- SoapClient->__soapCall()
- SoapVar->__construct()

SoapServer->addFunction()

(PHP 5 >= 5.0.1)

SoapServer->addFunction() — SOAP リクエストによって処理される単一もしくはいくつかの関数を追加する

説明

SoapServer
void **addFunction** (mixed $functions)

リモートクライアント用に単一もしくは複数の関数をエクスポートします。

パラメータ

functions

単一の関数をエクスポートするには、 このパラメータに文字列として関数名を渡してください。

いくつかの関数をエクスポートするには、関数名の配列を渡してください。

全ての関数をエクスポートする場合、特別な定数 **SOAP_FUNCTIONS_ALL** を渡してください。

> **注意**: functions は、全ての入力引数を WSDL ファイルで定義されている順序と同じ順序で受け取る必要があり（これらの関数は出力パラメータを引数として受け取ることはありません）、一つまたは複数の値を返す必要があります。 複数の値を返すには、名前付き出力パラメータの配列を返す必要があります。

返り値

値を返しません。

例

Example#1 いくつかの例

```
<?php

function echoString($inputString)
{
    return $inputString;
}

$server->addFunction("echoString");

function echoTwoStrings($inputString1, $inputString2)
{
    return array("outputString1" => $inputString1,
                 "outputString2" => $inputString2);
}
$server->addFunction(array("echoString", "echoTwoStrings"));

$server->addFunction(SOAP_FUNCTIONS_ALL);

?>
```

参考

- SoapServer->__construct()
- SoapServer->setClass()

SoapServer->__construct()

(No version information available, might be only in CVS)

SoapServer->__construct() — SoapServer コンストラクタ

説明

SoapServer
__construct (mixed $wsdl [, array $options])

このコンストラクタにより SoapServer オブジェクトを WSDL または非 WSDL モードで作成することが可能です。

パラメータ

wsdl

WSDL モードの場合、これに WSDL ファイルの URI を指定する必要があります。 その他の場合、**NULL** を指定し、uri オプションを設定する必要があります。

options

デフォルトの SOAP バージョン (soap_version)、内部の文字エンコーディング (encoding)、アクターの URI (actor) を指定することができます。

classmap オプションにより、WSDL 型を PHP のクラスにマッピングすることが可能です。 このオプションには、キーとしてWSDL 型、値として PHP クラスの名前を持つ配列を指定する必要があります。

typemap オプションは、型マッピングの配列です。 この配列のキーは type_name、 type_ns (名前空間 URI)、from_xml (引数として文字列をひとつ受け取るコールバック) そして to_xml (引数としてオブジェクトをひとつ受け取るコールバック) です。

その他には features や cache_wsdl といったオプションがあります。

例

Example#1 いくつかの例

```
<?php

$server = new SoapServer("some.wsdl");

$server = new SoapServer("some.wsdl", array('soap_version' => SOAP_1_2));

$server = new SoapServer("some.wsdl", array('actor' => "http://example.org/ts-tests/C"));

$server = new SoapServer("some.wsdl", array('encoding'=>'ISO-8859-1'));

$server = new SoapServer(null, array('uri' => "http://test-uri/"));

class MyBook {
    public $title;
    public $author;
}

$server = new SoapServer("books.wsdl", array('classmap' => array('book' => "MyBook")));

?>
```

SoapServer->fault()

(No version information available, might be only in CVS)

SoapServer->fault() — エラーを示す SoapServer フォールト を発行する

説明

```
SoapServer
void fault ( string $code , string $string [, string $actor [, mixed $details [, string $name ]]] )
```

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- SoapFault->__construct()

SoapServer->getFunctions()

(PHP 5 >= 5.0.1)

SoapServer->getFunctions() — 定義されている関数の一覧を返す

説明

```
SoapServer
array getFunctions ( void )
```

このメソッドは、 SoapServer->addFunction() もしくは SoapServer->setClass() で追加された全ての関数の一覧を返します。

返り値

全ての関数の一覧

例

Example#1 いくつかの例

```
<?php
$server = new SoapServer(NULL, array("uri" => "http://test-uri"));
$server->addFunction(SOAP_FUNCTIONS_ALL);
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $server->handle();
} else {
  echo "This SOAP server can handle following functions: ";
  $functions = $server->getFunctions();
  foreach($functions as $func) {
    echo $func . "¥n";
  }
}
?>
```

参考

- SoapServer->__construct()
- SoapServer->addFunction()
- SoapServer->setClass()

SoapServer->handle()

(PHP 5 >= 5.0.1)

SoapServer->handle() — SOAP リクエストを処理する

説明

```
SoapServer
void handle ([ string $soap_request ] )
```

SOAP リクエストを処理し、必要な関数をコールし、レスポンスを返送します。 back.

パラメータ

soap_request

SOAP リクエスト。もし引数が省略された場合、リクエストは $HTTP_RAW_POST_DATA PHP 変数の中にリクエストがあると仮定します。

返り値

値を返しません。

例

Example#1 いくつかの例

```php
<?php
function test($x)
{
     return $x;
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

参考

- SoapServer->__construct()

SoapServer->setClass()

(PHP 5 >= 5.0.1)

SoapServer->setClass() — SOAP リクエストを処理するクラスを設定する

説明

SoapServer
void **setClass** (string $class_name [, mixed $args [, mixed $...]])

指定されたクラスから全てのメソッドをエクスポートします。

このオブジェクトは SoapServer->setPersistence() メソッドにより指定した PHP セッションに関するリクエストをまたがる 持続性を持たせることができます

パラメータ

class_name

エクスポートするクラス名

args

これらのオプションパラメータは、 オブジェクト作成時にデフォルトのクラスコンストラクタに渡されます。

返り値

値を返しません。

例

Example#1 いくつかの例

```php
<?php

class foo {
     function foo()
     {
     }
}
$server->setClass("foo");

class bar {
     function bar($x, $y)
     {
     }
}
$server->setClass("bar", $arg1, $arg2);

?>
```

参考

- SoapServer->__construct()
- SoapServer->addFunction()
- SoapServer->setPersistence()

SoapServer->setPersistence()

(PHP 5 >= 5.1.2)

SoapServer->setPersistence() — SoapServer の持続モードを設定する

説明

SoapServer
void **setPersistence** (int $mode)

この関数により、ある PHP セッションにおいて複数のリクエスト間でデータを保持することが可能になります。 この関数は、サーバが

SoapServer->setClass() によりクラスから関数をエクスポートした場合のみ使用可能です。

パラメータ

mode

SOAP_PERSISTENCE_XXX 定数のうちの一つ

返り値

値を返しません。

例

Example#1 いくつかの例

```
<?php

$server->setPersistence(SOAP_PERSISTENCE_SESSION);

$server->setPersistence(SOAP_PERSISTENCE_REQUEST);

?>
```

> **注意**: 持続モード **SOAP_PERSISTENCE_SESSION** は、そのクラスのオブジェクトについてのみ持続させます。 クラスのスタティックなデータについては対象となりません。 self::$bar ではなく $this->bar を使用しましょう。

参考

- SoapServer->__construct()
- SoapServer->setClass()

SoapVar->__construct()

(No version information available, might be only in CVS)

SoapVar->__construct() — SoapVar コンストラクタ

説明

SoapVar
__construct (mixed $data , int $encoding [, string $type_name [, string $type_namespace [, string $node_name [, string $node_namespace]]]])

新規 SoapVar オブジェクトを生成します。

パラメータ

data

渡すもしくは返すデータ

encoding

エンコーディング ID。XSD_... 定数のうちの一つ。

type_name

型名

type_namespace

型の名前空間

node_name

XML ノードの名前

node_namespace

XML ノードの名前空間

例

Example#1 いくつかの例

```
<?php
class SOAPStruct {
     function SOAPStruct($s, $i, $f)
     {
         $this->varString = $s;
         $this->varInt = $i;
         $this->varFloat = $f;
     }
}
$client = new SoapClient(null, array('location' => "http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$struct = new SOAPStruct('arg', 34, 325.325);
$soapstruct = new SoapVar($struct, SOAP_ENC_OBJECT, "SOAPStruct", "http://soapinterop.org/xsd");
$client->echoStruct(new SoapParam($soapstruct, "inputStruct"));
```

```
?>
```

参考

- SoapClient->__soapCall()
- SoapParam->__construct()

---

use_soap_error_handler

(PHP 5)

use_soap_error_handler — SOAP エラーハンドラを使用して前の値を返すかどうかを設定する

説明

bool **use_soap_error_handler** ([bool $handler])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

ソケット関数

導入

ソケット拡張モジュールは、一般的な BSD ソケットに基づくソケット通信に関する低レベルインターフェースを実装し、 クライアントだけでなく、ソケットサーバとして動作させることが可能となります。

より一般的なクライアントサイドのソケットインターフェースについては、 stream_socket_client()、 stream_socket_server()、 fsockopen() および pfsockopen() を参照ください。

ここで説明するソケット関数を使用する場合、多くの関数は C 言語に同じ名前の関数が存在しますが、 しばしば定義が異なっていることに注意してください。 混乱を避けるには、説明をよく読んでください。

このようにソケットプログラミングと異なっている点はありますが、 それでも有用な多くの Unix man ページを参照することができます。 Web 上に C 言語のソケットプログラミングのチュートリアル情報が存在し、 その多くは、若干の修正により、 PHP におけるソケットプログラミングに適用することが可能です。 » Unix Socket FAQ が、 手始めとして適しているでしょう。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.3.0.

要件

外部ライブラリを必要としません。

インストール手順

ここに既述されたソケット関数は PHP 拡張モジュールの一部であり、 コンパイル時に **configure** にオプション --enable-sockets を指定することにより 使用可能となります。

> **注意**: IPv6 サポートは PHP 5.0.0 で追加されました。

実行時設定

設定ディレクティブは定義されていません。

リソース型

socket_accept()、 socket_create_listen() および socket_create() はソケットリソースを返します。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

AF_UNIX (integer)
AF_INET (integer)
AF_INET6 (integer)
SOCK_STREAM (integer)
SOCK_DGRAM (integer)
SOCK_RAW (integer)
SOCK_SEQPACKET (integer)
SOCK_RDM (integer)
MSG_OOB (integer)
MSG_WAITALL (integer)
MSG_PEEK (integer)
MSG_DONTROUTE (integer)
MSG_EOR (integer)
MSG_EOF (integer)
SO_DEBUG (integer)
SO_REUSEADDR (integer)
SO_KEEPALIVE (integer)
SO_DONTROUTE (integer)
SO_LINGER (integer)
SO_BROADCAST (integer)
SO_OOBINLINE (integer)
SO_SNDBUF (integer)
SO_RCVBUF (integer)
SO_SNDLOWAT (integer)
SO_RCVLOWAT (integer)
SO_SNDTIMEO (integer)
SO_RCVTIMEO (integer)
SO_TYPE (integer)
SO_ERROR (integer)
SOL_SOCKET (integer)
PHP_NORMAL_READ (integer)
PHP_BINARY_READ (integer)
SOL_TCP (integer)
SOL_UDP (integer)

ソケットのエラー

ソケット拡張モジュールは、 強力な BSD ソケットへの有用なインターフェースを提供するために作成されました。 関数は、Win32 および Unix の実装において等しく動作するように注意が払われています。 ソケット関数の多くは特定の条件で失敗し、エラーを記述する **E_WARNING** メッセージを出力します。 これは、時々開発者が望まない時に発生することがあります。例えば、 関数 socket_read() は突然 **E_WARNING** メッセージを出力する可能性があります。 これは、予測しない接続断が発生したためです。 @ 演算子により警告出力を抑制し、 socket_last_error() 関数によりアプリケーション内でエラーコードを取得することが一般に行われています。 エラーを記述する文字列を取得するためにこのエラーコードを指定して socket_strerror() 関数をコールすることが可能です。 詳細は、この関数の説明を参照してください。

> **注意**: **E_WARNING** メッセージは、 ソケット拡張モジュールにより英語で生成されますが、取得されるエラーメッセージは、 現在のロケール (**LC_MESSAGES**) に依存します。
>
> Warning - socket_bind() unable to bind address [98]: Die Adresse wird bereits verwendet

例

Example#1 ソケットの例: 簡易 TCP/IP サーバ

この例は、簡単な応答サーバです。変数 address と port を設定と実行環境に合うように変更してください。 このサーバに次のようなコマンドで接続することが可能です。 : **telnet 192.168.1.53 10000** (ただし、 アドレスとポートは設定に合わせます)入力したものは、サーバ側の出力となり、 エコーバックされます。接続を閉じるには、'quit' を入力します。

```
#!/usr/local/bin/php -q
<?php
error_reporting(E_ALL);

/* Allow the script to hang around waiting for connections. */
set_time_limit(0);

/* Turn on implicit output flushing so we see what we're getting
 * as it comes in. */
ob_implicit_flush();

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP)) === false) {
    echo "socket_create() failed: reason: " . socket_strerror(socket_last_error()) . "¥n";
```

```
}

if (socket_bind($sock, $address, $port) === false) {
    echo "socket_bind() failed: reason: " . socket_strerror(socket_last_error($sock)) . "¥n";
}

if (socket_listen($sock, 5) === false) {
    echo "socket_listen() failed: reason: " . socket_strerror(socket_last_error($sock)) . "¥n";
}

do {
    if (($msgsock = socket_accept($sock)) === false) {
        echo "socket_accept() failed: reason: " . socket_strerror(socket_last_error($sock)) . "¥n";
        break;
    }
    /* Send instructions. */
    $msg = "¥nWelcome to the PHP Test Server. ¥n" .
        "To quit, type 'quit'. To shut down the server type 'shutdown'.¥n";
    socket_write($msgsock, $msg, strlen($msg));

    do {
        if (false === ($buf = socket_read($msgsock, 2048, PHP_NORMAL_READ))) {
            echo "socket_read() failed: reason: " . socket_strerror(socket_last_error($msgsock)) . "¥n";
            break 2;
        }
        if (!$buf = trim($buf)) {
            continue;
        }
        if ($buf == 'quit') {
            break;
        }
        if ($buf == 'shutdown') {
            socket_close($msgsock);
            break 2;
        }
        $talkback = "PHP: You said '$buf'.¥n";
        socket_write($msgsock, $talkback, strlen($talkback));
        echo "$buf¥n";
    } while (true);
    socket_close($msgsock);
} while (true);

socket_close($sock);
?>
```

Example#2 ソケットの例: 簡易 TCP/IP クライアント

この例は、簡単な一回限りの HTTP クライアントです。ここでは、 あるページに接続して HEAD リクエストを送信し、応答を出力した後で終了します。

```
<?php
error_reporting(E_ALL);

echo "<h2>TCP/IP Connection</h2>¥n";

/* Get the port for the WWW service. */
$service_port = getservbyname('www', 'tcp');

/* Get the IP address for the target host. */
$address = gethostbyname('www.example.com');

/* Create a TCP/IP socket. */
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
if ($socket === false) {
    echo "socket_create() failed: reason: " . socket_strerror(socket_last_error()) . "¥n";
} else {
    echo "OK.¥n";
}

echo "Attempting to connect to '$address' on port '$service_port'...";
$result = socket_connect($socket, $address, $service_port);
if ($result === false) {
    echo "socket_connect() failed.¥nReason: ($result) " . socket_strerror(socket_last_error($socket)) . "¥n";
} else {
    echo "OK.¥n";
}

$in = "HEAD / HTTP/1.1¥r¥n";
$in .= "Host: www.example.com¥r¥n";
$in .= "Connection: Close¥r¥n¥r¥n";
$out = '';

echo "Sending HTTP HEAD request...";
socket_write($socket, $in, strlen($in));
echo "OK.¥n";

echo "Reading response:¥n¥n";
while ($out = socket_read($socket, 2048)) {
    echo $out;
}

echo "Closing socket...";
socket_close($socket);
echo "OK.¥n¥n";
?>
```

---

socket_accept

(PHP 4 >= 4.0.7, PHP 5)

socket_accept — ソケットへの接続を許可する

説明

resource **socket_accept** (resource $socket)

socket_create() を使用してソケット socket を作成した後、 socket_bind() で名前に関連付け、 socket_listen() で接続をモニタします。この関数は、 このソケットへの接続を許可します。接続に成功すると、新規の ソケット記述子が返されます。この記述子は通信の際に使用されます。ソケット上に複数の接続がキューで待っている場合、最初の接続が使用 されます。接続待ちがない場合、**socket_accept()** は接続が存在するまでブロックされます。 socket が socket_set_blocking() または socket_set_nonblock() により非ブロックモードで 作成された場合、**FALSE** が返されます。

socket_accept() により返されたソケットリソースは、 新規接続を許可するために使用することはできません。この場合でも 元の接続待ちのソケット socket は オープンされたままであり、再使用可能です。

パラメータ

socket

socket_create() で作成したソケットリソース。

返り値

成功した場合に新規ソケットリソースを、エラー時に **FALSE** を返します。 実際のエラーコードは、socket_last_error() を コールすることで取得可能です。このコードを socket_strerror() に渡すことで、 エラーの内容を文字列で取得することが可能です。

参考

- socket_connect()
- socket_listen()
- socket_create()
- socket_bind()
- socket_strerror()

---

socket_bind

(PHP 4 >= 4.0.7, PHP 5)

socket_bind — ソケットに名前をバインドする

説明

bool **socket_bind** (resource $socket , string $address [, int $port])

address で指定した名前を socket で指定したソケットにバインドします。 これは、socket_connect() あるいは socket_listen() を使用して接続が確立される前に行われます。

パラメータ

socket

socket_create() で作成した有効なソケット記述子。

address

ソケットの種類が **AF_INET** の場合、 address はドットで 4 つに区切られた表記 (例: 127.0.0.1) の IP アドレス。

ソケットの種類が **AF_UNIX** の場合、 address は Unix ドメインソケット (例: /tmp/my.sock)。

port (オプション)

パラメータ port は **AF_INET** ソケットに接続する場合にのみ使用され、 接続するリモートホストのポートを指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラーコードは socket_last_error() により取得できます。 このコードを socket_strerror() に渡すことにより、 エラー内容を表すテキストを得ることができます。

例

Example#1 socket_bind() を使用してソースアドレスを指定する

```
<?php
// 新しいソケットを作成する
$sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);

// コンピュータが所有する IP アドレスリストの例
$sourceips['kevin']    = '127.0.0.1';
$sourceips['madcoder'] = '127.0.0.2';

// ソースアドレスをバインドする
socket_bind($sock, $sourceips['madcoder']);

// 接続先アドレスと接続する
```

```
socket_connect($sock, '127.0.0.1', 80);

// 書き込む
$request = 'GET / HTTP/1.1' . "¥r¥n" .
           'Host: example.com' . "¥r¥n¥r¥n";
socket_write($sock, $request);

// 閉じる
socket_close($sock);

?>
```

注意

注意: この関数は、 socket_connect() の前に実行されている必要があります。

注意: Windows 9x/ME 互換性の注意: マシンに属しないアドレスにソケットをバインドしようとした場合、 socket_last_error() は無効なエラーコードを返すことがあります。

参考

- socket_connect()
- socket_listen()
- socket_create()
- socket_last_error()
- socket_strerror()

---

socket_clear_error

(PHP 4 >= 4.2.0, PHP 5)

socket_clear_error — ソケットのエラーまたは直近のエラーコードをクリアする

説明

void **socket_clear_error** ([resource $socket])

この関数は、指定したソケットまたは直近のグローバルなソケットエラー のエラーコードをクリアします。

この関数により、 ソケットまたは直近のグローバルな拡張エラーコードとなる エラーコードの値を明示的にリセットすることが可能になります。 これは、エラーが発生したかどうかをアプリケーション内で検出する際に有用です。

パラメータ

socket

socket_create() で作成したソケットリソース。

返り値

値を返しません。

参考

- socket_last_error()
- socket_strerror()

---

socket_close

(PHP 4 >= 4.0.7, PHP 5)

socket_close — ソケットリソースを閉じる

説明

void **socket_close** (resource $socket)

socket_close() は、 socket で指定したソケットリソースを閉じます。 この関数はソケット専用であり、その他のリソースに対しては用いることができません。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

返り値

値を返しません。

参考

- socket_bind()
- socket_listen()
- socket_create()
- socket_strerror()

---

socket_connect

(PHP 4 >= 4.0.7, PHP 5)

socket_connect — ソケット上の接続を初期化する

説明

bool **socket_connect** (resource $socket , string $address [, int $port])

ソケットリソース socket を用いて、 address への接続を初期化します。 このリソースは、socket_create() で作成した有効なソケットリソースである必要があります。

パラメータ

socket

address

パラメータ address には、 socket の種類が **AF_INET** の場合はドット区切り表記の IPv4 アドレス (例: 127.0.0.1)、 IPv6 サポートが有効で socket の種類が **AF_INET6** の場合は IPv6 アドレス (例: ::1)、 **AF_UNIX** の場合は Unix ドメインソケットのパス名を指定します。

port

パラメータ port は **AF_INET** ソケット あるいは **AF_INET6** に接続する場合にのみ必須となり、 接続するリモートホストのポートを指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーコードは、 socket_last_error() により取得できます。 このコードを socket_strerror() に渡すことにより、 エラー内容を表すテキストを得ることができます。

> **注意**: ソケットが非ブロッキングモードの場合、この関数は **FALSE** を返し、エラー Operation now in progress を発生させます。

参考

- socket_bind()
- socket_listen()
- socket_create()
- socket_last_error()
- socket_strerror()

---

socket_create_listen

(PHP 4 >= 4.0.7, PHP 5)

socket_create_listen — 接続を受けつけるためにポートにソケットをオープンする

説明

resource **socket_create_listen** (int $port [, int $backlog])

socket_create_listen() は、 **AF_INET** 型で すべての ローカルインターフェースの指定したポート上で新規接続を待ち受ける 新規ソケットリソースを作成します。

この関数は、新規接続のみを受け入れるソケットを作成しやすくするためのものです。

パラメータ

port

すべてのインターフェイスで待ち受けるポート。

backlog

backlog パラメータは、 接続の順番待ちのキューをどれだけの長さまで保持するかを定義します。 **SOMAXCONN** を backlog パラメータに渡します。詳細な情報は socket_listen() を参照ください。

返り値

socket_create_listen() は、 成功した場合に新規ソケットリソースを、エラー時に **FALSE** を返します。エラーコードは socket_last_error() で取得可能です。このコードを socket_strerror() に渡すと、エラーの詳細が文字列で取得可能です。

注意

> **注意**: 特定のインターフェースのみを listen するソケットを作成したい場合は socket_create()、 socket_bind() および socket_listen() を使用します。

参考

- socket_create()
- socket_create_pair()
- socket_bind()
- socket_listen()
- socket_last_error()
- socket_strerror()

---

socket_create_pair

(PHP 4 >= 4.0.7, PHP 5)

socket_create_pair — 区別できないソケットの組を作成し、配列に保存する

説明

bool **socket_create_pair** (int $domain , int $type , int $protocol , array &$fd)

socket_create_pair() は、接続されており区別できない 2 つのソケットを作成し、それを fd に保存します。 この関数は、一般に IPC (InterProcess Communication: プロセス間通信) で使用します。

パラメータ

domain

domain は、ソケットで使用するプロトコルの種類を指定します。 完全な一覧は socket_create() を参照ください。

type

type では、ソケットが使用する通信の形式を選択します。 完全な一覧は socket_create() を参照ください。

protocol

protocol は、指定した domain の中の特定のプロトコルを指定します。これは、返されるソケットとの通信に使用されます。 使用可能な値の名前は getprotobyname() で取得可能です。もし要求されるプロトコルが TCP あるいは UDP の場合、 対応する定数 **SOL_TCP** および **SOL_UDP** も使用可能です。

サポートするプロトコルの完全な一覧は socket_create() を参照ください。

fd

2 つのソケットリソースが格納される配列への参照。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 socket_create_pair() の例

```php
<?php
$sockets = array();
/* ソケットの組の設定 */
if (socket_create_pair(AF_UNIX, SOCK_STREAM, 0, $sockets) === false) {
    echo "socket_create_pair failed. Reason: ".socket_strerror(socket_last_error());
}
/* データの送受信 */
if (socket_write($sockets[0], "ABCdef123¥n", strlen("ABCdef123¥n")) === false) {
    echo "socket_write() failed. Reason: ".socket_strerror(socket_last_error($sockets[0]));
}
if (($data = socket_read($sockets[1], strlen("ABCdef123¥n"), PHP_BINARY_READ) === false) {
    echo "socket_read() failed. Reason: ".socket_strerror(socket_last_error($sockets[1]));
}
var_dump($data);

/* ソケットのクローズ */
socket_close($sockets[0]);
socket_close($sockets[1]);
?>
```

Example#2 socket_create_pair() での IPC の例

```php
<?php
$ary = array();
$strone = 'Message From Parent.';
$strtwo = 'Message From Child.';
if (socket_create_pair(AF_UNIX, SOCK_STREAM, 0, $ary) === false) {
    echo "socket_create_pair() failed. Reason: ".socket_strerror(socket_last_error());
}
$pid = pcntl_fork();
if ($pid == -1) {
    echo 'プロセスを fork できません。';
} elseif ($pid) {
    /* 親プロセス */
    socket_close($ary[0]);
    if (socket_write($ary[1], $strone, strlen($strone)) === false) {
        echo "socket_write() failed. Reason: ".socket_strerror(socket_last_error($ary[1]));
    }
```

```
    if (socket_read($ary[1], strlen($strtwo), PHP_BINARY_READ) == $strtwo) {
        echo "Recieved  $strtwo¥n";
    }
    socket_close($ary[1]);
} else {
    /* 子プロセス */
    socket_close($ary[1]);
    if (socket_write($ary[0], $strtwo, strlen($strtwo)) === false) {
        echo "socket_write()  failed.  Reason:  ".socket_strerror(socket_last_error($ary[0]));
    }
    if (socket_read($ary[0], strlen($strone), PHP_BINARY_READ) == $strone) {
        echo "Recieved  $strone¥n";
    }
    socket_close($ary[0]);
}
?>
```

参考

- socket_create()
- socket_create_listen()
- socket_bind()
- socket_listen()
- socket_last_error()
- socket_strerror()

---

socket_create

(PHP 4 >= 4.0.7, PHP 5)

socket_create — ソケット（通信時の終端）を作成する

説明

resource **socket_create** (int $domain , int $type , int $protocol)

通信のエンドポイント(終端)と呼ばれることもあるソケットのリソースを作成し、 返します。典型的なネットワーク接続は、2つのソケットから成り立ちます。 このとき、片方はクライアント、もう片方はサーバの役割をします。

パラメータ

domain

パラメータ domain には、 ソケットが利用するプロトコルファミリーを指定します。

指定可能なアドレス/プロトコルファミリーの一覧

ドメイン	説明
AF_INET	IPv4 インターネットプロトコル。 このプロトコルファミリーに属するプロトコルとしてよく知られているのは、 TCP や UDP です。
AF_INET6	IPv6 インターネットプロトコル。TCP と UDP が、このプロトコルファミリーで一般的なプロトコルです。このプロトコルのサポートは、 PHP 5.0.0 で追加されました。
AF_UNIX	ローカルでのコミュニケーションに用いられるプロトコルファミリーです。 高い効率と低いオーバーヘッドを誇るため、IPC（プロセス間通信）でよく使われます。

type

type パラメータは、 ソケットが利用する通信方式を指定します。

利用できるソケットのタイプ

タイプ	説明
SOCK_STREAM	このタイプでは、時系列的、高信頼性、全二重、接続型のバイトストリームが利用できます。 帯域外のデータ転送メカニズムがサポートされている場合もあります。 TCP プロトコルは、このソケットタイプに基づきます。
SOCK_DGRAM	このタイプでは、データグラム(非接続型で、信頼性の高くない 固定バイト長のメッセージ）がサポートされます。 UDP プロトコルは、このソケットタイプに基づきます。
SOCK_SEQPACKET	このタイプでは、時系列的な、信頼性のある、 双方向の接続指向型の固定長データグラム転送が利用できます。 パケットを消費する側は、一つのパケット全部を一度の read コールで読み込む必要があります。
SOCK_RAW	このタイプでは、素のネットワークプロトコルを操作できます。 この特殊なソケットを使って、どのタイプのプロトコルでもユーザの手で構築することができます。 よくある使い方として、(ping や traceroute などで行われているような) ICMP リクエストの作成があります。
SOCK_RDM	このタイプでは、信頼に足る、非時系列的なデータグラム転送が利用できます。 ほとんどのオペレーティングシステムでは実装されていないでしょう。

protocol

protocol は、ソケット上の通信で使われる domain で指定されたファミリーに属するプロトコルを指定します。 正しい値は、getprotobyname() を使うことで取得できます。利用したいプロトコルが、TCP または UDP の場合は、定数 **SOL_TCP** と **SOL_UDP** を指定することもできます。

一般的なプロトコルの一覧

名称	説明
icmp	Internet Control Message Protocol は、主にゲートウェイやホストが、 データグラム通信におけるエラーを報告するのに使われます。"ping" コマンド（最近のほとんどのオペーレーティングシステムに 搭載されています）が ICMP アプリケーションの一例です。
udp	User Datagram Protocol は、非接続指向の、信頼性の高くない、 固定のレコード長を用いるプロトコルです。このような側面のおかげで、 UDP はプロトコルとして最小限のオーバーヘッドしか要求しません。

名称	説明
tcp	Transmission Control Protocol は、信頼性の高い、接続指向かつ ストリーム指向の全二重通信プロトコルです。TCP は、 すべてのパケットが、送信された順序で(時系列的に)受信されることを 保証します。もし、何らかの理由でパケットが通信中に失われた場合、 TCP では、送信先から通知があるまで、パケットが再送信されるように なっています。信頼性とパフォーマンス上の理由から、TCP の実装は、 下層にあるデータグラム通信レイヤーのオクテット幅を 適当な長さに決定します。このため、TCP アプリケーションは、 レコードの全部が一度に転送されない場合も考慮しなければなりません。

返り値

socket_create() は、 成功時にソケットリソース、失敗時に **FALSE** を返します。 実際のエラーコードは、socket_last_error() を コールすることにより取得できます。このエラーコードをさらに socket_strerror()に渡すことにより、 エラーの内容を文字列で取得することが可能です。

エラー / 例外

domain や *type* に 不正な値が与えられた場合、**socket_create()** は、これらを それぞれ **AF_INET** と **SOCK_STREAM** であるとみなし、**E_WARNING** メッセージを出します。

参考

- socket_accept()
- socket_bind()
- socket_connect()
- socket_listen()
- socket_last_error()
- socket_strerror()

---

socket_get_option

(PHP 4 >= 4.3.0, PHP 5)

socket_get_option — ソケットのオプションを取得する

説明

mixed **socket_get_option** (*resource $socket , int $level , int $optname*)

socket_get_option() 関数は、ソケット *socket* のオプション *optname* の値を取得します。**socket_get_option()** は、失敗した場合に **FALSE** を返します。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

level

level パラメータは、オプションのプロトコルレベルを指定します。 例えば、オプションをソケットレベルで取得するには *level* パラメータに *SOL_SOCKET* を指定します。 *TCP* のようなそれ以外のレベルの場合、そのレベルのプロトコル番号を指定します。 プロトコル番号は getprotobyname() 関数を使用して取得可能です。

optname

使用可能なソケットオプション

オプション	説明	型
SO_DEBUG	デバッグ情報を記録するかどうかを報告します。	int
SO_BROADCAST	ブロードキャストメッセージの送信がサポートされているかどうかを報告します。	int
SO_REUSEADDR	ローカルアドレスが再使用可能するかどうかを報告します。	int
SO_KEEPALIVE	定期的なメッセージの送信によって接続がアクティブになっているかどうかを報告します。 もしソケットがこれらのメッセージに応答できなかった場合、 接続は崩壊し、ソケットへの書き込みを行うと SIGPIPE シグナルを受け取ります。	int
SO_LINGER	データがまだ残っているうちに socket_close() をコールした場合に、socket を残存させるかどうかを報告します。 デフォルトでは、ソケットが閉じられる際には未送信のデータをすべて送信しようとします。 接続ベースのソケットでは、socket_close() は接続先がデータを認識するまで閉じるのを待ちます。 l_onoff が非ゼロで l_linger がゼロの場合は、 その時点で未送信のデータはすべて破棄されます。 接続ベースのソケットの場合、接続先には RST (リセット) を送信します。 一方 l_onoff が非ゼロで l_linger も非ゼロの場合は、 socket_close() は全データを送信し終えるか l_linger で指定した時間が経過するまで処理をブロックします。 ソケットが非ブロックモードの場合は、 socket_close() は失敗してエラーを返します。	array。配列に含まれるキーは l_onoff および l_linger のふたつ。
SO_OOBINLINE	socket が帯域外のデータをインラインに残し続けるかを報告します。	int
SO_SNDBUF	送信バッファのサイズを報告します。	int
SO_RCVBUF	受信バッファのサイズを報告します。	int
SO_ERROR	エラーステータスに関する情報を報告し、それをクリアします。	int (socket_set_option() で設定することはできません)
SO_TYPE	socket の型 (たとえば SOCK_STREAM) を報告します。	int (socket_set_option() で設定することはできません)
SO_DONTROUTE	送信メッセージがルータを越えるかどうかを報告します。	int
SO_RCVLOWAT	socket の入力操作を行う際の最小バイト数を報告します。	int
SO_RCVTIMEO	入力操作のタイムアウト値を報告します。	array。配列に含まれるキーはふたつ。 sec はタイムアウトの秒単位の部分で、 usec はタイムアウトのミリ秒単位の部分。
SO_SNDTIMEO	出力関数がフロー制御のためにブロックするタイムアウト値を報告します。	array。配列に含まれるキーはふたつ。 sec はタイムアウトの秒単位の部分で、 usec はタイムアウトのミリ秒単位の部分。
SO_SNDLOWAT	socket の出力操作を行う際の最小バイト数を報告します。	int

返り値

*指定したオプションの値、あるいはエラー時に **FALSE** を返します。*

例

Example#1 socket_set_option() の例

```
<?php
$socket = socket_create_listen(1223);

$linger = array('l_linger' => 1, 'l_onoff' => 1);
socket_set_option($socket, SOL_SOCKET, SO_LINGER, $linger);

var_dump(socket_get_option($socket, SOL_SOCKET, SO_REUSEADDR));
?>
```

変更履歴

バージョン	説明
4.3.0	関数の名前が変わりました。以前は *socket_getopt()* という名前でした。

---

socket_getpeername

(PHP 4 >= 4.0.7, PHP 5)

socket_getpeername — 指定したソケットのリモート側に問い合わせ、その型に応じてホスト/ポート、あるいは Unix ファイルシステムのパスを返す

説明

*bool **socket_getpeername** (resource $socket , string &$address [, int &$port])*

指定したソケットのリモート側に問い合わせ、 その型に応じてホスト/ポート、あるいは Unix ファイルシステムのパスを返します。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

address

*指定されたソケットの型が **AF_INET** あるいは **AF_INET6** であった場合、**socket_getpeername()** はピア (リモート) の IP アドレス を適切な書式 (例: 127.0.0.1 あるいは fe80::1) に したものを address パラメータに、そしてもし オプションの port パラメータが指定されていれば*

そこにポートを格納します。

指定されたソケットの型が **AF_UNIX** であった場合、 **socket_getpeername()** は Unix ファイルシステムの パス（例: /var/run/daemon.sock）を address パラメータに格納します。

port

指定した場合は、 address に関連付けるポートを保持します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 **socket_getpeername()** は、 ソケットの型が **AF_INET**、 **AF_INET6** あるいは **AF_UNIX** のいずれでもない場合にも **FALSE** を返します。この場合には、 直近のソケットエラーコードは更新されません。

注意

> **注意**: **socket_getpeername()** は、 socket_accept() で作成した **AF_UNIX** ソケットとともに使用することはできません。意味のある値が返されるのは、 socket_connect() で作成したソケットか socket_bind() に続くプライマリサーバソケットのみです。

参考

- socket_getsockname()
- socket_last_error()
- socket_strerror()

---

socket_getsockname

(PHP 4 >= 4.0.7, PHP 5)

socket_getsockname — 指定したソケットのローカル側に問い合わせ、その型に応じてホスト/ポート、あるいは Unix ファイルシステムのパスを返す

説明

bool **socket_getsockname** (resource $socket , string &$addr [, int &$port])

> **注意**: **socket_getsockname()** は、 socket_accept() で作成した **AF_UNIX** ソケットとともに使用することはできません。意味のある値が返されるのは、 socket_accept() で作成したソケットか socket_bind() に続くプライマリサーバソケットのみです。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

addr

指定されたソケットの型が **AF_INET** あるいは **AF_INET6** であった場合、**socket_getsockname()** はローカルの IP アドレス を適切な書式（例: 127.0.0.1 あるいは fe80::1）に したものを address パラメータに、そしてもし オプションの port パラメータが指定されていれば そこにポートを格納します。

指定されたソケットの型が **AF_UNIX** であった場合、 socket_getpeername() は Unix ファイルシステムのパス （例: /var/run/daemon.sock）を address パラメータに格納します。

port

指定した場合は、関連付けるポートを保持します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 **socket_getsockname()** は、 ソケットの型が **AF_INET**、 **AF_INET6** あるいは **AF_UNIX** のいずれでもない場合にも **FALSE** を返します。この場合には、 直近のソケットエラーコードは更新されません。

参考

- socket_getpeername()
- socket_last_error()
- socket_strerror()

---

socket_last_error

(PHP 4 >= 4.0.7, PHP 5)

socket_last_error — ソケットの直近のエラーを返す

説明

int **socket_last_error** ([resource $socket])

ソケットリソースがこの関数に渡された場合、この特定のソケットに発生した 直近のエラーが返されます。ソケットリソースが省略された場合、 直近にエラーを発生したソケット関数のエラーコードが返されます。 後者は、失敗した場合にソケットを返さない socket_create() のような関数や 特定のソケットに直接関係ない理由で失敗する可能性がある socket_select() で特に有用です。 このエラーコードは、指定したエラーコードを表す文字列を得るために

パラメータ

socket

socket_create() で作成したソケットリソース。

返り値

この関数は、ソケットのエラーコードを返します。

例

Example#1 socket_last_error() の例

```
<?php
$socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP);

if ($socket === false) {
    $errorcode = socket_last_error();
    $errormsg = socket_strerror($errorcode);

    die("ソケットを作成できません: [$errorcode] $errormsg");
}
?>
```

注意

> **注意**: **socket_last_error()** はエラーコードをクリアしません。 クリアするには socket_clear_error() を使用してください。

---

socket_listen

(PHP 4 >= 4.0.7, PHP 5)

socket_listen — ソケット上で接続待ち(listen)する

説明

bool **socket_listen** (resource $socket [, int $backlog])

ソケット socket が socket_create() を用いて作成され、 socket_bind() で名前が付けられた後、 socket 上の接続要求を待つための通信ができるようになります。

socket_listen() は、ソケットが SOCK_STREAM 型または SOCK_SEQPACKET 型の場合のみ利用可能です。

パラメータ

socket

socket_create() で作成したソケットリソース。

backlog

最大 backlog 個の接続を処理用の キューで待ち受けることが可能です。もし待ちうけ用のキューが いっぱいになった場合、クライアントでは ECONNREFUSED の通知とともにエラーが発生します。あるいは、もし基盤となるプロトコルが リクエストの再送をサポートしている場合、再試行が成功するまで リクエストは無視されます。

> **注意**: backlog パラメータに指定できる値の最大値は プラットフォームに大きく依存します。Linux では、最大値は **SOMAXCONN** に切り詰められます。win32 では、 もし **SOMAXCONN** を渡した場合、backlog の 最大値を適切な値に設定する責任はサービスの 提供側が負います。 このプラットフォームでは、実際の backlog の値を見つける標準的な 手段が提供されていません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 エラーコードは socket_last_error() で取得可能で、このコードを socket_strerror() に指定することにより エラーの内容を文字列として取得可能です。

参考

- socket_accept()
- socket_bind()
- socket_connect()
- socket_create()
- socket_strerror()

---

socket_read

(PHP 4 >= 4.0.7, PHP 5)

socket_read — ソケットから最大バイト長まで読みこむ

説明

string **socket_read** (resource $socket , int $length [, int $type])

関数 **socket_read()** は、関数 socket_create() または socket_accept() により作成されたソケット リソース socket から読み込みます。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

length

読み込まれる最大バイト長は、length パラメータで 指定します。 読み込みを終了するために \r、\n、\0 を使用することが可能です（これは、以下に示す type に依存します）。

type

オプションのパラメータ type は、名前のある定数です。

- **PHP_BINARY_READ**（デフォルト） - システムの recv() 関数を 使用します。バイナリデータ読み込みに関して安全です。
- **PHP_NORMAL_READ** - 読み込みは、\n あるいは \r で中断されます。

返り値

socket_read() は、成功時に文字列としてデータを返し、 エラー時（リモートホストが接続をクローズした場合を含みます）に **FALSE** を返します。 エラーコードは socket_last_error() で取得可能であり、 このコードは、エラー文字列を取得するために socket_strerror() に渡すことができます。

注意: **socket_read()** は、読み込むデータがもう存在しない 場合に長さゼロの文字列("")を返します。

変更履歴

バージョン	説明
4.1.0	type のデフォルト値が **PHP_NORMAL_READ** から **PHP_BINARY_READ** に変わりました。

参考

- socket_accept()
- socket_bind()
- socket_connect()
- socket_listen()
- socket_last_error()
- socket_strerror()
- socket_write()

---

socket_recv

(PHP 4 >= 4.0.7, PHP 5)

socket_recv — 接続したソケットからデータを受信する

説明

int **socket_recv** (resource $socket , string &$buf , int $len , int $flags)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

socket_recvfrom

(PHP 4 >= 4.0.7, PHP 5)

socket_recvfrom — 接続しているかどうかによらず、ソケットからデータを受信する

説明

int **socket_recvfrom** (resource $socket , string &$buf , int $len , int $flags , string &$name [, int &$port])

socket_recvfrom() 関数は、 ポート port（**AF_UNIX** 型のソケットである場合を除く）上の name から受信した len バイトのデータを buf に格納します。 **socket_recvfrom()** は、 接続済みのソケットだけでなく接続していないソケットに対しても使用可能です。 さらに、フラグを指定することでこの関数の挙動を設定できます。

name と port は参照渡しとしなければなりません。接続していないソケットの場合は、 name はリモートホストの IP アドレスか UNIX ソケットへのパスとなります。接続済みのソケットの場合は、 name は **NULL** とします。 また、**AF_INET** あるいは **AF_INET6** 形式のまだ接続していないソケットの場合、 port にはリモートホストのポート番号を指定します。

パラメータ

socket

socket には、 socket_create() で作成したソケットリソースを指定します。

buf

受信したデータが buf に格納されます。

len

最大 len バイトまでのデータをリモートホストから取得します。

flags

flags の値は、以下のフラグの任意の組み合わせを 論理 OR 演算子 (|) で連結したものとなります。

flags に使用できる値

フラグ	説明
MSG_OOB	帯域外 (out-of-band) のデータを処理する。
MSG_PEEK	受信キューの先頭にあるデータを受信し、 そのデータをそのままキューに残しておく。
MSG_WAITALL	少なくとも len バイト受信するまではブロックする。 しかし、もし何らかのシグナルを受信したりリモートホストとの接続が切断された場合は これより少ないバイト数を返す可能性がある。
MSG_DONTWAIT	通常はブロックする場面であってもそのまま return する。

name

AF_UNIX 型のソケットの場合は、 name はファイルへのパスとなります。 それ以外の場合は、未接続のソケットの場合には name はリモートホストの IP アドレスとなります。 接続済みソケットの場合は NULL となります。

port

この引数は AF_INET 型あるいは AF_INET6 型のソケットに対してのみ適用され、 データを受信するリモートホストのポートを指定します。 接続済みソケットの場合は port は NULL となります。

返り値

socket_recvfrom() は、受信したバイト数を返します。 あるいはエラー時には -1 を返します。 エラーコードを取得するには socket_last_error() をコールします。取得したエラーコードを socket_strerror() に渡すと、 そのエラーについての説明を得ることができます。

例

Example#1 socket_recvfrom() の例

```
<?php
error_reporting(E_ALL | E_STRICT);

$socket = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
socket_bind($socket, '127.0.0.1', 1223);

$from = "";
$port = 0;
socket_recvfrom($socket, $buf, 12, 0, $from, $port);

echo "リモートアドレス $from のポート $port から $buf を受信しました" . PHP_EOL;
?>
```

この例は、127.0.0.1 のポート 1223 との UDP ソケットを確立し、受信したデータを最大 12 バイトまで表示します。

変更履歴

バージョン	説明
4.3.0	socket_recvfrom() はバイナリセーフとなりました。

参考

- socket_recv()
- socket_send()
- socket_sendto()
- socket_create()

---

socket_select

(PHP 4 >= 4.0.7, PHP 5)

socket_select — 与えられたソケットの配列に対し、指定した有効時間で select() システムコールを実行する

説明

*int **socket_select** (array &$read , array &$write , array &$except , int $tv_sec [, int $tv_usec])*

socket_select() はソケットの配列を受け取り、 それらの状態が変化するまで待ちます。BSD のソケットについての知識がある方なら、 これらのソケットの配列が、いわゆるファイル記述子セットであることがご理解いただけるでしょう。 3 つの独立した配列でソケットリソースを監視します。

パラメータ

read

配列 read に挙げられたソケットでは、 文字が読み込み可能になっているかどうか（厳密に言うと、読み込みが ブロックされていないかどうか - 実際には、ソケット記述子はファイルの 終端でも有効です。そのような場合、socket_read() は長さゼロの文字列を返します）を監視します。

write

配列 write に挙げられたソケットでは、 書き込みがブロックされていないかどうかを監視します。

except

配列 except に挙げられたソケットでは、 例外を監視します。

tv_sec

*tv_sec および tv_usec は、ともにタイムアウトを指定するパラメータです。 タイムアウトは、**socket_select()** が結果を返すまでの経過時間の最大値です。 tv_sec はゼロにすることも可能で、そうすると **socket_select()** は結果をすぐに返します。 これはポーリングをする際に有用です。tv_sec に **NULL**（タイムアウトしない）を指定すると、 **socket_select()** は無期限にブロックします。*

tv_usec

警告

終了時に配列は書き換えられ、 どのソケットの状態が変わったのかがわかるようになります。

***socket_select()** のすべての配列を設定する必要はありません。 使用しないものについては空の配列や **NULL** をかわりに指定しておくことが可能です。 また、これらの配列は参照渡し であり、 **socket_select()** をコールした後でその中身が書き換えられていることに注意しましょう。*

> ***注意**: 現状の Zend Engine の制限により、関数の参照渡しパラメータに **NULL** のような定数値を直接渡すことができません。一時的な変数を使用するか、 あるいは一番左に一時変数を使用する式を使用してください。*
>
> ***Example#1 socket_select() での NULL の使用***
>
> ```
> <?php
> $e = NULL;
> socket_select($r, $w, $e, 0);
> ?>
> ```

返り値

*成功した場合は、**socket_select()** は配列内で 変化のあったソケットリソースの数を返します。もし何かがおこる前に タイムアウト時間が経過した場合は、ゼロを返すことになります。 エラー時には **FALSE** が返されます。エラーコードは socket_last_error() で取得可能です。*

> ***注意**: エラーかどうかを調べる際には、必ず === 演算子を 使用するようにしましょう。**socket_select()** は 0 を返す場合もあり、このような場合に == を用いて比較すると、エラーと判定されてしまいます。*
>
> ***Example#2 socket_select() の返す結果を知る***
>
> ```
> <?php
> $e = NULL;
> if (false === socket_select($r, $w, $e, 0)) {
>     echo "socket_select() は失敗しました。原因: " .
>         socket_strerror(socket_last_error()) . "¥n";
> }
> ?>
> ```

例

Example#3 socket_select() の例

```
<?php
/* 読み込み用の配列を準備する */
$read   = array($socket1, $socket2);
$write  = NULL;
$except = NULL;
$num_changed_sockets  = socket_select($read, $write, $except, 0);

if ($num_changed_sockets  === false) {
    /* エラー処理 */
} else if ($num_changed_sockets  > 0) {
    /* すくなくともひとつのソケットで、何らかの出来事が起こっています */
}
?>
```

注意

> ***注意**: ソケットの実装によっては、取り扱いに注意すべきものがあることを知っておいてください。 基本的なルールは以下のとおりです。*
>
> - *基本的に **socket_select()** のタイムアウトは 指定しないように心がけましょう。もしデータがなかった場合に、 プログラム側でそれを判定できなくなってしまいます。タイムアウトに 依存しているコードは移植性が悪く、デバッグが困難です。*
> - ***socket_select()** のコール後に値をチェックして 適切に処理するつもりがないソケットリソースは、決して配列に追加してはいけません。 **socket_select()** から値が返ってきたあとは、 配列内のすべてのソケットリソースをチェックする必要があります。すべての書き込み用ソケットは書き込める必要がありますし、 またすべての読み込み用ソケットは読み込める必要があります。*
> - *配列で返されたソケットに対して読み込み/書き込みをする場合には、 指定したデータを必ずしもすべて読み込み/書き込みするとは限らないことを 知っておいてください。たった 1 バイトしか読み込み/書き込みが できなかった場合にも対処できるよう準備しておきましょう。*
> - *ほとんどのソケット実装で、except でキャッチできる 例外はただひとつ、すなわちソケットが受け取ったデータが帯域外で あったということだけです。*

参考

- socket_read()
- socket_write()
- socket_last_error()

- socket_strerror()

---

socket_send

(PHP 4 >= 4.0.7, PHP 5)

socket_send — 接続したソケットにデータを送信する

説明

int **socket_send** (resource $socket , string $buf , int $len , int $flags)

関数 **socket_send()** は、 buf からソケット socket に len バイトのデータを送信します。

パラメータ

socket

socket_create() あるいは socket_accept() が作成したソケットリソース。

buf

リモートホストに送信するデータを含むバッファ。

len

buf からリモートホストに送信するバイト数。

flags

flags の値は、以下のフラグの任意の組み合わせを 論理 OR 演算子 (|) で連結したものとなります。

flags がとりうる値

MSG_OOB	OOB (out-of-band: 帯域外) データを送信します。
MSG_EOR	レコードにマークをつけます。 レコードでデータがそろいます。
MSG_EOF	ソケットの送信側を閉じ、そのことを知らせる通知を送信データの最後に付加します。 トランザクションでデータがそろいます。
MSG_DONTROUTE	ルータを使用せず、直接つながっているインターフェースのみを使用します。

返り値

参考

- socket_sendto()

---

socket_sendto

(PHP 4 >= 4.0.7, PHP 5)

socket_sendto — 接続しているかどうかによらずソケットにメッセージを送信する

説明

int **socket_sendto** (resource $socket , string $buf , int $len , int $flags , string $addr [, int $port])

関数 **socket_sendto()** は、 アドレス addr の port を使用し、 buf からソケット socket に len バイトのデータを送信します。

パラメータ

socket

socket_create() で作成したソケットリソース。

buf

送信するデータが、buf から取り出されます。

len

buf から len バイト分のデータが送信されます。

flags

flags の値は、以下のフラグの任意の組み合わせを 論理 OR 演算子 (|) で連結したものとなります。

flags がとりうる値

MSG_OOB	OOB (out-of-band: 帯域外) データを送信します。
MSG_EOR	レコードにマークをつけます。 レコードでデータがそろいます。
MSG_EOF	ソケットの送信側を閉じ、そのことを知らせる通知を送信データの最後に付加します。 トランザクションでデータがそろいます。
MSG_DONTROUTE	ルータを使用せず、直接つながっているインターフェースのみを使用します。

addr

リモートホストの IP アドレス。

port

port は、 データの送信先となるリモートホストのポート番号です。

返り値

socket_sendto() は、リモートホストに送信したバイト数を返します。 エラーが発生した場合は *-1* を返します。

例

Example#1 socket_sendto() の例

```
<?php
     $sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);

    $msg = "Ping !";
    $len = strlen($msg);

    socket_sendto($sock, $msg, $len, 0, '127.0.0.1', 1223);
    socket_close($sock);
?>
```

参考

- socket_send()

---

socket_set_block

(PHP 4 >= 4.2.0, PHP 5)

socket_set_block — ソケットリソースをブロックモードに設定する

説明

bool **socket_set_block** (*resource $socket*)

socket_set_block() 関数は、 *socket* パラメータで指定したソケットから **O_NONBLOCK** フラグを取り除きます。

受信や送信、接続、待機といった操作をブロックモードのソケットに対して行うと、 その処理が完了するか何らかのシグナルを受信するまではスクリプトを停止します。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 socket_set_block() の例

```
<?php
$socket = socket_create_listen(1223);
socket_set_block($socket);

socket_accept($socket);
?>
```

この例は、すべてのインターフェイス上でポート *1223* を待ち受けるソケットを作成し、それを **O_BLOCK** モードに設定します。 socket_accept() は、 接続が受け付けられるまで処理を停止して待ち続けます。

参考

- socket_set_nonblock()
- socket_set_option()

---

socket_set_nonblock

(PHP 4 >= 4.0.7, PHP 5)

socket_set_nonblock — ソケットリソースを非ブロックモードに設定する

説明

bool **socket_set_nonblock** (*resource $socket*)

socket_set_nonblock() 関数は、 *socket* パラメータで指定したソケットに **O_NONBLOCK** フラグを設定します。

受信や送信、接続、待機といった操作を非ブロックモードのソケットに対して行うと、 その処理が完了するか何らかのシグナルを受信するまではスクリプトは停止しません。 また、その操作がブロックされると、呼び出し元の関数は失敗します。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 socket_set_nonblock() の例

```
<?php
$socket = socket_create_listen(1223);
socket_set_nonblock($socket);

socket_accept($socket);
?>
```

この例は、すべてのインターフェイス上でポート 1223 を待ち受けるソケットを作成し、それを **O_NONBLOCK** モードに設定します。
socket_accept() は、 その時点で待機中の接続がない場合はすぐに失敗します。

参考

- socket_set_block()
- socket_set_option()

---

socket_set_option

(PHP 4 >= 4.3.0, PHP 5)

socket_set_option — ソケットのオプションを設定する

説明

bool **socket_set_option** (resource $socket , int $level , int $optname , mixed $optval)

socket_set_option() 関数は、level が指すプロトコルレベルでソケット socket のオプション optname に値 optval を設定します。

パラメータ

socket

socket_create() あるいは socket_accept() で作成したソケットリソース。

level

level パラメータは、オプションのプロトコルレベルを指定します。 例えば、オプションをソケットレベルで取得するには level パラメータに SOL_SOCKET を指定します。 TCP のようなそれ以外のレベルの場合、そのレベルのプロトコル番号を指定します。 プロトコル番号は getprotobyname() 関数を使用して取得可能です。

optname

使用可能なソケットオプションは socket_get_option() 関数と同じです。

optval

オプションの値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 socket_set_option() の例

```
<?php
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);

if (!is_resource($socket)) {
    echo 'ソケットを作成できません:  '. socket_strerror(socket_last_error()) . PHP_EOL;
}

if (!socket_set_option($socket, SOL_SOCKET, SO_REUSEADDR, 1)) {
    echo 'ソケットのオプションを設定できません:  '. socket_strerror(socket_last_error()) . PHP_EOL;
}

if (!socket_bind($socket, '127.0.0.1', 1223)) {
    echo 'ソケットをバインドできません:  '. socket_strerror(socket_last_error()) . PHP_EOL;
}

$rval = socket_get_option($socket, SOL_SOCKET, SO_REUSEADDR);

if ($rval === false) {
    echo 'ソケットのオプションを取得できません:  '. socket_strerror(socket_last_error()) . PHP_EOL;
} else if ($rval !== 0) {
    echo 'SO_REUSEADDR がソケットに設定されています!' . PHP_EOL;
```

```
}
?>
```

変更履歴

バージョン	説明
4.3.0	関数の名前が変わりました。以前は socket_setopt() という名前でした。

---

socket_shutdown

(PHP 4 >= 4.0.7, PHP 5)

socket_shutdown — 受信、送信、または送受信用のソケットをシャットダウンする

説明

bool **socket_shutdown** (resource $socket [, int $how])

socket_shutdown() 関数は、socket から送られてくる受信、送信あるいはすべて（デフォルト）のデータを停止します。

パラメータ

socket

socket_create() で作成したソケットリソース。

how

how の値は以下のうちのひとつです。

how のとりうる値

0	ソケットの読み込みを停止します。
1	ソケットの書き込みを停止します。
2	ソケットの読み込み・書き込みを停止します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

socket_strerror

(PHP 4 >= 4.0.7, PHP 5)

socket_strerror — ソケットエラーの内容を文字列として返す

説明

string **socket_strerror** (int $errno)

socket_strerror() は、パラメータ errno に socket_last_error() の返すソケットエラーコードを受け取り、 対応する内容を文字列で返します。

> **注意**: socket 拡張モジュールが出すエラーメッセージは英語ですが、 この関数から取得したシステムメッセージは現在のロケール (**LC_MESSAGES**) にあわせた表示となります。

パラメータ

errno

ソケットのエラー番号。たいていは socket_last_error() が返したものとなるでしょう。

返り値

errno パラメータに対応するエラーメッセージを返します。

例

Example#1 socket_strerror() の例

```
<?php
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP))) {
    echo "socket_create() failed: reason: " . socket_strerror(socket_last_error()) . "¥n";
}

if (false == (@socket_bind($socket, '127.0.0.1', 80))) {
    echo "socket_bind() failed: reason: " . socket_strerror(socket_last_error($socket)) . "¥n";
}
?>
```

上記の例の出力はおそらく次のようになります（このスクリプトがルート権限で実行されていないことを仮定します）。

```
socket_bind() failed: reason: Permission denied
```

参考

- socket_accept()
- socket_bind()
- socket_connect()
- socket_listen()
- socket_create()

---

socket_write

(PHP 4 >= 4.0.7, PHP 5)

socket_write — ソケットに書き込む

説明

int **socket_write** (resource $socket , string $buffer [, int $length])

関数 **socket_write()** は、 buffer の内容をソケット socket に書き込みます。

パラメータ

socket

buffer

書き込まれるバッファ。

length

オプションのパラメータ length で、 ソケットに書き込むバイト数を指定することが可能です。 この値がバッファの長さより大きい場合、自動的にバッファのサイズに切り詰められます。

返り値

ソケットへの書き込みに成功したデータのバイト数を返します。エラー時には **FALSE** を返します。エラーコードは socket_last_error() を用いて取得することができ、この値を socket_strerror() に渡すことでエラー情報を文字列で取得可能です。

> **注意**: **socket_write()** がゼロを返すことも十分ありえます。 これは、書き込むデータが存在しなかったことを意味します。 エラーをチェックするために **FALSE** かどうかを調べる際には、必ず === 演算子を使用しましょう。

注意

> **注意**: **socket_write()** は、バッファの内容を必ずしもすべて 書き込むとは限りません。ネットワークバッファの状態にもよりますが、たとえ 1 バイトだけ書き込まれたのであったとしても、それはエラーではなく正常な動作です。 そのため、データがすべて書き込まれたのかどうかに注意する必要があります。

参考

- socket_accept()
- socket_bind()
- socket_connect()
- socket_listen()
- socket_read()
- socket_strerror()

目次

- socket_read — ソケットから最大バイト長まで読みこむ
- socket_recv — 接続したソケットからデータを受信する
- socket_recvfrom — 接続しているかどうかによらず、ソケットからデータを受信する
- socket_select — 与えられたソケットの配列に対し、指定した有効時間で select() システムコールを実行する
- socket_send — 接続したソケットにデータを送信する
- socket_sendto — 接続しているかどうかによらずソケットにメッセージを送信する
- socket_set_block — ソケットリソースをブロックモードに設定する
- socket_set_nonblock — ソケットリソースを非ブロックモードに設定する
- socket_set_option — ソケットのオプションを設定する
- socket_shutdown — 受信、送信、または送受信用のソケットをシャットダウンする
- socket_strerror — ソケットエラーの内容を文字列として返す
- socket_write — ソケットに書き込む

Standard PHP Library (SPL) 関数

導入

SPLは、標準的な問題を解決するためのインターフェイスやクラスを集めたものです。

ヒント

SPL のより詳細なドキュメントは » こちら にあります。

インストール手順

このエクステンションは、PHP 5 ではデフォルトでコンパイルされ利用可能です。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

警告

PHP 5.1 以降、SPL はクラス定数を使用します。それより前のリリースでは、 **RIT_LEAVES_ONLY** のような形式のグローバル定数を使用します。

RecursiveIteratorIterator::LEAVES_ONLY (integer)
RecursiveIteratorIterator::SELF_FIRST (integer)
RecursiveIteratorIterator::CHILD_FIRST (integer)
CachingIterator::CALL_TOSTRING (integer)
CachingIterator::CATCH_GET_CHILD (integer)

ArrayIterator::current

(PHP 5)

ArrayIterator::current — 現在の配列エントリを返す

説明

mixed **ArrayIterator::current** (void)

この関数は現在の配列エントリを返します。

Example#1 ArrayIterator::current() の例

```
<?php
$array = array('1' => 'one',
               '2' => 'two',
               '3' => 'three');

$arrayobject = new ArrayObject($array);

for($iterator = $arrayobject->getIterator();
    $iterator->valid();
    $iterator->next()) {

    echo $iterator->key() . ' => ' . $iterator->current() . "¥n";
}
?>
```

上の例の出力は以下となります。

```
1 => one
2 => two
3 => three
```

ArrayIterator::key

(PHP 5)

ArrayIterator::key — 現在の配列キーを返す

説明

mixed **ArrayIterator::key** (void)

この関数は、現在の配列キーを返します

Example#1 ArrayIterator::key() の例

```
<?php
$array = array('key' => 'value');

$arrayobject = new ArrayObject($array);
$iterator = $arrayobject->getIterator();

echo $iterator->key(); //key
?>
```

ArrayIterator::next

(PHP 5)

ArrayIterator::next — 次のエントリに移動する

説明

void **ArrayIterator::next** (void)

この関数は次のエントリにイテレータを移動します。

Example#1 ArrayIterator::next() の例

```
<?php
$arrayobject = new ArrayObject();

$arrayobject[] = 'zero';
$arrayobject[] = 'one';

$iterator = $arrayobject->getIterator();

while($iterator->valid()) {
    echo $iterator->key() . ' => ' . $iterator->current() . "¥n";

    $iterator->next();
}
?>
```

上の例の出力は以下となります。

```
0 => zero
1 => one
```

ArrayIterator::rewind

(PHP 5)

ArrayIterator::rewind — 配列を最初に巻き戻す

説明

void **ArrayIterator::rewind** (void)

この関数はイテレータを最初に巻き戻します。

Example#1 ArrayIterator::rewind() の例

```
<?php
$arrayobject = new ArrayObject();

$arrayobject[] = 'zero';
$arrayobject[] = 'one';
$arrayobject[] = 'two';

$iterator = $arrayobject->getIterator();

$iterator->next();
echo $iterator->key(); //1

$iterator->rewind(); //最初に巻き戻す
echo $iterator->key(); //0
?>
```

ArrayIterator::seek

(PHP 5)

ArrayIterator::seek — 位置を検索します

説明

void **ArrayIterator::seek** (int $position)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ArrayIterator::valid

(PHP 5)

ArrayIterator::valid — 配列がまだエントリを持っているかどうかチェックする

説明

bool **ArrayIterator::valid** (void)

この関数は、配列がまだエントリを持っているかどうかチェックします。

Example#1 ArrayIterator::valid() の例

```
<?php
$array = array('1' => 'one');

$arrayobject  = new ArrayObject($array);
$iterator  = $arrayobject->getIterator();

var_dump($iterator->valid()); //bool(true)

$iterator->next(); //  次のアイテムに進める

//配列要素が 1 つしかないので bool(false)
var_dump($iterator->valid());
?>
```

ArrayObject::append

(PHP 5)

ArrayObject::append — 値を追加する

説明

void **ArrayObject::append** (mixed $newval)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ArrayObject::__construct

(PHP 5)

ArrayObject::__construct — 新規配列オブジェクトを生成する

説明

ArrayObject **ArrayObject::__construct** (mixed $input)

新規配列オブジェクトを生成します。input パラメータには、配列もしくは他の ArrayObject を指定可能です。

Example#1 ArrayObject::__construct() の例

```
<?php
$array = array('1' => 'one',
               '2' => 'two',
               '3' => 'three');

$arrayobject  = new ArrayObject($array);

var_dump($arrayobject);
?>
```

上の例の出力は以下となります。

```
object(ArrayObject)#1 (3) {
```

```
  [1]=>
  string(3) "one"
  [2]=>
  string(3) "two"
  [3]=>
  string(5) "three"
}
```

ArrayObject::count

(PHP 5)

ArrayObject::count — イテレータにある要素の数を返す

説明

int **ArrayObject::count** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ArrayObject::getIterator

(PHP 5)

ArrayObject::getIterator — ArrayObject インスタンスから新規イテレータを生成する

説明

ArrayIterator **ArrayObject::getIterator** (void)

この関数は ArrayObject インスタンスからイテレータを返します。

Example#1 ArrayObject::getIterator() の例

```php
<?php
$array = array('1' => 'one',
               '2' => 'two',
               '3' => 'three');

$arrayobject = new ArrayObject($array);

$iterator = $arrayobject->getIterator();

while($iterator->valid()) {
    echo $iterator->key() . ' => ' . $iterator->current() . "¥n";

    $iterator->next();
}
?>
```

上の例は以下を出力します。

```
1 => one
2 => two
3 => three
```

ArrayObject::offsetExists

(PHP 5)

ArrayObject::offsetExists — 要求された $index が存在するかどうかを返す

説明

bool **ArrayObject::offsetExists** (mixed $index)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ArrayObject::offsetGet

(PHP 5)

ArrayObject::offsetGet — 指定した $index の値を返す

説明

mixed **ArrayObject::offsetGet** (*mixed $index*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ArrayObject::offsetSet

(PHP 5)

ArrayObject::offsetSet — *指定した $index に $newval をセットする*

説明

void **ArrayObject::offsetSet** (*mixed $index* , *mixed $newval*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ArrayObject::offsetUnset

(PHP 5)

ArrayObject::offsetUnset — *指定した $index の値を解除する*

説明

void **ArrayObject::offsetUnset** (*mixed $index*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

CachingIterator::hasNext

(PHP 5)

CachingIterator::hasNext — *内部イテレータが有効な次の要素を持つかどうかをチェックする*

説明

bool **CachingIterator::hasNext** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

CachingIterator::next

(PHP 5)

CachingIterator::next — *イテレータを前方に移動する*

説明

void **CachingIterator::next** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

CachingIterator::rewind

(PHP 5)

CachingIterator::rewind — *イテレータを巻き戻す*

説明

void **CachingIterator::rewind** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

CachingIterator::__toString

(PHP 5)

CachingIterator::__toString — 現在の要素の文字列表現を返す

説明

string **CachingIterator::__toString** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

CachingIterator::valid

(PHP 5)

CachingIterator::valid — 現在の要素が有効かどうかをチェックする

説明

bool **CachingIterator::valid** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

CachingRecursiveIterator::getChildren

(PHP 5 <= 5.0.5)

CachingRecursiveIterator::getChildren — 内部イテレータの子を CachingRecursiveIterator として返す

説明

CachingRecursiveIterator **CachingRecursiveIterator::getChildren** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

CachingRecursiveIterator::hasChildren

(PHP 5 <= 5.0.5)

CachingRecursiveIterator::hasChildren — 内部イテレータの現在の要素が子を持つかどうかチェックする

説明

bolean **CachingRecursiveIterator::hasChildren** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::__construct

(PHP 5)

DirectoryIterator::__construct — パスから新規ディレクトリイテレータを生成する

説明

DirectoryIterator **DirectoryIterator::__construct** (string $path)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::current

(PHP 5)

DirectoryIterator::current — これ自身を返す (Iterator インターフェースが必要)

説明

DirectoryIterator **DirectoryIterator::current** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getATime

(PHP 5 >= 5.0.2)

DirectoryIterator::getATime — ファイルの最終アクセス時刻を取得する

説明

int **DirectoryIterator::getATime** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getCTime

(PHP 5 >= 5.0.2)

DirectoryIterator::getCTime — ファイルの inode 修正時刻を取得する

説明

int **DirectoryIterator::getCTime** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getFilename

(PHP 5)

DirectoryIterator::getFilename — 現在のディレクトリエントリのファイル名を返す

説明

string **DirectoryIterator::getFilename** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getGroup

(PHP 5 >= 5.0.2)

DirectoryIterator::getGroup — ファイルのグループを取得する

説明

int **DirectoryIterator::getGroup** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getInode

(PHP 5 >= 5.0.2)

DirectoryIterator::getInode — ファイルの inode を取得する

説明

int **DirectoryIterator::getInode** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getMTime

(PHP 5 >= 5.0.2)

DirectoryIterator::getMTime — ファイルの最終修正時刻を取得する

説明

int **DirectoryIterator::getMTime** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getOwner

(PHP 5 >= 5.0.2)

DirectoryIterator::getOwner — ファイルの所有者を取得する

説明

int **DirectoryIterator::getOwner** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getPath

(PHP 5 <= 5.1.1)

DirectoryIterator::getPath — ディレクトリパスを返す

説明

string **DirectoryIterator::getPath** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getPathname

(PHP 5 <= 5.1.1)

DirectoryIterator::getPathname — 現在のディレクトリエントリのパスとファイル名を返す

説明

string **DirectoryIterator::getPathname** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getPerms

(PHP 5 >= 5.0.2)

DirectoryIterator::getPerms — ファイルのパーミッションを取得する

説明

int **DirectoryIterator::getPerms** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getSize

(PHP 5 >= 5.0.2)

DirectoryIterator::getSize — ファイルサイズを取得する

説明

int **DirectoryIterator::getSize** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::getType

(PHP 5 >= 5.0.2)

DirectoryIterator::getType — ファイルタイプを取得する

説明

string **DirectoryIterator::getType** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::isDir

(PHP 5 <= 5.1.1)

DirectoryIterator::isDir — ファイルがディレクトリであれば *true* を返す

説明

bool **DirectoryIterator::isDir** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::isDot

(PHP 5)

DirectoryIterator::isDot — 現在のエントリが '.' もしくは '..' の場合 *true* を返す

説明

bool **DirectoryIterator::isDot** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::isExecutable

(PHP 5 <= 5.1.1)

DirectoryIterator::isExecutable — ファイルが実行可能な場合 *true* を返す

説明

bool **DirectoryIterator::isExecutable** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::isFile

(PHP 5 <= 5.1.1)

DirectoryIterator::isFile — ファイルが通常のファイルの場合 *true* を返す

説明

bool **DirectoryIterator::isFile** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::isLink

(PHP 5 <= 5.1.1)

DirectoryIterator::isLink — ファイルがシンボリックリンクの場合 *true* を返す

説明

bool **DirectoryIterator::isLink** (*void*)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::isReadable

(PHP 5 <= 5.1.1)

DirectoryIterator::isReadable — ファイルが読込可能であれば true を返す

説明

bool **DirectoryIterator::isReadable** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::isWritable

(PHP 5 <= 5.1.1)

DirectoryIterator::isWritable — ファイルが書き込み可能であれば true を返す

説明

bool **DirectoryIterator::isWritable** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::key

(PHP 5)

DirectoryIterator::key — 現在のディレクトリエントリを返す

説明

string **DirectoryIterator::key** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::next

(PHP 5)

DirectoryIterator::next — 次のエントリに移動する

説明

void **DirectoryIterator::next** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::rewind

(PHP 5)

DirectoryIterator::rewind — ディレクトリを最初に巻き戻す

説明

void **DirectoryIterator::rewind** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

DirectoryIterator::valid

(PHP 5)

DirectoryIterator::valid — ディレクトリがまだエントリを持っているかどうかチェックする

説明

string **DirectoryIterator::valid** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

FilterIterator::current

(PHP 5)

FilterIterator::current — 現在の要素の値を取得する

説明

mixed **FilterIterator::current** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

FilterIterator::getInnerIterator

(PHP 5)

FilterIterator::getInnerIterator — 内部イテレータを取得する

説明

Iterator **FilterIterator::getInnerIterator** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

FilterIterator::key

(PHP 5)

FilterIterator::key — 現在のキーを取得する

説明

mixed **FilterIterator::key** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

FilterIterator::next

(PHP 5)

FilterIterator::next — イテレータを前に移動する

説明

void **FilterIterator::next** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

FilterIterator::rewind

(PHP 5)

FilterIterator::rewind — イテレータを巻き戻す

説明

void **FilterIterator::rewind** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

FilterIterator::valid

(PHP 5)

FilterIterator::valid — 現在の要素が有効かどうかをチェックする

説明

bool **FilterIterator::valid** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

LimitIterator::getPosition

(PHP 5)

LimitIterator::getPosition — 現在の位置を返す

説明

int **LimitIterator::getPosition** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

LimitIterator::next

(PHP 5)

LimitIterator::next — イテレータを前に移動する

説明

void **LimitIterator::next** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

LimitIterator::rewind

(PHP 5)

LimitIterator::rewind — イテレータを指定したオフセットに巻き戻す

説明

void **LimitIterator::rewind** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

LimitIterator::seek

(PHP 5)

LimitIterator::seek — 与えられた位置を検索する

説明

void **LimitIterator::seek** (int $position)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

LimitIterator::valid

(PHP 5)

LimitIterator::valid — 現在の要素が有効かどうかをチェックする

説明

bool **LimitIterator::valid** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ParentIterator::getChildren

(PHP 5 <= 5.2.1)

ParentIterator::getChildren — ParentIterator に含まれる内部イテレータの子を返す

説明

ParentIterator **ParentIterator::getChildren** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ParentIterator::hasChildren

(PHP 5 <= 5.2.1)

ParentIterator::hasChildren — 内部イテレータの現在の要素が子を持つかどうかをチェックする

説明

bool **ParentIterator::hasChildren** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ParentIterator::next

(PHP 5)

ParentIterator::next — イテレータを前に移動する

説明

void **ParentIterator::next** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ParentIterator::rewind

(PHP 5)

ParentIterator::rewind — イテレータを巻き戻す

説明

void **ParentIterator::rewind** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveDirectoryIterator::getChildren

(No version information available, might be only in CVS)

RecursiveDirectoryIterator::getChildren — ディレクトリであれば、現在のエントリに対するイテレータを返す

説明

object **RecursiveDirectoryIterator::getChildren** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveDirectoryIterator::hasChildren

(PHP 5)

RecursiveDirectoryIterator::hasChildren — 現在のエントリがディレクトリかつ '.' もしくは '..' でないかどうかを返す

説明

bool **RecursiveDirectoryIterator::hasChildren** ([bool $allow_links])
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveDirectoryIterator::key

(PHP 5)

RecursiveDirectoryIterator::key — 現在のディレクトリエントリのパスとファイル名を返す

説明

string **RecursiveDirectoryIterator::key** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveDirectoryIterator::next

(PHP 5)

RecursiveDirectoryIterator::next — 次のエントリに移動する

説明

void **RecursiveDirectoryIterator::next** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveDirectoryIterator::rewind

(PHP 5)

RecursiveDirectoryIterator::rewind — ディレクトリを最初に巻き戻す

説明

void **RecursiveDirectoryIterator::rewind** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveIteratorIterator::current

(PHP 5)

RecursiveIteratorIterator::current — 現在の要素の値にアクセスする

説明

mixed **RecursiveIteratorIterator::current** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveIteratorIterator::getDepth

(PHP 5)

RecursiveIteratorIterator::getDepth — 再帰的なイテレーションにおける現在の深さを取得する

説明

int **RecursiveIteratorIterator::getDepth** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveIteratorIterator::getSubIterator

(PHP 5)

RecursiveIteratorIterator::getSubIterator — 現在の有効なサブイテレータを取得する

説明

RecursiveIterator **RecursiveIteratorIterator::getSubIterator** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveIteratorIterator::key

(PHP 5)

RecursiveIteratorIterator::key — 現在のキーにアクセスする

説明

mixed **RecursiveIteratorIterator::key** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveIteratorIterator::next

(PHP 5)

RecursiveIteratorIterator::next — 次の要素に移動する

説明

void **RecursiveIteratorIterator::next** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveIteratorIterator::rewind

(PHP 5)

RecursiveIteratorIterator::rewind — トップレベルの内部イテレータの先頭要素にイテレータを巻き戻す

説明

void **RecursiveIteratorIterator::rewind** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

RecursiveIteratorIterator::valid

(PHP 5)

RecursiveIteratorIterator::valid — 現在の位置が有効かどうかをチェックする

説明

bolean **RecursiveIteratorIterator::valid** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SimpleXMLIterator::current

(PHP 5 >= 5.1.3)

SimpleXMLIterator::current — 現在の SimpleXML エントリを返す

説明

mixed **SimpleXMLIterator::current** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SimpleXMLIterator::getChildren

(PHP 5 >= 5.1.3)

SimpleXMLIterator::getChildren — SimpleXML である場合、現在のエントリに対するイテレータを返す

説明

object **SimpleXMLIterator::getChildren** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SimpleXMLIterator::hasChildren

(PHP 5 >= 5.1.3)

SimpleXMLIterator::hasChildren — 現在のエントリが SimpleXML オブジェクトかどうかを返す

説明

bool **SimpleXMLIterator::hasChildren** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SimpleXMLIterator::key

(PHP 5 >= 5.1.3)

SimpleXMLIterator::key — 現在の SimpleXML のキーを返す

説明

mixed **SimpleXMLIterator::key** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SimpleXMLIterator::next

(PHP 5 >= 5.1.3)

SimpleXMLIterator::next — 次のエントリに移動する

説明

void **SimpleXMLIterator::next** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SimpleXMLIterator::rewind

(PHP 5 >= 5.1.3)

SimpleXMLIterator::rewind — SimpleXML を最初に巻き戻す

説明

void **SimpleXMLIterator::rewind** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

SimpleXMLIterator::valid

(PHP 5 >= 5.1.3)

SimpleXMLIterator::valid — SimpleXML がまだエントリを持っているかどうかをチェックする

説明

bool **SimpleXMLIterator::valid** (void)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

class_implements

(PHP 5)

class_implements — 与えられたクラスが実装しているインターフェースを返す

説明

array **class_implements** (*mixed $class [, bool $autoload]*)

この関数は、与えられたクラス *class* とその親が実装しているインターフェースを配列で返します。

パラメータ

class

オブジェクト（クラスインターフェース）もしくは文字列（クラス名）を指定します。

autoload

__autoload マジックメソッドを通じて、 この関数にクラスを自動的にロードさせるかどうかを指定します。 デフォルトは **TRUE** です。

返り値

配列もしくはエラー時に **FALSE** を返します。

変更履歴

バージョン	説明
5.1.0	文字列として *class* パラメータを渡すオプションが追加されました。 *autoload* パラメータが追加されました。

例

Example#1 class_implements() の例

```
<?php

interface foo { }
class bar implements foo {}

print_r(class_implements(new bar));

// PHP 5.1.0 以降、パラメータを文字列として指定しても良い
print_r(class_implements('bar'));


function __autoload($class_name) {
    require_once $class_name . '.php';
}

// 'not_loaded' クラスをロードするために __autoload を使用する
print_r(class_implements('not_loaded', true));

?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [foo] => foo
)

Array
(
     [interface_of_not_loaded] => interface_of_not_loaded
)
```

参考

- class_parents()

---

class_parents

(PHP 5)

class_parents — 与えられたクラスの親クラスを返す

説明

array **class_parents** (*mixed $class [, bool $autoload]*)

この関数は、与えられたクラス *class* の親クラス名を 配列で返します。

パラメータ

class

オブジェクトもしくはクラスの文字列を指定します。

autoload

__autoload マジックメソッドを通じて、 この関数にクラスを自動的にロードさせるかどうかを指定します。 デフォルトは TRUE です。

返り値

配列もしくはエラー時に FALSE を返します。

変更履歴

バージョン	説明
5.1.0	文字列として class パラメータを渡すオプションが追加されました。 autoload パラメータが追加されました。

例

Example#1 class_parents() の例

```
<?php

class foo { }
class bar extends foo {}

print_r(class_parents(new bar));

// PHP 5.1.0 以降、パラメータを文字列として指定しても良い
print_r(class_parents('bar'));


function __autoload($class_name) {
    require_once $class_name . '.php';
}

// 'not_loaded' クラスをロードするために __autoload を使用する
print_r(class_parents('not_loaded', true));
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [foo] => foo
)

Array
(
    [parent_of_not_loaded] => parent_of_not_loaded
)
```

参考

- class_implements()

---

iterator_count

(PHP 5 >= 5.1.3)

iterator_count — イテレータにある要素をカウントする

説明

int **iterator_count** (IteratorAggregate $iterator)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

iterator_to_array

(PHP 5 >= 5.1.3)

iterator_to_array — イテレータを配列にコピーする

説明

array **iterator_to_array** (IteratorAggregate $iterator [, bool $use_keys])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

spl_autoload_call

(PHP 5 >= 5.1.2)

spl_autoload_call — 要求されたクラスを読み込むために、すべての登録済みの __autoload() 関数を試す

説明

void **spl_autoload_call** (string $class_name)

この関数は、登録済みの __autoload 関数を使用して クラスあるいはインターフェイスを手動で探すために使用することができます。

---

spl_autoload_extensions

(PHP 5 >= 5.1.2)

spl_autoload_extensions — spl_autoload 用のデフォルトの拡張子を登録し、それを返す

説明

string **spl_autoload_extensions** ([string $file_extensions])

この関数は、__autoload 用の組み込み関数である spl_autoload が使用するファイル拡張子を変更したり調べたりします。 引数なしでコールした場合は、現在の拡張子一覧をカンマ区切りで返します。 ファイル拡張子を変更するには、 新しい拡張子一覧をカンマ区切り文字列で表したものを引数に指定して、 この関数を実行します。

---

spl_autoload_functions

(PHP 5 >= 5.1.2)

spl_autoload_functions — すべての登録済み __autoload() 関数を返す

説明

array **spl_autoload_functions** (void)

この関数は、すべての登録済み __autoload 関数を配列で返します。 autoload スタックが有効になっていない場合は、false が返されます。 関数が何も登録されていない場合は、空の配列が返されます。

---

spl_autoload_register

(PHP 5 >= 5.1.2)

spl_autoload_register — 指定した関数を __autoload() の実装として登録する

説明

bool **spl_autoload_register** ([callback $autoload_function])

指定した関数を、spl が提供する __autoload スタックに登録します。 スタックがまだアクティブになっていない場合は、まずアクティブにします。 パラメータが指定されなかった場合は、デフォルト実装である spl_autoload が登録されます。登録に成功した場合は true、 失敗した場合は false が返されます。

もしあなたのコード中に __autoload 関数が存在するのなら、 それを明示的に __autoload スタックに登録しなければなりません。 なぜなら、spl_autoload_register() は、 spl_autoload() あるいは spl_autoload_call() によって __autoload 関数のエンジンキャッシュを効率的に置き換えるからです。

---

spl_autoload_unregister

(PHP 5 >= 5.1.2)

spl_autoload_unregister — 指定した関数の、__autoload() の実装としての登録を解除する

説明

bool **spl_autoload_unregister** (mixed $autoload_function)

spl が提供する __autoload スタックから、関数の登録を解除します。 スタックがアクティブであり、かつこの関数の実行後にスタックが空になった場合は スタックが非アクティブ化されます。登録解除に成功した場合は true、 失敗した場合は false が返されます。

この関数の結果として autoload スタックがアクティブになったとした場合に、 既存の __autoload 関数が再アクティブ化されることはありません。

---

spl_autoload

(PHP 5 >= 5.1.2)

spl_autoload — __autoload() のデフォルト実装

説明

void **spl_autoload** (string $class_name [, string $file_extensions])

この関数は、__autoload() のデフォルト実装として使用されることを意図しています。 ほかに何も指定されておらず、autoload_register() がパラメータなしでコールされた場合には、 その後の __autoload() のコール時にはこの関数が使用されます。 デフォルトでは、クラス名を小文字にして .inc および .php を拡張子につけたファイル名のファイルが存在するかどうかを すべてのインクルードパスから探します。

---

spl_classes

(PHP 5)

spl_classes — 利用可能な SPL クラスを返す

説明

array **spl_classes** (void)

この関数は現在利用可能な SPL クラスを配列で返します。

Example#1 spl_classes() の例

```
<?php

print_r(spl_classes());

?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
      [ArrayObject] => ArrayObject
      [ArrayIterator] => ArrayIterator
      [CachingIterator] => CachingIterator
       [CachingRecursiveIterator] => CachingRecursiveIterator
      [DirectoryIterator] => DirectoryIterator
      [FilterIterator] => FilterIterator
      [LimitIterator] => LimitIterator
      [ParentIterator] => ParentIterator
       [RecursiveDirectoryIterator] => RecursiveDirectoryIterator
      [RecursiveIterator] => RecursiveIterator
       [RecursiveIteratorIterator] => RecursiveIteratorIterator
      [SeekableIterator] => SeekableIterator
      [SimpleXMLIterator] => SimpleXMLIterator
)
```

---

spl_object_hash

(PHP 5 >= 5.2.0)

spl_object_hash — 指定したオブジェクトのハッシュ ID を返す

説明

string **spl_object_hash** (object $obj)

この関数は、オブジェクトの一意な識別子を返します。この ID は、 オブジェクトを保存する際のハッシュのキーとして使用できますし、 オブジェクトを識別するための値として使用することもできます。

パラメータ

object

何らかのオブジェクト。

返り値

各オブジェクトに固有で、同一オブジェクトに対しては常に同じ値となる文字列を返します。

例

Example#1 spl_object_hash() の例

```
<?php
$id = spl_object_hash($object);
$storage[$id] = $object;
?>
```

目次

- ArrayIterator::rewind — 配列を最初に巻き戻す
- ArrayIterator::seek — 位置を検索します
- ArrayIterator::valid — 配列がまだエントリを持っているかどうかチェックする
- ArrayObject::append — 値を追加する
- ArrayObject::__construct — 新規配列オブジェクトを生成する
- ArrayObject::count — イテレータにある要素の数を返す
- ArrayObject::getIterator — ArrayObject インスタンスから新規イテレータを生成する
- ArrayObject::offsetExists — 要求された $index が存在するかどうかを返す
- ArrayObject::offsetGet — 指定した $index の値を返す
- ArrayObject::offsetSet — 指定した $index に $newval をセットする
- ArrayObject::offsetUnset — 指定した $index の値を解除する
- CachingIterator::hasNext — 内部イテレータが有効な次の要素を持つかどうかをチェックする
- CachingIterator::next — イテレータを前方に移動する
- CachingIterator::rewind — イテレータを巻き戻す
- CachingIterator::__toString — 現在の要素の文字列表現を返す
- CachingIterator::valid — 現在の要素が有効かどうかをチェックする
- CachingRecursiveIterator::getChildren — 内部イテレータの子を CachingRecursiveIterator として返す
- CachingRecursiveIterator::hasChildren — 内部イテレータの現在の要素が子を持つかどうかチェックする
- DirectoryIterator::__construct — パスから新規ディレクトリイテレータを生成する
- DirectoryIterator::current — これ自身を返す (Iterator インターフェースが必要)
- DirectoryIterator::getATime — ファイルの最終アクセス時刻を取得する
- DirectoryIterator::getCTime — ファイルの inode 修正時刻を取得する
- DirectoryIterator::getFilename — 現在のディレクトリエントリのファイル名を返す
- DirectoryIterator::getGroup — ファイルのグループを取得する
- DirectoryIterator::getInode — ファイルの inode を取得する
- DirectoryIterator::getMTime — ファイルの最終修正時刻を取得する
- DirectoryIterator::getOwner — ファイルの所有者を取得する
- DirectoryIterator::getPath — ディレクトリパスを返す
- DirectoryIterator::getPathname — 現在のディレクトリエントリのパスとファイル名を返す
- DirectoryIterator::getPerms — ファイルのパーミッションを取得する
- DirectoryIterator::getSize — ファイルサイズを取得する
- DirectoryIterator::getType — ファイルタイプを取得する
- DirectoryIterator::isDir — ファイルがディレクトリであれば true を返す
- DirectoryIterator::isDot — 現在のエントリが '.' もしくは '..' の場合 true を返す
- DirectoryIterator::isExecutable — ファイルが実行可能な場合 true を返す
- DirectoryIterator::isFile — ファイルが通常のファイルの場合 true を返す
- DirectoryIterator::isLink — ファイルがシンボリックリンクの場合 true を返す
- DirectoryIterator::isReadable — ファイルが読込可能であれば true を返す
- DirectoryIterator::isWritable — ファイルが書き込み可能であれば true を返す
- DirectoryIterator::key — 現在のディレクトリエントリを返す
- DirectoryIterator::next — 次のエントリに移動する
- DirectoryIterator::rewind — ディレクトリを最初に巻き戻す
- DirectoryIterator::valid — ディレクトリがまだエントリを持っているかどうかチェックする
- FilterIterator::current — 現在の要素の値を取得する
- FilterIterator::getInnerIterator — 内部イテレータを取得する
- FilterIterator::key — 現在のキーを取得する
- FilterIterator::next — イテレータを前に移動する
- FilterIterator::rewind — イテレータを巻き戻す
- FilterIterator::valid — 現在の要素が有効かどうかをチェックする
- LimitIterator::getPosition — 現在の位置を返す
- LimitIterator::next — イテレータを前に移動する
- LimitIterator::rewind — イテレータを指定したオフセットに巻き戻す
- LimitIterator::seek — 与えられた位置を検索する
- LimitIterator::valid — 現在の要素が有効かどうかをチェックする
- ParentIterator::getChildren — ParentIterator に含まれる内部イテレータの子を返す
- ParentIterator::hasChildren — 内部イテレータの現在の要素が子を持つかどうかをチェックする
- ParentIterator::next — イテレータを前に移動する
- ParentIterator::rewind — イテレータを巻き戻す
- RecursiveDirectoryIterator::getChildren — ディレクトリであれば、現在のエントリに対するイテレータを返す
- RecursiveDirectoryIterator::hasChildren — 現在のエントリがディレクトリかつ '.' もしくは '..' でないかどうかを返す
- RecursiveDirectoryIterator::key — 現在のディレクトリエントリのパスとファイル名を返す
- RecursiveDirectoryIterator::next — 次のエントリに移動する
- RecursiveDirectoryIterator::rewind — ディレクトリを最初に巻き戻す
- RecursiveIteratorIterator::current — 現在の要素の値にアクセスする

- RecursiveIteratorIterator::getDepth — 再帰的なイテレーションにおける現在の深さを取得する
- RecursiveIteratorIterator::getSubIterator — 現在の有効なサブイテレータを取得する
- RecursiveIteratorIterator::key — 現在のキーにアクセスする
- RecursiveIteratorIterator::next — 次の要素に移動する
- RecursiveIteratorIterator::rewind — トップレベルの内部イテレータの先頭要素にイテレータを巻き戻す
- RecursiveIteratorIterator::valid — 現在の位置が有効かどうかをチェックする
- SimpleXMLIterator::current — 現在の SimpleXML エントリを返す
- SimpleXMLIterator::getChildren — SimpleXML である場合、現在のエントリに対するイテレータを返す
- SimpleXMLIterator::hasChildren — 現在のエントリが SimpleXML オブジェクトかどうかを返す
- SimpleXMLIterator::key — 現在の SimpleXML のキーを返す
- SimpleXMLIterator::next — 次のエントリに移動する
- SimpleXMLIterator::rewind — SimpleXML を最初に巻き戻す
- SimpleXMLIterator::valid — SimpleXML がまだエントリを持っているかどうかをチェックする
- class_implements — 与えられたクラスが実装しているインターフェースを返す
- class_parents — 与えられたクラスの親クラスを返す
- iterator_count — イテレータにある要素をカウントする
- iterator_to_array — イテレータを配列にコピーする
- spl_autoload_call — 要求されたクラスを読み込むために、すべての登録済みの __autoload() 関数を試す
- spl_autoload_extensions — spl_autoload 用のデフォルトの拡張子を登録し、それを返す
- spl_autoload_functions — すべての登録済み __autoload() 関数を返す
- spl_autoload_register — 指定した関数を __autoload() の実装として登録する
- spl_autoload_unregister — 指定した関数の、__autoload() の実装としての登録を解除する
- spl_autoload — __autoload() のデフォルト実装
- spl_classes — 利用可能な SPL クラスを返す
- spl_object_hash — 指定したオブジェクトのハッシュ ID を返す

SQLite 関数

導入

この拡張モジュールは、SQLLite Embeddable SQL Database Engine 用の 拡張モジュールです。SQLiteは、組込み可能なSQLデータベースエンジン を実装するCライブラリです。SQLiteライブラリをリンクするプログラム は、別のRDBMSプロセスを実行することなくSQLデータベースにアクセスすることができます。

SQLiteは、巨大なデータベースサーバーに接続するために使用されるク ライアントライブラリではありません。 SQLiteがそのサーバーなのです。SQLiteライブラリは、ディスク上のデータベースを直接読み書きします。

> **注意**: より詳細な情報については、SQLiteのWebサイト (» http://sqlite.org/) を参照してください。

インストール

このパッケージに付属する INSTALL ファイルを読んでください。 または、単に PEAR インストーラで **pecl install sqlite** を実行してください。SQLite 自体も既に含まれており、 他のソフトウエアをインストールする必要は全くありません。

Windowsユーザは、DLL版のSQLite拡張モジュールを次の場所から入手可 能です。 (» php_sqlite.dll)

PHP 5 では、SQLite 拡張モジュールとエンジンは PHP 自身にバンドルされ、デフォルトでコンパイルされます。 しかし、PHP 5.1.0 以降では 手動で有効にする必要があります (共有モジュールとしてバンドルされるからです)。 さらに、PHP 5.1.0 以降では SQLite は PDO に依存するようになりました。そのため、 php.ini に以下の行を (この順に) 追加して PDO も 有効にしておく必要があります。

```
extension=php_pdo.dll
extension=php_sqlite.dll
```

Linux あるいは Unix オペレーティングシステムでは、 もし PDO を共有モジュールとしてビルドしたのなら SQLite も共有モジュールとしてビルドする必要があります。 そのためには、設定オプション **--with-sqlite=shared** を指定します。

SQLite 3 は、PDO SQLite でサポートされます。

> **注意**: **権限を持たないアカウントに対する Windows 版のインストール** Windows オペレーティングシステムでは、権限のないアカウントは デフォルトで設定される TMP 環境変数を利用できません。 これにより SQLite は Windows ディレクトリにテンポラリファイルを作成しますが、 望まれるものではありません。 そのため Web サーバもしくは Web サーバが動作しているユーザーアカウントに対して TMP 環境変数を設定すべきです。 もし、Apache を使用しているなら、httpd.conf ファイル内で **SetEnv** ディレクティブを使用することで実現可能です。 例えば、次のような感じです。
>
> ```
> SetEnv TMP c:/temp
> ```
>
> もしサーバレベルでこの設定を行うことができないのであれば、 スクリプト内で設定することができます。
> ```
> putenv('TMP=C:/temp');
> ```
> この設定は Web サーバがファイルを生成した後で書き込んだり削除したりする 権限を持たせるディレクトリを指定する必要があります。 そうでない場合、次のようなエラーメッセージを受け取るでしょう。 malformed database schema - unable to open a temporary database file for storing temporary tables

要件

以下の関数を利用可能とするには、SQLiteサポートを有効にしてPHPをコ ンパイルするか、php.iniで動的にSQLite拡張モジュールをロードする必要があります。

リソース型

SQLiteインターフェイスでは2種類のリソースが使用されています。最初 のリソースはデータベース接続で、2番目は結果セットです。

定義済の定数

関数sqlite_fetch_array() と sqlite_current() は、結果配列の種別を表すために定 数を使用します。以下の定数が定義されています。

SQLite 結果型定数
SQLITE_ASSOC (int)
カラムは、フィールド名を配列インデックスとする配列に返されます。
SQLITE_BOTH (int)
カラムは、数値インデックスとフィールド名による配列インデックスを 共に有する配列に返されます。
SQLITE_NUM (int)
カラムは、フィールドへの数値インデックスを有する配列に返されます。 このインデックスは 0 から始まり、結果の先頭フィールドとなります。

関数の戻り値はステータスコードです。以下の定数が定義されています。

SQLite ステータスコード定数
SQLITE_OK (int)
成功しました
SQLITE_ERROR (int)
SQL エラーあるいはデータベースが存在しません
SQLITE_INTERNAL (int)
SQLiteの内部ロジックエラーです
SQLITE_PERM (int)
アクセス権限がありません
SQLITE_ABORT (int)
コールバックルーチンが中断を要求しました
SQLITE_BUSY (int)
データベースファイルがロックされています
SQLITE_LOCKED (int)
データベース内のテーブルがロックされています
SQLITE_NOMEM (int)
メモリの割り当てに失敗しました
SQLITE_READONLY (int)
読み込み専用データベースに書き込もうとしました
SQLITE_INTERRUPT (int)
処理が内部的に終了しました
SQLITE_IOERR (int)
ディスク I/O エラーが発生しました
SQLITE_CORRUPT (int)
データベースのディスクイメージが不正です
SQLITE_NOTFOUND (int)
(内部的な) テーブルもしくはレコードが存在しません
SQLITE_FULL (int)
データベースが一杯のため挿入に失敗しました
SQLITE_CANTOPEN (int)
データベースファイルをオープンできません
SQLITE_PROTOCOL (int)
データベースロックプロトコルエラーです
SQLITE_EMPTY (int)
(内部的な) データベーステーブルが空です
SQLITE_SCHEMA (int)
データベーススキーマが変更されました
SQLITE_TOOBIG (int)
テーブルの 1 行に対するデータが多すぎます
SQLITE_CONSTRAINT (int)
制約違反のため中止しました
SQLITE_MISMATCH (int)
データ型が一致しません
SQLITE_MISUSE (int)
ライブラリが不正確に使用されました
SQLITE_NOLFS (int)
OS 機能の使用はホスト上でサポートされていません
SQLITE_AUTH (int)
認証に失敗しました
SQLITE_ROW (int)
内部プロセスが他の行を準備しました
SQLITE_DONE (int)
内部プロセスが実行を完了しました

定義済みクラス

SQLiteDatabase

オープンされている SQLite データベースを表す

コンストラクタ

- __construct - 新規 SQLiteDatabase オブジェクトを生成する

メソッド

- query - クエリを実行する
- queryExec - 結果を返さないクエリを実行する
- arrayQuery - クエリを実行し、結果を配列として返す
- singleQuery - クエリを実行し、単一カラムに対する配列もしくは先頭行の値を返す
- unbufferedQuery - バッファされていないクエリを実行する
- lastInsertRowid - 直近に挿入された行の行 ID を返す
- changes - 直近のステートメントにより更新された行数を返す

- createAggregate - SQL ステートメントで使用する集約 UDF を登録する
- createFunction - SQL ステートメントで使用する UDF を登録する
- busyTimeout - ビジータイムアウト時間を設定または無効にする
- lastError - 直近に発生したエラーのエラーコードを返す
- fetchColumnTypes - 特定のテーブルからカラム型の配列を返す

SQLiteResult

バッファされた SQLite の結果セットを表す

メソッド

- fetch - 結果セットから次行を配列として取得する
- fetchObject - 結果セットから次行をオブジェクトとして取得する
- fetchSingle - 結果セットから先頭カラムを文字列として取得する
- fetchAll - 結果セットから全行を配列の配列として取得する
- column - 結果セットの現在行からカラムを取得する
- numFields - 結果セット内のフィールド数を返す
- fieldName - 結果セット内の特定フィールドの名前を返す
- current - 結果セットから現在行を配列として取得する
- key - 現在行のインデックスを返す
- next - 次の行番号へシークする
- valid - まだ行が残っているかどうかを返す
- rewind - 結果セットの先頭の行番号へシークする
- prev - 結果セットの前の行番号へシークする
- hasPrev - 前の行が利用可能かどうかを返す
- numRows - 結果セットの行数を返す
- seek - 特定の行番号へシークする

SQLiteUnbuffered

バッファされていない SQLite 結果セットを表します。 欠課されていない結果セットはシーケンシャルで、前方シークのみ可能です。

メソッド

- fetch - 結果セットから次行を配列として取得する
- fetchObject - 結果セットから次行をオブジェクトとして取得する
- fetchSingle - 結果セットから先頭カラムを文字列として取得する
- fetchAll - 結果セットから全行を配列の配列として取得する
- column - 結果セットの現在行からカラムを取得する
- numFields - 結果セット内のフィールド数を返す
- fieldName - 結果セット内の特定フィールドの名前を返す
- current - 結果セットから現在行を配列として取得する
- next - 次の行番号へシークする
- valid - まだ行が残っているかどうかを返す

実行時設定

php.ini の設定により動作が変化します。

SQLite設定オプション

名前	デフォルト	変更の可否	変更履歴
sqlite.assoc_case	0	PHP_INI_ALL	PHP 5.0.0 から利用可能です

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

sqlite.assoc_case int

ハッシュのインデックスに大文字小文字混用(0)、 大文字(1)、小文字 (2)のど れを使用するかを指定します。

このオプションは、データベーススキーマ中での実際のフィールド名の ケースによらず、カラム名が常に大文字または小文字で返されるような 他の

データベースシステムとの互換性が必要な場合に特に有用です。

SQLiteライブラリは、カラム名をそのままのケース(これは、 スキーマで使用したケースに一致します)で返します。 sqlite.assoc_case に 0 を指定した場合、そのままのケースは保持されます。このオプションを 1 または 2 に設定した場合、 PHPはハッシュキーのケースをそれぞれ大文字または小文字のキーに変 換します。

このオプションを使用することで若干の性能劣化がありますが、 PHPスクリプトで自分で大文字/小文字変換を行うよりはかなり高速です。

---

sqlite_array_query

SQLiteDatabase->arrayQuery

(No version information available, might be only in CVS)

SQLiteDatabase->arrayQuery — 指定したデータベースに対してクエリを実行し、配列を返す

説明

```
array sqlite_array_query ( resource $dbhandle , string $query [, int $result_type [, bool $decode_binary ]] )
array sqlite_array_query ( string $query , resource $dbhandle [, int $result_type [, bool $decode_binary ]] )
```

オブジェクト指向言語型スタイル (メソッド):

```
SQLiteDatabase
array arrayQuery ( string $query [, int $result_type [, bool $decode_binary ]] )
```

sqlite_array_query() は与えられたクエリを実行し、 結果セット全体を配列で返します。これは、結果セットの各レコードに関して sqlite_query()に続いて sqlite_fetch_array()をコールすることに似ています。 **sqlite_array_query()** は前述の方法よりも著しく高速です。

ヒント

sqlite_array_query() は、返すレコードが45件以下 のクエリーで最も有効です。これ以上のデータがある場合には、 より性能を最適化するために、代わりに sqlite_unbuffered_query()を使用するようなスクリ プトを書くことをお薦めします。

パラメータ

query

実行するクエリ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

result_type

オプションの result_type パラメータには定数を指定でき、返される配列の添字を定義します。 **SQLITE_ASSOC** を用いると、連想配列の添字(名前フィールド)のみが返されます。一方、**SQLITE_NUM** は、数値の添字(フィールド番号)のみを返します。**SQLITE_BOTH** は、連想配列の添字と数値の添字の両方を返します。 **SQLITE_BOTH** がこの関数のデフォルトです。

decode_binary

decode_binary パラメータが **TRUE** (デフォルト)に設定された場合、PHP はバイナリエンコーディングをデコードします。これは、sqlite_escape_string() によりエンコードされたデータに適用されます。sqlite をサポートする他のアプリケーションにより作成されたデータベースを処理する時以外は、この値をデフォルトのままにしておくべきです。

> **注意**: (MySQL のような)他のデータベースエクステンションとの互換性のため、 2 種類の構文がサポートされています。推奨されるのは最初の構文で、dbhandle パラメータを関数の最初のパラメータとするものです。

返り値

結果セット全体の配列、その他の場合は **FALSE** を返します。

SQLITE_ASSOC および **SQLITE_BOTH** で返されるカラム名は、設定オプション sqlite.assoc_case の値に基づき、大文字小文字が変換されます。

例

Example#1 手続き言語型スタイル

```php
<?php
$dbhandle = sqlite_open('sqlitedb');
$result = sqlite_array_query($dbhandle, 'SELECT name, email FROM users LIMIT 25', SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Example#2 オブジェクト指向言語型スタイル

```php
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');
$result = $dbhandle->arrayQuery('SELECT name, email FROM users LIMIT 25', SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

参考

- sqlite_query()
- sqlite_fetch_array()
- sqlite_fetch_string()

---

sqlite_busy_timeout

SQLiteDatabase->busyTimeout

(No version information available, might be only in CVS)

SQLiteDatabase->busyTimeout — ビジータイムアウト時間を設定またはビジーハンドラを無効にする

説明

void **sqlite_busy_timeout** (resource $dbhandle , int $milliseconds)

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
void **busyTimeout** (int $milliseconds)

SQLite データベース database が利用可能となるまでの 最大の待ち時間を milliseconds に設定します。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

milliseconds

ミリ秒での数。 milliseconds が 0の場合、 ビジーハンドラは無効となり、他のプロセス/スレッドが更新用にデータベー スをロックしている際に、sqliteはSQLITE_BUSYを 直ちに返します。

PHPはデフォルトでデータベースがオープンされる際のビジータイムアウト を 60 秒に設定しています。

注意: 1 秒は 1000 ミリ秒です。

返り値

値を返しません。

例

Example#1 手続き言語型スタイル

```
<?php
$dbhandle = sqlite_open('sqlitedb');
sqlite_busy_timeout($dbhandle, 10000); // タイムアウトを 10 秒に設定する
sqlite_busy_timeout($dbhandle, 0); // ビジーハンドラを無効にする
?>
```

Example#2 オブジェクト指向言語型スタイル

```
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');
$dbhandle->busyTimeout(10000); // 10 秒にする
$dbhandle->busyTimeout(0); // 無効にする
?>
```

参考

- sqlite_open()

---

sqlite_changes

SQLiteDatabase->changes

(No version information available, might be only in CVS)

SQLiteDatabase->changes — 直近のSQLステートメントにより変更されたレコード数を返す

説明

int **sqlite_changes** (resource $dbhandle)

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
int **changes** (void)

データベースハンドル dbhandle に対して実行された 直近のSQLステートメントによって変更されたレコード数を返します。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

例

Example#1 手続き言語型スタイル

```
<?php
$dbhandle = sqlite_open('mysqlitedb');
$query = sqlite_query($dbhandle, "UPDATE users SET email='jDoe@example.com' WHERE username='jDoe'");
if (!$query) {
    exit('Error in query.');
} else {
    echo 'Number of rows modified: ', sqlite_changes($dbhandle);
}
?>
```

Example#2 オブジェクト指向言語型スタイル

```
<?php
$dbhandle = new SQLiteDatabase('mysqlitedb');
$query = $dbhandle->query("UPDATE users SET email='jDoe@example.com' WHERE username='jDoe'");
if (!$query) {
    exit('Error in query.');
} else {
    echo 'Number of rows modified: ', $dbhandle->changes();
}
?>
```

参考

- sqlite_open()

---

sqlite_close

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_close — オープンされたSQLiteデータベースを閉じる

説明

void **sqlite_close** (resource $dbhandle)

指定したデータベースハンドル database を閉じます。 このデータベースに持続性がある場合、このデータベースは閉じられ、 持続的データベースリストから削除されます。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。

返り値

値を返しません。

例

Example#1 sqlite_close() の例

```
<?php
$dbhandle = sqlite_open('sqlitedb');
sqlite_close($dbhandle);
?>
```

参考

- sqlite_open()
- sqlite_popen()

---

sqlite_column

SQLiteResult->column

SQLiteUnbuffered->column

(No version information available, might be only in CVS)

SQLiteUnbuffered->column — カレントの結果セットのレコードからカラムを1列取得する

説明

```
mixed sqlite_column ( resource $result , mixed $index_or_name [, bool $decode_binary ] )
SQLiteResult
mixed column ( mixed $index_or_name [, bool $decode_binary ] )
SQLiteUnbuffered
mixed column ( mixed $index_or_name [, bool $decode_binary ] )
```

クエリ結果ハンドルresult の現在のレコードから index_or_name (文字列の場合) というカラム名、 または、カラム番号 index_or_name (整数の場合) の値を取得します。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

index_or_name

取得するカラムインデックス、もしくは名前

decode_binary

decode_binary パラメータが TRUE (デフォルト)に設定された場合、PHP はバイナリエンコーディングをデコードします。これは、sqlite_escape_string() によりエンコードされたデータに適用されます。sqlite をサポートする他のアプリケーションにより作成されたデータベースを処理する時以外は、この値をデフォルトのままにしておくべきです。

注意

> **注意**: 多くのカラムもしくは膨大なデータを含んでいるカラムを含むような 大量の結果セットの反復処理を行う場合にこの関数を使用してください。

参考

- sqlite_fetch_string()

---

sqlite_create_aggregate

SQLiteDatabase->createAggregate

(No version information available, might be only in CVS)

SQLiteDatabase->createAggregate — SQLステートメントで使用する集約UDFを登録する

説明

```
void sqlite_create_aggregate ( resource $dbhandle , string $function_name , callback $step_func , callback
$finalize_func [, int $num_args ] )
```

オブジェクト指向言語型スタイル (メソッド):

```
SQLiteDatabase
void createAggregate ( string $function_name , callback $step_func , callback $finalize_func [, int $num_args ] )
```

sqlite_create_aggregate() は、 sqlite_create_function() に似ていますが、 クエリの全レコードを通じて集約された結果を計算するために使用される 関数を登録するところが異なります。

この関数と sqlite_create_function() の主な違い は、二つの関数が集約を管理するために必要であることです。 step_func は、結果セットの各レコードに関して コールされます。PHP関数は、結果を加算し、集約コンテキストに保存する 必要があります。 全レコードが処理された後、finalize_func がコールされ、 集約コンテキストからデータが取得され、結果が返されます。 コールバック関数は SQLite が認識可能な型 (すなわち スカラー型) を返す必要があります。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

function_name

SQL ステートメントで使用される関数名

step_func

結果セットの各レコードに対してコールされるコールバック関数

finalize_func

各レコードからの "段階的な" データを集約するためのコールバック関数

num_args

見積もられた引数の数をコールバック関数が受け入れる場合に SQLite パーサへ渡すヒント

返り値

値を返しません。

例

Example#1 max_length 集約関数の例

```
<?php
$data = array(
    'one',
    'two',
    'three',
    'four',
    'five',
    'six',
    'seven',
    'eight',
    'nine',
    'ten',
    );
$dbhandle = sqlite_open(':memory:');
sqlite_query($dbhandle, "CREATE TABLE strings(a)");
foreach ($data as $str) {
    $str = sqlite_escape_string($str);
    sqlite_query($dbhandle, "INSERT INTO strings VALUES ('$str')");
}

function max_len_step(&$context, $string)
{
    if (strlen($string) > $context) {
        $context = strlen($string);
    }
}

function max_len_finalize(&$context)
{
    return $context;
}

sqlite_create_aggregate($dbhandle, 'max_len', 'max_len_step', 'max_len_finalize');

var_dump(sqlite_array_query($dbhandle, 'SELECT max_len(a) from strings'));

?>
```

この例では、テーブルのあるカラムに存在する最長な文字列長を計算する 集約関数を生成します。各レコードに対して max_len_step 関数がコールされ、context パラメータが渡されます。 コンテキストパラメータは他の PHP 変数と同様で、 配列もしくはオブジェクト値を保持するよう設定されます。 この例では、単純にこれまでの最大長を保持するために使用しています。 もし string が現在の最大長よりも長い場合、 新しい最大長を保持するためにコンテキストを更新します。

全てのレコードが処理された後、SQLite は集約結果を決定するために max_len_finalize 関数をコールします。 ここで、context 内に見つかったデータに基づいた 計算のような処理を実行することができます。 ただ、この簡単な例では、クエリの処理が進むと共に結果を計算しているので、単純にコンテキストの値を返す必要があります。

> **注意**: 上記の例は、カラムがバイナリデータを含む場合に正しく動作しません。 なぜそうなるか、またどのようにバイナリエンコーディングを考慮するか についてはマニュアルページの sqlite_udf_decode_binary() を参照ください。

ヒント

SQLite がクエリを処理するために大量のメモリを使用する原因になるので、 コンテキストの値のコピーをストアした後でそれらを処理することは 推奨されません。 もし 32 バイトの文字列を含む 100 万レコードがメモリにストアされた場合、 どの程度のメモリが必要になるか考えてみてください。

ヒント

sqlite_create_function() や **sqlite_create_aggregate()** を用いることで、 SQLite のネイティブな SQL 関数をオーバーライドすることが可能です。

参考

- sqlite_create_function()
- sqlite_udf_encode_binary()
- sqlite_udf_decode_binary()

---

sqlite_create_function

SQLiteDatabase->createFunction

(No version information available, might be only in CVS)

SQLiteDatabase->createFunction — SQLステートメントで使用するために"通常の"ユーザ定義関数を登録する

説明

void **sqlite_create_function** (resource $dbhandle , string $function_name , callback $callback [, int $num_args])

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
void **createFunction** (string $function_name , callback $callback [, int $num_args])

sqlite_create_function() により、SQLiteにPHP関数 をUDF (ユーザ定義関数)として登録することが可能で す。この関数は、SQLステートメン

トの中からコールできます。

UDFは、SELECTおよびUPDATEステートメント、そして、トリガーの中のよう に関数をコールできる全てのSQLステートメントで使用可能です。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

function_name

SQL ステートメントで使用する関数名

callback

定義された SQL 関数を処理するためのコールバック関数

> **注意**: コールバック関数は SQLite で有効な型（例えば スカラー型）を返す必要があります

num_args

コールバック関数が規定の引数の数を受け入れるかどうかを決定するため SQLite パーサに渡すヒント

> **注意**: (MySQL のような)他のデータベースエクステンションとの互換性のため、 2 種類の構文がサポートされています。推奨されるのは最初の構文で、dbhandle パラメータを関数の最初のパラメータとするものです。

返り値

値を返しません。

例

Example#1 sqlite_create_function() の例

```
<?php
function md5_and_reverse($string)
{
    return strrev(md5($string));
}

if ($dbhandle = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {

    sqlite_create_function($dbhandle, 'md5rev', 'md5_and_reverse', 1);

    $sql  = 'SELECT md5rev(filename) FROM files';
    $rows = sqlite_array_query($dbhandle, $sql);
} else {
    echo 'Error opening sqlite db: ' . $sqliteerror;
    exit;
}
?>
```

この例では、文字列のMD5サムを計算し、順番を反転する関数が記述されています。 このSQLステートメントが実行された場合、 関数により変換されたファイル名の値を返します。 $rows により返されるデータには、 処理結果が含まれています。

この技術の美しいところは、データのクエリーを実行した後で foreach() ループにより結果を処理する必要がないことです。

PHP は、データベースが最初にオープンされる際に php という名前の特別な関数を登録します。 このphp関数は、事前に登録することなしにあらゆるPHP関数をコールする ために使用可能です。

Example#2 PHP 関数の使用例

```
<?php
$rows = sqlite_array_query($dbhandle, "SELECT php('md5', filename) from files");
?>
```

この例は、データベースの各 filename カラムにつ いて md5() をコールし、その結果を $rows に返します。

> **注意**: 性能上の理由から、PHPはUDFとの間で送受信されるバイナリデータを自動 的にエンコード/デコードしません。この方法でバイナリデータを処理す る必要がある場合、パラメータを手動でエンコード/デコードし、 値を返すようにする必要があります。 詳細については、sqlite_udf_encode_binary() および sqlite_udf_decode_binary()を参照して下さ い。

ヒント

適用するアプリケーションの主要な要求が高い性能でない限り、バイナリ データの処理を行うためにUDFを使用することは推奨されません。

ヒント

SQLiteのネーティブSQL関数をオーバーライドするために **sqlite_create_function()**および sqlite_create_aggregate()も使用可能です。

参考

- sqlite_create_aggregate()

---

sqlite_current

SQLiteResult->current

SQLiteUnbuffered->current

(No version information available, might be only in CVS)

SQLiteUnbuffered->current — 結果セットからカレントのレコードを配列として取得する

説明

array **sqlite_current** (resource $result [, int $result_type [, bool $decode_binary]])

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
array **current** ([int $result_type [, bool $decode_binary]])
SQLiteUnbuffered
array **current** ([int $result_type [, bool $decode_binary]])

sqlite_current() は sqlite_fetch_array() と同じですが、 データを返す前に次のレコードに移動せず、カレントの位置からのみデー タを返すという違いがあります。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

result_type

オプションの result_type パラメータには定数を指定でき、返される配列の添字を定義します。 **SQLITE_ASSOC** を用いると、連想配列の添字(名前フィールド)のみが返されます。一方、**SQLITE_NUM** は、数値の添字(フィールド番号)のみを返します。**SQLITE_BOTH** は、連想配列の添字と数値の添字の両方を返します。 **SQLITE_BOTH** がこの関数のデフォルトです。

decode_binary

decode_binary パラメータが **TRUE** (デフォルト)に設定された場合、PHP はバイナリエンコーディングをデコードします。これは、sqlite_escape_string() によりエンコードされたデータに適用されます。sqlite をサポートする他のアプリケーションにより作成されたデータベースを処理する時以外は、この値をデフォルトのままにしておくべきです。

返り値

結果セットからカレントレコードの配列を返します。 現在の位置が最終レコード以降の場合、 **FALSE** を返します。

SQLITE_ASSOC および **SQLITE_BOTH** で返されるカラム名は、設定オプション sqlite.assoc_case の値に基づき、大文字小文字が変換されます。

参考

- sqlite_seek()
- sqlite_next()
- sqlite_fetch_array()

---

sqlite_error_string

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_error_string — エラーコードの説明を返す

説明

string **sqlite_error_string** (int $error_code)

sqlite_last_error()から返される error_code の可読性が高い説明を返します。

参考

- sqlite_last_error()

---

sqlite_escape_string

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_escape_string — クエリパラメータ用に文字列をエスケープする

説明

string **sqlite_escape_string** (string $item)

sqlite_escape_string() は、 item で指定した文字列を SQLite SQLステートメントで使用できるように正しくクオートします。 この際、シングルクオート(')は2重にされ、 クエリ文字列のバイナリセーフでない文字がチェックされます。

item がNUL文字を含む場合、 または、16進表現の0x01で始まる場合、 PHPはバイナリデータを安全に保存/取得するためのバイナリエンコーディ ングを適用します。

このエンコーディングは、データ挿入を安全に行いますが、 テキスト比較は単純化され、バイナリデータを含むカラムについては クエリの中でLIKE句を使用できません。 実際には、バイナリカラムでこのようなことをするスキーマにはしないの で、これが問題になることはないでしょう。 (実際に

は、ファイルのような他の手段でバイナリデータを保存する方が良いでしょう。)

警告

addslashes()をSQLiteクエリの文字列をクオートするために使用するべきではありません。さもないと、データを取得する際に奇妙な結果を発生する可能性があります。

> **注意**: この関数を sqlite_create_function()または sqlite_create_aggregate()により作成したUDFから返す値をエンコードするために使用しないでください。代わりに sqlite_udf_encode_binary() を使用してください。

参考

- sqlite_udf_encode_binary()

---

sqlite_exec

SQLiteDatabase->exec

(No version information available, might be only in CVS)

SQLiteDatabase->exec — 与えられたデータベースに対して結果を伴わないクエリを実行する

説明

```
bool sqlite_exec ( resource $dbhandle , string $query [, string &$error_msg ] )
bool sqlite_exec ( string $query , resource $dbhandle )
```

オブジェクト指向言語型スタイル(メソッド):

```
SQLiteDatabase
bool queryExec ( string $query [, string &$error_msg ] )
```

与えられたデータベースハンドル(dbhandle パラメータで指定される)に対して query によって指定される SQL ステートメントを実行します。

警告

SQLiteは、セミコロンで区切られた複数のクエリを実行します。これにより、ファイルからロードするかスクリプトに埋め込んだ SQL をバッチ実行することができます。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、sqlite_open() から返されます。このパラメータは、オブジェクト指向言語型メソッドを使用する場合は不要です。

query

実行するクエリを指定します。

error_msg

エラーが発生した場合、指定された変数に詰め込まれます。SQL 文法エラーは sqlite_last_error() 関数では取得できないので、これは特に重要です。

> **注意**: (MySQL のような)他のデータベースエクステンションとの互換性のため、2 種類の構文がサポートされています。推奨されるのは最初の構文で、dbhandle パラメータを関数の最初のパラメータとするものです。

返り値

この関数はブール型の結果を返します。成功時は **TRUE**、失敗時は **FALSE** を返します。もしレコードを返すクエリを実行する必要がある場合は sqlite_query() を参照ください。

SQLITE_ASSOC および **SQLITE_BOTH** で返されるカラム名は、設定オプション sqlite.assoc_case の値に基づき、大文字小文字が変換されます。

変更履歴

バージョン	説明
5.1.0	error_msg パラメータが追加されました。

例

Example#1 手続き型言語スタイルでの例

```
<?php
$dbhandle = sqlite_open('mysqlitedb');
$query = sqlite_exec($dbhandle, "UPDATE users SET email='jDoe@example.com' WHERE username='jDoe'", $error);
if (!$query) {
    exit("Error in query: '$error'");
} else {
    echo 'Number of rows modified: ', sqlite_changes($dbhandle);
}
?>
```

Example#2 オブジェクト指向言語型スタイルでの例

```
<?php
$dbhandle = new SQLiteDatabase('mysqlitedb');
```

```
$query = $dbhandle->queryExec("UPDATE users SET email='jDoe@example.com' WHERE username='jDoe'", $error);
if (!$query) {
    exit("Error in query: '$error'");
} else {
    echo 'Number of rows modified: ', $dbhandle->changes();
}
?>
```

参考

- sqlite_query()
- sqlite_unbuffered_query()
- sqlite_array_query()

---

sqlite_factory

(PHP 5)

sqlite_factory — SQLite データベースをオープンし、SQLiteDatabse オブジェクトを返す

説明

SQLiteDatabase **sqlite_factory** (string $filename [, int $mode [, string &$error_message]])

sqlite_factory() は sqlite_open() と同様に SQLite データベースをオープン、もしくはデータベースが存在しない場合は生成しようと試みます。しかしながら、リソースの代わりに SQLiteDatabase オブジェクトが返されます。詳細な使用法と注意事項については、 sqlite_open() を参照ください。

パラメータ

filename

SQLite データベースのファイル名

mode

ファイルのモード。読み込み専用モードでデータベースをオープンするために 使用することを目的としています。現在、このパラメータは SQLite ライブラリに無視されます。このモードのデフォルト値は、 8 進数値 0666 で、これは推奨される値です。

error_message

参照として渡され、エラーが発生した場合に データベースがオープンできなかった原因を説明する 記述的なエラーメッセージを保持するために設定されます。

返り値

成功時に SQLiteDatabase オブジェクト、失敗時に **NULL** を返します。

例

Example#1 sqlite_factory() の例

```
<?php
$dbhandle = sqlite_factory('sqlitedb');
$dbhandle->query('SELECT user_id, username FROM users');

/* functionally equivalent to: */

$dbhandle = new SQLiteDatabase('sqlitedb');
$dbhandle->query('SELECT user_id, username FROM users');

?>
```

参考

- sqlite_open()
- sqlite_popen()

---

sqlite_fetch_all

SQLiteResult->fetchAll

SQLiteUnbuffered->fetchAll

(No version information available, might be only in CVS)

SQLiteUnbuffered->fetchAll — 結果セットから全てのレコードを配列の配列として取得する

説明

array **sqlite_fetch_all** (resource $result [, int $result_type [, bool $decode_binary]])

オブジェクト指向言語型スタイル (メソッド):

```
SQLiteResult
array fetchAll ([ int $result_type [, bool $decode_binary ]] )
SQLiteUnbuffered
array fetchAll ([ int $result_type [, bool $decode_binary ]] )
```

sqlite_fetch_all() *は結果リソース* result *から全結果セットの配列を返します。 これは* sqlite_query() *(もしくは* sqlite_unbuffered_query()*) の後に結果セットの各レコードに対して* sqlite_fetch_array() *を行うことと等価です。*

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

result_type

オプションの result_type パラメータには定数を指定でき、返される配列の添字を定義します。 **SQLITE_ASSOC** *を用いると、連想配列の添字(名前フィールド)のみが返されます。一方、* **SQLITE_NUM** *は、数値の添字(フィールド番号)のみを返します。* **SQLITE_BOTH** *は、連想配列の添字と数値の添字の両方を返します。* **SQLITE_BOTH** *がこの関数のデフォルトです。*

decode_binary

decode_binary パラメータが **TRUE** *(デフォルト)に設定された場合、PHP はバイナリエンコーディングをデコードします。これは、* sqlite_escape_string() *によりエンコードされたデータに適用されます。sqlite をサポートする他のアプリケーションにより作成されたデータベースを処理する時以外は、この値をデフォルトのままにしておくべきです。*

返り値

結果セットからのカレントレコードの配列を返します。 sqlite_query() *の直後にコールされた場合、 全ての行を返します。* sqlite_fetch_array() *の後にコールされた場合、残りの行を返します。 結果セットに行が残っていない場合、空の配列を返します。*

SQLITE_ASSOC *および* **SQLITE_BOTH** *で返されるカラム名は、設定オプション* sqlite.assoc_case *の値に基づき、大文字小文字が変換されます。*

例

Example#1 手続き型言語スタイルでの例

```php
<?php
$dbhandle = sqlite_open('sqlitedb');
$query = sqlite_query($dbhandle, 'SELECT name, email FROM users LIMIT 25');
$result = sqlite_fetch_all($query, SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Example#2 オブジェクト指向言語型スタイルでの例

```php
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');

$query = $dbhandle->query('SELECT name, email FROM users LIMIT 25'); // buffered result set
$query = $dbhandle->unbufferedQuery('SELECT name, email FROM users LIMIT 25'); // unbuffered result set

$result = $query->fetchAll(SQLITE_ASSOC);
foreach ($result as $entry) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

参考

- sqlite_fetch_array()

---

sqlite_fetch_array

SQLiteResult->fetch

SQLiteUnbuffered->fetch

(No version information available, might be only in CVS)

SQLiteUnbuffered->fetch — 結果セットから次のレコードを配列として取得する

説明

array **sqlite_fetch_array** *(resource $result [, int $result_type [, bool $decode_binary]])*

オブジェクト指向言語型スタイル (メソッド):

```
SQLiteResult
array fetch ([ int $result_type [, bool $decode_binary ]] )
SQLiteUnbuffered
array fetch ([ int $result_type [, bool $decode_binary ]] )
```

指定した結果ハンドル result から次のレコードを取得 します。レコードがもうない場合は **FALSE***を返し、それ以外は レコードデータを含む連想配列を返します。*

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

result_type

オプションの result_type パラメータには定数を指定でき、返される配列の添字を定義します。 SQLITE_ASSOC を用いると、連想配列の添字(名前フィールド)のみが返されます。一方、SQLITE_NUM は、数値の添字(フィールド番号)のみを返します。SQLITE_BOTH は、連想配列の添字と数値の添字の両方を返します。 SQLITE_BOTH がこの関数のデフォルトです。

decode_binary

decode_binary パラメータが TRUE (デフォルト)に設定された場合、PHP はバイナリエンコーディングをデコードします。これは、sqlite_escape_string() によりエンコードされたデータに適用されます。sqlite をサポートする他のアプリケーションにより作成されたデータベースを処理する時以外は、この値をデフォルトのままにしておくべきです。

返り値

結果セットの次レコードの配列を返します。 次レコードの位置が最終レコード以降の場合、FALSE を返します。

SQLITE_ASSOC および SQLITE_BOTH で返されるカラム名は、設定オプション sqlite.assoc_case の値に基づき、大文字小文字が変換されます。

例

Example#1 手続き型言語スタイルでの例

```
<?php
$dbhandle = sqlite_open('sqlitedb');
$query = sqlite_query($dbhandle, 'SELECT name, email FROM users LIMIT 25');
while ($entry = sqlite_fetch_array($query, SQLITE_ASSOC)) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

Example#2 オブジェクト指向言語型スタイルでの例

```
<?php
$dbhandle = new SQLiteDatabase('sqlitedb');

$query = $dbhandle->query('SELECT name, email FROM users LIMIT 25'); // buffered result set
$query = $dbhandle->unbufferedQuery('SELECT name, email FROM users LIMIT 25'); // unbuffered result set

while ($entry = $query->fetch(SQLITE_ASSOC)) {
    echo 'Name: ' . $entry['name'] . '  E-mail: ' . $entry['email'];
}
?>
```

参考

- sqlite_array_query()
- sqlite_fetch_string()

---

sqlite_fetch_column_types

SQLiteDatabase->fetchColumnTypes

(No version information available, might be only in CVS)

SQLiteDatabase->fetchColumnTypes — 特定のテーブルからカラム型の配列を返す

説明

*array **sqlite_fetch_column_types** (string $table_name , resource $dbhandle [, int $result_type])*

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
*array **fetchColumnTypes** (string $table_name [, int $result_type])*

***sqlite_fetch_column_types()** は、 指定されたテーブル table_name からカラムのデータ型の配列を返します。*

パラメータ

table_name

問い合わせるテーブル名

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

result_type

オプションパラメータ result_type は定数を受け付け、 返される配列をどの様にインデックス付けするかを決定します。 SQLITE_ASSOC を使用すると連想インデックス (名前付けられたフィールド) のみを返し、 SQLITE_NUM の場合は数値インデックス (順序を表すフィールド番号) のみを返します。 SQLITE_BOTH は、連想、 数値の両インデックスを返します。 SQLITE_ASSOC がこの関数のデフォルトです。

返り値

カラムのデータ型の配列を返します。エラー時は **FALSE** を返します。

SQLITE_ASSOC および **SQLITE_BOTH** で返されるカラム名は、設定オプション sqlite.assoc_case の値に基づき、大文字小文字が変換されます。

変更履歴

バージョン	説明
5.1.0	*result_type* が追加されました。

例

Example#1 手続き型言語スタイルでの例

```
<?php
$db = sqlite_open('mysqlitedb');
sqlite_query($db, 'CREATE TABLE foo (bar varchar(10), arf text)');
$cols = sqlite_fetch_column_types('foo', $db, SQLITE_ASSOC);

foreach ($cols as $column => $type) {
    echo "Column: $column   Type: $type";
}
?>
```

Example#2 オブジェクト指向言語型スタイルでの例

```
<?php
$db = new SQLiteDatabase('mysqlitedb');
$db->query('CREATE TABLE foo (bar varchar(10), arf text)');
$cols = $db->fetchColumnTypes('foo', SQLITE_ASSOC);

foreach ($cols as $column => $type) {
    echo "Column: $column   Type: $type";
}
?>
```

上の例の出力は以下となります。

```
Column: bar   Type: VARCHAR
Column: arf   Type: TEXT
```

---

sqlite_fetch_object

SQLiteResult->fetchObject

SQLiteUnbuffered->fetchObject

(No version information available, might be only in CVS)

SQLiteUnbuffered->fetchObject — 結果セットから次のレコードをオブジェクトとして取得する

説明

object **sqlite_fetch_object** (resource $result [, string $class_name [, array $ctor_params [, bool $decode_binary]]])

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
object **fetchObject** ([string $class_name [, array $ctor_params [, bool $decode_binary]]])
SQLiteUnbuffered
object **fetchObject** ([string $class_name [, array $ctor_params [, bool $decode_binary]]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

sqlite_fetch_single

SQLiteResult->fetchSingle

SQLiteUnbuffered->fetchSingle

(No version information available, might be only in CVS)

SQLiteUnbuffered->fetchSingle — 結果セットの最初のカラムを文字列として取得する

説明

string ***sqlite_fetch_single*** *(resource $result [, bool $decode_binary])*

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
string ***fetchSingle*** *([bool $decode_binary])*
SQLiteUnbuffered
string ***fetchSingle*** *([bool $decode_binary])*

sqlite_fetch_single() *は、 レコードセットの最初のカラムの値を返すことを除いて sqlite_fetch_array() と等価です。*

この関数は、データの単一カラムの値を確認するだけの場合に データを処理する最善の方法です。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

decode_binary

decode_binary パラメータが ***TRUE*** *(デフォルト)に設定された場合、PHP はバイナリエンコーディングをデコードします。これは、sqlite_escape_string() によりエンコードされたデータに適用されます。sqlite をサポートする他のアプリケーションにより作成されたデータベースを処理する時以外は、この値をデフォルトのままにしておくべきです。*

例

Example#1 sqlite_fetch_single() の例

```
<?php
if ($dbhandle = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {

    $sql = "SELECT id FROM sometable WHERE id = 42";
    $res = sqlite_query($dbhandle, $sql);

    if (sqlite_num_rows($res) > 0) {
        echo sqlite_fetch_single($res); // 42
    }

    sqlite_close($dbhandle);
}
?>
```

参考

- *sqlite_fetch_array()*

---

sqlite_fetch_string

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_fetch_string — sqlite_fetch_single() のエイリアス

説明

この関数は次の関数のエイリアスです。 sqlite_fetch_single().

---

sqlite_field_name

SQLiteResult->fieldName

SQLiteUnbuffered->fieldName

(No version information available, might be only in CVS)

SQLiteUnbuffered->fieldName — 特定のフィールドの名前を返す

説明

string ***sqlite_field_name*** *(resource $result , int $field_index)*

Object oriented style (method):

SQLiteResult
string ***fieldName*** *(int $field_index)*
SQLiteUnbuffered
string ***fieldName*** *(int $field_index)*

*Given the ordinal column number, field_index , **sqlite_field_name()** returns the name of that field in the result set result .*

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

field_index

結果セットにおけるオリジナルのカラム番号

返り値

*与えられたオリジナルのカラム番号での SQLite 結果セット中のフィールド名を返します。エラーの場合は、**FALSE** を返します。*

***SQLITE_ASSOC** および **SQLITE_BOTH** で返されるカラム名は、設定オプション sqlite.assoc_case の値に基づき、大文字小文字が変換されます。*

---

sqlite_has_more

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_has_more — まだレコードがあるかないかを返す

説明

*bool **sqlite_has_more** (resource $result)*

与えられた結果セット中にまだレコードがあるかどうかを見つけます。

パラメータ

result

SQLite 結果リソース

返り値

*result ハンドルにまだレコードがある場合 **TRUE** を返します。そうでない場合 **FALSE** を返します。*

参考

- *sqlite_num_rows()*
- *sqlite_changes()*

---

sqlite_has_prev

SQLiteResult->hasPrev

(No version information available, might be only in CVS)

SQLiteResult->hasPrev — 前のレコードがあるかどうかを返す

説明

*bool **sqlite_has_prev** (resource $result)*

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
*bool **hasPrev** (void)*

与えられた結果ハンドルに前のレコードがあるかどうかを検査します。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

> ***注意**: この関数は、バッファなしの結果ハンドルで使用することはできません。*

返り値

*結果ハンドル result に前のレコードがある場合 **TRUE**、 そうでない場合 **FALSE** を返します。*

参考

- *sqlite_prev()*
- *sqlite_has_more()*
- *sqlite_num_rows()*

---

sqlite_key

SQLiteResult->key

(No version information available, might be only in CVS)

SQLiteResult->key — カレントレコードのインデックスを返す

説明

int **sqlite_key** (resource $result)

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
int **key** (void)

sqlite_key() は、バッファされた結果セット result のカレントレコードのインデックスを返します。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

注意: この関数は、バッファなしの結果ハンドルで使用することはできません。

返り値

バッファされた結果セット result のカレントレコードのインデックスを返します。

変更履歴

バージョン	説明
5.0.4	PHP 5.0.4 以降、**sqlite_key()** は SQLiteResult オブジェクトのメソッドとしてコールする場合のみ利用可能です。 手続き型としてはコールできません。

参考

- sqlite_next()
- sqlite_current()
- sqlite_rewind()

---

sqlite_last_error

SQLiteDatabase->lastError

(No version information available, might be only in CVS)

SQLiteDatabase->lastError — データベースに関する直近のエラーコードを返す

説明

int **sqlite_last_error** (resource $dbhandle)

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
int **lastError** (void)

dbhandle (データベースハンドル)、 上で実行された直近の処理のエラーコード、 あるいはエラーが発生しなかった場合に 0 を返します。人が読み取れるエラーコードの詳細は、 sqlite_error_string() で取得可能です。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

参考

- sqlite_error_string()

---

sqlite_last_insert_rowid

SQLiteDatabase->lastInsertRowid

(No version information available, might be only in CVS)

SQLiteDatabase->lastInsertRowid — 直近に挿入されたレコードのrowidを返す

説明

int **sqlite_last_insert_rowid** (resource $dbhandle)

Object oriented style (method):

SQLiteDatabase
int **lastInsertRowid** (void)

データベースdbhandle に直近に挿入されたレコードの rowid を返します。ただし、この rowid が auto-increment フィールドとして作成されている場合に限ります。

ヒント

テーブルスキーマで INTEGER PRIMARY KEYと宣言することにより、 SQLiteでauto-incrementフィールドを作成することができます。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

---

sqlite_libencoding

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_libencoding — リンクされているSQLiteライブラリのエンコーディングを返す

説明

string **sqlite_libencoding** (void)

SQLiteライブラリは、ISO-8859-1またはUTF-8互換モードのどちらかでコン パイルすることができます。この関数により、使用するライブラリのエン コーディングを特定することが可能です。

警告

デフォルトのPHPのディストリビューションでは、libsqliteをISO-8859-1 エンコーディングモードで構築します。しかし、この名前は誤りです。ISO-8859-1を処理するというのではなく、このモードは文字列の比較やソー トに使用するカレントのロケールを使用します。 このため、ISO-8859-1ではなく、'8ビット'とみなすべきです。

UTF-8サポートを指定してコンパイルした場合、sqliteは データ中のUTF-8 マルチバイト文字のエンコードおよびデコードを行います。 しかし、データ処理を完全に行うことはまだできず(例えば、正規化は行わ れません)、いくつかの比較処理は、まだ、正しく行うことができません。

警告

UTF-8サポートを指定してコンパイルされたバージョンのSQLiteライブラ リを組み込んでWebサーバ版のPHPを使用することは推奨されません。 これは、UTF-8エンコーディングで問題が検出された場合にlibsqliteがプ ロセスを強制終了するためです。

参考

- **sqlite_lib_version()**

---

sqlite_libversion

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_libversion — リンクされているSQLiteライブラリのバージョンを返す

説明

string **sqlite_libversion** (void)

リンクされているSQLiteライブラリのバージョンを文字列として返します。

参考

- sqlite_libencoding()

---

sqlite_next

SQLiteResult->next

SQLiteUnbuffered->next

(No version information available, might be only in CVS)

SQLiteUnbuffered->next — 次のレコード番号へシークする

説明

*bool **sqlite_next** (resource $result)*

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
*bool **next** (void)*
SQLiteUnbuffered
*bool **next** (void)*

***sqlite_next()**は、結果ハンドル result を次のレコードへ進めます。*

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

返り値

*成功時は **TRUE**、もしレコードがない場合は **FALSE** を返します。*

参考

- *sqlite_seek()*
- *sqlite_current()*
- *sqlite_rewind()*

---

sqlite_num_fields

SQLiteResult->numFields

SQLiteUnbuffered->numFields

(No version information available, might be only in CVS)

SQLiteUnbuffered->numFields — 結果セットのフィールド数を返す

説明

*int **sqlite_num_fields** (resource $result)*

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
*int **numFields** (void)*
SQLiteUnbuffered
*int **numFields** (void)*

結果セット result のフィールド数を返します。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

参考

- *sqlite_changes()*
- *sqlite_num_rows()*

---

sqlite_num_rows

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_num_rows — 結果セットのレコード数を返す

説明

*int **sqlite_num_rows** (resource $result)*

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
*int **numRows** (void)*

バッファされた結果セット result のレコード数を返します。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

注意: この関数は、バッファなしの結果ハンドルで使用することはできません。

例

Example#1 手続き型言語スタイルでの例

```
<?php
$db = sqlite_open('mysqlitedb');
$result = sqlite_query($db, "SELECT * FROM mytable WHERE name='John Doe'");
$rows = sqlite_num_rows($result);

echo "Number of rows: $rows";
?>
```

Example#2 オブジェクト指向言語型スタイルでの例

```
<?php
$db = new SQLiteDatabase('mysqlitedb');
$result = $db->query("SELECT * FROM mytable WHERE name='John Doe'");
$rows = $result->numRows();

echo "Number of rows: $rows";
?>
```

参考

- sqlite_changes()
- sqlite_query()
- sqlite_num_fields()

---

sqlite_open

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_open — SQLiteデータベースをオープンする。データベースが存在しない場合は作 成する

説明

resource **sqlite_open** (string $filename [, int $mode [, string &$error_message]])

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
__construct (string $filename [, int $mode [, string &$error_message]])

SQLite データベースをオープン、 もしくは存在しない場合データベースを生成します。

パラメータ

filename

SQLite データベースのファイル名。もしこのファイルが存在しない場合、 SQLite はファイルを生成しようとします。 データを挿入したり、データベーススキーマを変更、 もしくはデータベースが存在しない場合にデータベースを生成する場合、 PHP はファイルに対する書き込み権限を持っている必要があります。

mode

ファイルのモード。読み込み専用モードでデータベースをオープンするために 使用することを目的としています。現在、このパラメータは SQLite ライブラリに無視されます。このモードのデフォルト値は、 8 進数値 0666 で、これは推奨される値です。

error_message

参照として渡され、エラーが発生した場合に データベースがオープンできなかった原因を説明する 記述的なエラーメッセージを保持するために設定されます。

返り値

成功時にリソース (データベースハンドル)、失敗時に **FALSE** を返します。

例

Example#1 sqlite_open() の例

```
<?php
if ($db = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {
    sqlite_query($db, 'CREATE TABLE foo (bar varchar(10))');
    sqlite_query($db, "INSERT INTO foo VALUES ('fnord')");
    $result = sqlite_query($db, 'select bar from foo');
    var_dump(sqlite_fetch_array($result));
} else {
    die($sqliteerror);
}
```

```
?>
```

注意

ヒント

Unix環境では、SQLiteはシステムコールfork()を使用するスクリプトの影 響を受け易いです。このようなスクリプトがある場合、 子プロセスを生成する前にハンドルを閉じ、子プロセスまたは親プロセス から再度オープンすることが推奨されます。 この問題に関する詳細な情報については、 マルチスレッドおよびSQLiteという名前のセクショ ンにある » SQLiteライブラリへのC言語インターフェイス を参照してください。

ヒント

NFSパーティションにマウントされたSQLiteデータベースを処理すること は推奨されません。ロックに関してNFSは著しい問題があるので、 データベースを全くオープンすることさえできない可能性があります。ま た、成功した場合でも、ロックに関する動作は予測できない結果を生む可 能性があります。

> **注意**: SQLiteライブラリ バージョン2.8.2以降、 コンピュータのメモリ上にのみ存在するデータベースを作成するために filename に:memory:を指定 することができます。 これは、メモリ上のデータベースは処理完了時に破棄されるため、 テンポラリな処理を行う場合には有用です。他のデータベースをロードし、 データを相互に移動したりクエリを実行したりするために ATTACH DATABASE SQLステートメントと組み合わせる 場合にも有用です。
>
> **注意**: SQLite は セーフモード および open_basedir に対応しています。

参考

- sqlite_popen()
- sqlite_close()
- sqlite_factory()

---

sqlite_popen

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_popen — SQLiteデータベースへの持続的ハンドルをオープンする。存在しない場合 には、データベースを作成する

説明

```
resource sqlite_popen ( string $filename [, int $mode [, string &$error_message ]] )
```

この関数はsqlite_open()と同じように動作しますが、 PHPの持続的リソース機能を使用するところが異なります。 パラメータの意味に関する詳細については、マニュアルの sqlite_open() を参照してください。

sqlite_popen() は、まず、持続的ハンドルが指定し たfilename に関してすでにオープンされているか どうかを調べます。みつかった場合には、スクリプトのそのハンドルを返 し、それ以外は、データベースのハンドルを新規にオープンします。

この手法の利点は、持続性のあるWebサーバ SAPI(通常のCGIまたはCLI以外 の全てのSAPI)により処理される各ページについてデータベースやインデッ クススキーマを再読み込みする性能上のコストを払うことがないことです。

> **注意**: 持続的ハンドルを使用している時にデータベースが(crontab等の)バック グラウンドプロセスにより更新され、このプロセスが上書きすることによ り(消去して、再構築するか、カレントのバージョンを置換するために更 新後のバージョンを移動)データベースを再生成する場合、 古いバージョンのデータベースに関する持続的ハンドルが再利用されると いった予測できない動作を引き起こす可能性があります。 この問題を回避するために、バックグラウンドプロセスが同じデータベー スファイルをオープンするようにし、更新をトランザクションで行うよう にしてください。

パラメータ

filename

SQLite データベースのファイル名。もしこのファイルが存在しない場合、 SQLite はファイルを生成しようとします。 データを挿入したり、データベーススキーマを変更、 もしくはデータベースが存在しない場合にデータベースを生成する場合、 PHP はファイルに対する書き込み権限を持っている必要があります。

mode

ファイルのモード。読み込み専用モードでデータベースをオープンするために 使用することを目的としています。現在、このパラメータは SQLite ライブラリに無視されます。このモードのデフォルト値は、 8 進数値 0666 で、これは推奨される値です。

error_message

参照として渡され、エラーが発生した場合に データベースがオープンできなかった原因を説明する 記述的なエラーメッセージを保持するために設定されます。

返り値

成功時にリソース (データベースハンドル)、失敗時に **FALSE** を返します。

参考

- sqlite_open()
- sqlite_close()
- sqlite_factory()

---

sqlite_prev

SQLiteResult->prev

(No version information available, might be only in CVS)

SQLiteResult->prev — 結果セットの前のレコード番号へシークする

説明

bool **sqlite_prev** (resource $result)

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
bool **prev** (void)

sqlite_prev() は、結果ハンドル result を前のレコードに戻します。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

注意: この関数は、バッファなしの結果ハンドルで使用することはできません。

返り値

成功時は **TRUE**、もしレコードがない場合は **FALSE** を返します。

参考

- sqlite_has_prev()
- sqlite_rewind()
- sqlite_next()

---

sqlite_query

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_query — 指定したデータベースに対してクエリを実行し、結果ハンドル を返す

説明

resource **sqlite_query** (resource $dbhandle , string $query [, int $result_type [, string &$error_msg]])
resource **sqlite_query** (string $query , resource $dbhandle [, int $result_type [, string &$error_msg]])

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
SQLiteResult **query** (string $query [, int $result_type [, string &$error_msg]])

指定したデータベースにより query で指定した SQL ステートメントを実行します。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

query

実行するクエリ

result_type

オプションの result_type パラメータには定数を指定でき、返される配列の添字を定義します。 **SQLITE_ASSOC** を用いると、連想配列の添字(名前フィールド)のみが返されます。一方、**SQLITE_NUM** は、数値の添字(フィールド番号)のみを返します。**SQLITE_BOTH** は、連想配列の添字と数値の添字の両方を返します。 **SQLITE_BOTH** がこの関数のデフォルトです。

error_msg

もしエラーが発生した場合、ここに保存されます。SQL 構文のエラーは sqlite_last_error() 関数で取得することが できないため、このパラメータが特に重要となります。

注意: (MySQL のような)他のデータベースエクステンションとの互換性のため、 2 種類の構文がサポートされています。推奨されるのは最初の構文で、dbhandle パラメータを関数の最初のパラメータとするものです。

返り値

この関数は結果ハンドル、もしくは失敗時に **FALSE** を返します。 レコードを返すクエリの場合、結果ハンドルは sqlite_fetch_array() や sqlite_seek() のような関数で使用することができます。

クエリの型によらず、この関数はクエリが失敗した場合に**FALSE** を返し ます。

sqlite_query() は、バッファリングされ、シーク可 能な結果ハンドルを返します。これは、レコードにランダムにアクセスす る必要がある小さなクエリの場合に有用です。バッファリングされた結果 ハンドルは、結果全体を保持するためのメモリを確保し、結果が取得され るまでは値を返しません。 データに連続的にアクセスしたい場合、かわりにより高性能な sqlite_unbuffered_query() を使用することが 推奨されます。

変更履歴

バージョン	説明
5.1.0	*error_msg* パラメータが追加されました。

注意

警告

SQLiteは、セミコロンで区切られた複数のクエリを実行します。 これにより、ファイルからロードするかスクリプトに埋め込んだ SQL をバッチ実行することができます。 しかしながら、これは関数の結果が使用されない場合のみ動作します。 使用されない場合、最初の SQL ステートメントのみ実行されます。 関数 sqlite_exec() は常に複数の SQL ステートメントを実行します。

*複数のクエリを実行する際、この関数の返り値は、エラーの場合に **FALSE**となります。しかし、それ以外の場合は不定となります。（成功した場合に **TRUE**となるか、結果ハンドルを返す可能性があります）*

参考

- *sqlite_unbuffered_query()*
- *sqlite_array_query()*

---

sqlite_rewind

SQLiteResult->rewind

(No version information available, might be only in CVS)

SQLiteResult->rewind — 先頭レコード番号へシークする

説明

*bool **sqlite_rewind** (resource $result)*

オブジェクト指向言語型スタイル（メソッド）:

SQLiteResult
*bool **rewind** (void)*

***sqlite_rewind()**は、 結果セットの先頭レコードへシークします。*

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

> ***注意**: この関数は、バッファなしの結果ハンドルで使用することはできません。*

返り値

*結果セットにレコードがもうない場合は、**FALSE**、 そうでない場合は **TRUE** を返します。*

参考

- *sqlite_next()*
- *sqlite_current()*
- *sqlite_seek()*

---

sqlite_seek

SQLiteResult->seek

(No version information available, might be only in CVS)

SQLiteResult->seek — 特定のレコード番号へシークする

説明

*bool **sqlite_seek** (resource $result , int $rownum)*

オブジェクト指向言語型スタイル（メソッド）:

SQLiteResult
*bool **seek** (int $rownum)*

***sqlite_seek()**は、パラメータ rownum で指定したレコードにシークします。*

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

> ***注意**: この関数は、バッファなしの結果ハンドルで使用することはできません。*

rownum

シークするオリジナルのレコード番号。 レコード番号は 0 から始まります (0 が先頭レコード) 。

> ***注意**: この関数は、バッファなしの結果ハンドルで使用することはできません。*

返り値

*レコードが存在しない場合は **FALSE**、それ以外の場合に **TRUE** を返します。*

参考

- *sqlite_next()*
- *sqlite_current()*
- *sqlite_rewind()*

---

sqlite_single_query

SQLiteDatabase->singleQuery

(No version information available, might be only in CVS)

SQLiteDatabase->singleQuery — クエリを実行し、単一カラムもしくは先頭レコードの値に対する配列を返す

説明

*array **sqlite_single_query** (resource $db , string $query [, bool $first_row_only [, bool $decode_binary]])*

オブジェクト指向言語型スタイル (メソッド):

SQLiteDatabase
*array **singleQuery** (string $query [, bool $first_row_only [, bool $decode_binary]])*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

sqlite_udf_decode_binary

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_udf_decode_binary — UDFにパラメータとして渡されたバイナリデータをデコードする

説明

*string **sqlite_udf_decode_binary** (string $data)*

***sqlite_udf_decode_binary()** は、 sqlite_udf_encode_binary() または sqlite_escape_string()によりパラメータに適用され たバイナリエンコーディングをデコードします。*

バイナリデータをUDFにより処理させる必要がある場合、UDFに渡されたパ ラメータに関してこの関数をコールする必要があります。 これは、PHPにより適用されたバイナリエンコーディングが内容と元のパラ メータを隠蔽するためです。

PHP は、自動的にエンコード/デコード処理を行いません。これは、これを 行うと著しい性能劣化を生じる可能性があるためです。

例

Example#1 バイナリ対応 max_length 集約関数の例

```
<?php
$data = array(
    'one',
    'two',
    'three',
    'four',
    'five',
    'six',
    'seven',
    'eight',
    'nine',
    'ten',
    );
$db = sqlite_open(':memory:');
sqlite_query($db, "CREATE TABLE strings(a)");
foreach ($data as $str) {
    $str = sqlite_escape_string($str);
    sqlite_query($db, "INSERT INTO strings VALUES ('$str')");
}

function max_len_step(&$context, $string)
{
```

```
    $string = sqlite_udf_decode_binary($string);
    if (strlen($string) > $context) {
        $context = strlen($string);
    }
}

function max_len_finalize(&$context)
{
    return $context;
}

sqlite_create_aggregate($db, 'max_len', 'max_len_step', 'max_len_finalize');

var_dump(sqlite_array_query($db, 'SELECT max_len(a) from strings'));

?>
```

参考

- sqlite_udf_encode_binary()
- sqlite_create_function()
- sqlite_create_aggregate()

---

sqlite_udf_encode_binary

(PHP 5, PECL sqlite:1.0-1.0.3)

sqlite_udf_encode_binary — UDFから返す前にバイナリデータをエンコードする

説明

string **sqlite_udf_encode_binary** (string $data)

sqlite_udf_encode_binary() は、 (元のlibsqlite APIはバイナリ対応ではないため、) クエリから安全に値を返せるようにdata にバイナリエンコーディングを適用します。

バイナリセーフでないと思われるデータ(例: ヌルバイトを末尾以外の場所 に含むデータや先頭文字に0x01を含むデータ)の場合、 UDFからの返り値をエンコードするために、この関数をコールする必要があ ります。

PHPは、このエンコード/デコード処理を自動的に行いません。 これは、性能上著しい影響があるためです。

> **注意**: UDFから返す文字列をクオートするために sqlite_escape_string()を使用しないでください。 代わりにこの関数を使用してください。

参考

- sqlite_udf_decode_binary()
- sqlite_escape_string()
- sqlite_create_function()
- sqlite_create_aggregate()

---

sqlite_unbuffered_query

SQLiteDatabase->unbufferedQuery

(No version information available, might be only in CVS)

SQLiteDatabase->unbufferedQuery — 事前取得していないクエリを実行し、全てのデータをバッファリングする

説明

```
resource sqlite_unbuffered_query ( resource $dbhandle , string $query [, int $result_type [, string &$error_msg ]] )
resource sqlite_unbuffered_query ( string $query , resource $dbhandle [, int $result_type [, string &$error_msg ]] )
```

オブジェクト指向言語型スタイル (メソッド):

```
SQLiteDatabase
SQLiteUnbuffered bufferedQuery ( string $query [, int $result_type [, string &$error_msg ]] )
```

sqlite_unbuffered_query() は sqlite_query() と同じですが、 連続的に前方のみにアクセス可能な結果セットが返され、 各レコードを 1 件ずつ読み込むことだけしかできないところが異なります。

この関数は、一度に 1 件ずつレコードを処理するだけでよく、ランダムに データをアクセスする必要がない、HTMLテーブルのようなものを生成するのに適しています。

> **注意**: sqlite_seek(), sqlite_rewind(), sqlite_next(), sqlite_current() および sqlite_num_rows() のような関数は、この関数から返 された結果ハンドルでは動作しません。

パラメータ

dbhandle

SQLite データベースリソース。手続きに従って、 sqlite_open() から返されます。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

query

実行するクエリ

result_type

*オプションの result_type パラメータには定数を指定でき、返される配列の添字を定義します。 **SQLITE_ASSOC** を用いると、連想配列の添字(名前フィールド)のみが返されます。一方、**SQLITE_NUM** は、数値の添字(フィールド番号)のみを返します。**SQLITE_BOTH** は、連想配列の添字と数値の添字の両方を返します。 **SQLITE_BOTH** がこの関数のデフォルトです。*

error_msg

もしエラーが発生した場合、ここに保存されます。SQL 構文のエラーは sqlite_last_error() 関数で取得することが できないため、このパラメータが特に重要となります。

> ***注意**: (MySQL のような)他のデータベースエクステンションとの互換性のため、 2 種類の構文がサポートされています。推奨されるのは最初の構文で、dbhandle パラメータを関数の最初のパラメータとするものです。*

返り値

*結果セットもしくは失敗時に **FALSE** を返します。*

***sqlite_unbuffered_query()** は、 各レコードを順番に読み込む場合にのみ利用可能な シーケンシャルで前進のみ可能な結果セットを返します。*

変更履歴

バージョン	説明
5.1.0	*error_msg* パラメータが追加されました。

参考

- *sqlite_query()*

---

sqlite_valid

SQLiteResult->valid

SQLiteUnbuffered->valid

(No version information available, might be only in CVS)

SQLiteUnbuffered->valid — まだレコードが残っているかどうかを返す

説明

*bool **sqlite_valid** (resource $result)*

オブジェクト指向言語型スタイル (メソッド):

SQLiteResult
*bool **valid** (void)*
SQLiteUnbuffered
*bool **valid** (void)*

与えられた結果ハンドルにまだレコードが残っているかを検査します。

パラメータ

result

SQLite 結果リソース。 このパラメータは、 オブジェクト指向言語型メソッドを使用する場合は不要です。

> ***注意**: この関数は、バッファなしの結果ハンドルで使用することはできません。*

返り値

*結果ハンドル result にレコードが残っている場合 **TRUE**、そうでない場合は **FALSE** を返します。*

参考

- *sqlite_num_rows()*
- *sqlite_changes()*

目次

- sqlite_column — カレントの結果セットのレコードからカラムを1列取得する
- sqlite_create_aggregate — SQLステートメントで使用する集約UDFを登録する
- sqlite_create_function — SQLステートメントで使用するために"通常の"ユーザ定義関数を登録する
- sqlite_current — 結果セットからカレントのレコードを配列として取得する
- sqlite_error_string — エラーコードの説明を返す
- sqlite_escape_string — クエリパラメータ用に文字列をエスケープする
- sqlite_exec — 与えられたデータベースに対して結果を伴わないクエリを実行する
- sqlite_factory — SQLite データベースをオープンし、SQLiteDatabse オブジェクトを返す
- sqlite_fetch_all — 結果セットから全てのレコードを配列の配列として取得する
- sqlite_fetch_array — 結果セットから次のレコードを配列として取得する
- sqlite_fetch_column_types — 特定のテーブルからカラム型の配列を返す
- sqlite_fetch_object — 結果セットから次のレコードをオブジェクトとして取得する
- sqlite_fetch_single — 結果セットの最初のカラムを文字列として取得する
- sqlite_fetch_string — sqlite_fetch_single のエイリアス
- sqlite_field_name — 特定のフィールドの名前を返す
- sqlite_has_more — まだレコードがあるかないかを返す
- sqlite_has_prev — 前のレコードがあるかどうかを返す
- sqlite_key — カレントレコードのインデックスを返す
- sqlite_last_error — データベースに関する直近のエラーコードを返す
- sqlite_last_insert_rowid — 直近に挿入されたレコードのrowidを返す
- sqlite_libencoding — リンクされているSQLiteライブラリのエンコーディングを返す
- sqlite_libversion — リンクされているSQLiteライブラリのバージョンを返す
- sqlite_next — 次のレコード番号へシークする
- sqlite_num_fields — 結果セットのフィールド数を返す
- sqlite_num_rows — 結果セットのレコード数を返す
- sqlite_open — SQLiteデータベースをオープンする。データベースが存在しない場合は作 成する
- sqlite_popen — SQLiteデータベースへの持続的ハンドルをオープンする。存在しない場合 には、データベースを作成する
- sqlite_prev — 結果セットの前のレコード番号へシークする
- sqlite_query — 指定したデータベースに対してクエリを実行し、結果ハンドル を返す
- sqlite_rewind — 先頭レコード番号へシークする
- sqlite_seek — 特定のレコード番号へシークする
- sqlite_single_query — クエリを実行し、単一カラムもしくは先頭レコードの値に対する配列を返す
- sqlite_udf_decode_binary — UDFにパラメータとして渡されたバイナリデータをデコードする
- sqlite_udf_encode_binary — UDFから返す前にバイナリデータをエンコードする
- sqlite_unbuffered_query — 事前取得していないクエリを実行し、全てのデータをバッファリングする
- sqlite_valid — まだレコードが残っているかどうかを返す

SQLite 関数 (PDO_SQLITE)

導入

PDO_SQLITE は、PHP から SQLite 2 や SQLite 3 データベースへのアクセスを可能にするための PHP Data Objects (PDO) インターフェース を実装したドライバです。

PHP 5.1 では、SQLite 拡張モジュールも SQLite 2 データベースに対するドライバを提供しています。 理論的には PDO_SQLITE ドライバの一部ではなく動作も同様なので、 平行してドキュメント化されています。PDO 用 SQLite 2 ドライバは主に レガシーな SQLite 2 データベースファイルをより高速でより効果的な SQLite 3 ドライバを使用するアプリケーションへ 簡単にインポートするために提供されています。 結果として、SQLite 2 ドライバは SQLite 3 ドライバよりも機能豊富ではありません。

PDO_SQLITE DSN

(No version information available, might be only in CVS)

PDO_SQLITE DSN — SQLite データベースに接続する

説明

PDO_SQLITE データソース名 (DSN) は以下の要素で構成されます:

DSN 接頭辞 (SQLite 3)

DSN 接頭辞は **sqlite:** です。

- ディスク上のデータベースにアクセスするには、 DSN 接頭辞に絶対パスを付加してください。
- メモリ内にデータベースを生成するには、 DSN 接頭辞に :memory: を付加してください。

DSN 接頭辞 (SQLite 2)

PHP 5.1 における SQLite 拡張モジュールは SQLite 2 データベースへのアクセスと生成機能を サポートする PDO ドライバを提供しています。これにより、PHP の以前のバージョンにおける SQLite 拡張を用いて生成したデータベースにアクセスすることが可能です。

注意: SQLite 2 ドライバは PDO と ext/sqlite を有効にした場合、 PHP 5.1.x でのみ利用可能です。現時点では PECL 経由では利用できません。

SQLite 2 データベースに接続するための DSN 接頭辞は **sqlite2:** です。

- ディスク上のデータベースにアクセスするには、 DSN 接頭辞に絶対パスを付加してください。
- メモリ内にデータベースを生成するには、 DSN 接頭辞に memory: を付加してください。

例

Example#1 PDO_SQLITE DSN の例

以下の例は、SQLite データベースへの接続するための PDO_SQLITE DSN を表します:

```
sqlite:/opt/databases/mydb.sq3
sqlite::memory:
sqlite2:/opt/databases/mydb.sq2
sqlite2::memory:
```

PDO->sqliteCreateAggregate()

(No version information available, might be only in CVS)

PDO->sqliteCreateAggregate() — SQL 文で使用する集約ユーザ定義関数 (UDF) を登録する

説明

PDO
bool **sqliteCreateAggregate** (string $function_name , callback $step_func , callback $finalize_func [, int $num_args])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドは PDO->sqliteCreateFunction() と似ていますが、 この関数で登録した関数は、クエリのすべての行の内容を集約する関数を登録します。

この関数と PDO->sqliteCreateFunction() の最大の違いは、 集約関数を作成するためには 2 つの関数が必要であるということです。

パラメータ

function_name

SQL 文で使用する関数の名前。

step_func

結果セットの各行についてコールされるコールバック関数。 この PHP 関数は、結果を蓄積して集約コンテキストに保存しなければなりません。

この関数は次のように定義しなければなりません。

step (mixed $context , int $rownumber , mixed $value1 [, mixed $value2 [, mixed $..]])

context は最初の行では **NULL** となります。 それ以降の行では、その前の step 関数が返した値を保持します。 これを使用して、集約の状態を管理します。

rownumber は現在の行番号を保持します。

finalize_func

すべての行が処理された後でコールされるコールバック関数。 ここでは、集約コンテキストからデータを取得して結果を返します。 コールバック関数の返す値は、SQLite が理解できる形式 (すなわち スカラー型) でなければなりません。

この関数は次のように定義しなければなりません。

fini (mixed $context , int $rownumber)

context には、最後の step 関数の返り値が格納されます。

rownumber は、この集約関数が処理した行数を保持します。

この関数の返り値が、集約の返り値となります。

num_args

コールバック関数があらかじめ定義済みの引数を受け取る場合に、 SQLite のパーサに渡すヒント。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 集約関数 max_length の例

```
<?php
$data = array(
    'one',
    'two',
    'three',
    'four',
```

```
    'five',
    'six',
    'seven',
    'eight',
    'nine',
    'ten',
    );
$db = new PDO('sqlite::memory:');
$db->exec("CREATE TABLE strings(a)");
$insert = $db->prepare('INSERT INTO strings VALUES (?)');
foreach ($data as $str) {
    $insert->execute(array($str));
}
$insert = null;

function max_len_step(&$context, $rownumber, $string)
{
    if (strlen($string) > $context) {
        $context = strlen($string);
    }
}

function max_len_finalize(&$context, $rownumber)
{
    return $context;
}

$db->sqliteCreateAggregate('max_len', 'max_len_step', 'max_len_finalize');

var_dump($db->query('SELECT max_len(a) from strings')->fetchAll());

?>
```

この例では、 テーブルのカラムの中で一番長い文字列の長さを計算する集約関数を作成します。 各行について max_len_step 関数がコールされ、 context パラメータが渡されます。 このパラメータには、他の PHP 変数と同様に、配列やオブジェクトが設定されます。 この例では、これまでに登場した値のうち長さが最大のものの長さを保持しています。 string が現在の最大値より長い場合に、 その値で現在の最大値を更新します。

すべての行に対する処理が終わると、SQLite は max_len_finalize 関数をコールして集約結果を決定します。 ここでは、context の内容に基づいた、 なんらかの計算を行うことができます。 しかし、この例ではクエリを処理している過程で既に結果が決定しているので、 ここでは単に context の値を返しているだけです。

ヒント

結果の値を context に溜め込んでおき、最後に一括して処理するという方法は推奨 「しません」。これは、SQLite のメモリ消費量が大きくなるからです。 仮に 32 バイトの長さのデータが百万件あったとして、 それを溜め込むためにどれだけのメモリが必要になるか考えてみましょう。

ヒント

PDO->sqliteCreateFunction() および PDO->sqliteCreateAggregate() を使用して、 SQLite のネイティブ SQL 関数を上書きすることができます。

注意: このメソッドは、SQLite2 ドライバでは使用できません。 代わりに、古い形式の sqlite API を使用してください。

参考

- PDO->sqliteCreateFunction()
- sqlite_create_function()
- sqlite_create_aggregate()

PDO->sqliteCreateFunction()

(No version information available, might be only in CVS)

PDO->sqliteCreateFunction() — SQL 文で使用するユーザ定義関数 (UDF) を登録する

説明

PDO
bool **sqliteCreateFunction** (string $function_name , callback $callback [, int $num_args])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

このメソッドを使用すると、 PHP の関数を、UDF (User Defined Function: ユーザ定義関数) として SQLite に登録できるようになります。 登録した関数は、SQL 文の中で使用することが可能です。

UDF は、関数をコールできるあらゆる SQL 文 (SELECT および UPDATE 文、そしてトリガなど) で使用することが可能です。

パラメータ

function_name

SQL 文で使用する関数の名前。

callback

定義した SQL 関数を処理するコールバック関数。

注意: コールバック関数の返す値は、SQLite が理解できる形式 (すなわち スカラー型) でなければなりません。

num_args

コールバック関数があらかじめ定義済みの引数を受け取る場合に、 SQLite のパーサに渡すヒント。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 PDO::sqliteCreateFunction() の例

```
<?php
function md5_and_reverse($string)
{
    return strrev(md5($string));
}

$db = new PDO('sqlite:sqlitedb');
$db->sqliteCreateFunction('md5rev', 'md5_and_reverse', 1);
$rows = $db->query('SELECT md5rev(filename) FROM files')->fetchAll();
?>
```

この例では、文字列の md5 を計算してそれを反転させる関数を定義しています。 SQL 文が実行されると、filename をこの関数で変換した値が返されます。 $rows に返されるデータの中に、処理結果が含まれます。

この方法を使用すると、データを取得した後に foreach() で結果をひとつひとつ処理していく必要がなくなり、美しいコードが記述できます。

ヒント

PDO->sqliteCreateFunction() および PDO->sqliteCreateAggregate() を使用して、 SQLite のネイティブ SQL 関数を上書きすることができます。

> **注意**: このメソッドは、SQLite2 ドライバでは使用できません。 代わりに、古い形式の sqlite API を使用してください。

参考

- PDO->sqliteCreateAggregate()
- sqlite_create_function()
- sqlite_create_aggregate()

目次

Secure Shell2 関数

導入

安全な暗号化通信を使用したリモートマシン上にあるリソースへのアクセス手段（シェル、リモート実行、トンネリング、ファイル転送）を提供する » libssh2 のバインディングです。

インストール手順

Windows 用バイナリは » http://snaps.php.net/ にあります。 インストールするには php_ssh2.dll を php.ini ファイルの extension_dir ディレクティブで指定されたフォルダに ダウンロードしてください。そして php.ini に extension=php_ssh2.dll を追加し、Web サーバを再起動して有効にしてください。

```
extension_dir=c:/php5/exts/
extension=php_ssh2.dll
```

Linux, BSD, あるいは他の *nix 系 では以下の手順でコンパイルすることができます:

- » OpenSSL をダウンロードしインストールします。 もしディストリビューションのパッケージングシステム経由で OpenSSL をインストールする場合、開発用ライブラリもインストールしてください。 これは典型的には openssl-dev や openssl_devel、 もしくはいくつかのバリエーションで名前付けされたパッケージです。
- » libssh2 をダウンロードしインストールします。典型的には libssh2 のソースツリーから次のコマンドを実行することを意味します。
  ```
  ./configure && make all install
  ```
- PECL/ssh2 のために PEAR インストーラを実行します: pear install ssh2
- ビルドプロセスが示したディレクトリから php.ini ファイルの extension_dir で指定された場所に ssh2.so をコピーします。
- php.ini に extension=ssh2.so を追加します。
- php.ini の設定を再読込させるため、Web サーバを再起動します。

> **注意**: **開発バージョン** 現時点では PECL/ssh2 の 安定 バージョンはありません。 PECL/ssh2 の ベータ バージョンをインストールするには次を実行してください: pear install ssh2-beta

ヒント

PEAR コマンドを使用しないで PECL/ssh2 をコンパイルする

自動的に PECL/ssh2 をダウンロード、インストールする pear install ssh2 を使用するよりもむしろ » PECL から tarball をダウンロードしても良いです。 展開した tarball のルートから ssh2.so を生成するには次を実行します: phpize && ./configure --with-ssh2 && make 生成した後、上記のステップ 4 からインストールを続けてください。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/ssh2.

> **注意**: libssh2 ライブラリのバージョン 0.4 もしくはそれ以降が必要です (ひょっとするともっと新しいかも知れませんので、 リリースノートを参照してください) 。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

SSH2_FINGERPRINT_MD5 (integer)
ホスト鍵のフィンガープリントを MD5 ハッシュとして要求する ssh2_fingerprint() 用フラグ
SSH2_FINGERPRINT_SHA1 (integer)
ホスト鍵のフィンガープリントを SHA1 ハッシュとして要求する ssh2_fingerprint() 用フラグ
SSH2_FINGERPRINT_HEX (integer)
ホスト鍵のフィンガープリントを 16 進法の文字列として要求する ssh2_fingerprint() 用フラグ
SSH2_FINGERPRINT_RAW (integer)
ホスト鍵のフィンガープリントを 8 ビットキャラクタの文字列として要求する ssh2_fingerprint() 用フラグ
SSH2_TERM_UNIT_CHARS (integer)
文字単位で 幅 と 高さ を指定するための ssh2_shell() 用フラグ
SSH2_TERM_UNIT_PIXELS (integer)
ピクセル単位で 幅 と 高さ を指定するための ssh2_shell() 用フラグ
SSH2_DEFAULT_TERM_WIDTH (integer)
ssh2_shell() で要求されるデフォルトのターミナル幅
SSH2_DEFAULT_TERM_HEIGHT (integer)
ssh2_shell() で要求されるデフォルトのターミナル高さ
SSH2_DEFAULT_TERM_UNIT (integer)
ssh2_shell() で要求されるデフォルトのターミナル単位
SSH2_STREAM_STDIO (integer)
STDIO サブチャネルを要求する ssh2_fetch_stream() 用フラグ
SSH2_STREAM_STDERR (integer)
STDERR サブチャネルを要求する ssh2_fetch_stream() 用フラグ
SSH2_DEFAULT_TERMINAL (string)
ssh2_shell() によって要求される デフォルトのターミナル型 (例えば vt102, ansi, xterm, vanilla)

---

ssh2_auth_hostbased_file

(PECL ssh2:0.10-0.9)

ssh2_auth_hostbased_file — ホスト公開鍵を使用して認証を行う

説明

bool **ssh2_auth_hostbased_file** (resource $session , string $username , string $hostname , string $pubkeyfile , string $privkeyfile [, string $passphrase [, string $local_username]])

ファイルから読み込まれたホスト公開鍵を使用して認証を行います。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

username

hostname

pubkeyfile

privkeyfile

passphrase

もし privkeyfile が暗号化されている (そのはずです) 場合、パスフレーズを渡す必要があります。

local_username

もし local_username を省略した場合、 username の値を使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ホスト公開鍵を使用した認証

```
<?php
$connection = ssh2_connect('shell.example.com', 22, array('hostkey'=>'ssh-rsa'));

if (ssh2_auth_hostbased_file($connection, 'remoteusername', 'myhost.example.com',
                             '/usr/local/etc/hostkey_rsa.pub',
```

```
                            '/usr/local/etc/hostkey_rsa', 'secret',
                            'localusername')) {
  echo "Public Key Hostbased Authentication Successful¥n";
} else {
  die('Public Key Hostbased Authentication Failed');
}
?>
```

注意

注意: **ssh2_auth_hostbased_file()** には libssh2 >= 0.7 と PHP/SSH2 >= 0.7 が必要です。

---

ssh2_auth_none

(PECL ssh2:0.10-0.9)

ssh2_auth_none — "none" として認証する

説明

mixed **ssh2_auth_none** (resource $session , string $username)

通常失敗する（そしてそうあるべき）"none" 認証を試みます。 失敗の一環として、サーバは可能な認証メソッドの一覧を返します。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

username

リモートのユーザ名。

返り値

もしサーバが username に対する認証メソッドとして "none" を受け入れる場合、この関数は単純に **TRUE** を返します。

例

Example#1 認証メソッドの一覧の取得

```
<?php
$connection = ssh2_connect('shell.example.com', 22);

$auth_methods = ssh2_auth_none($connection, 'user');

if (in_array('password', $auth_methods)) {
  echo "Server supports password based authentication¥n";
}
?>
```

---

ssh2_auth_password

(PECL ssh2:0.10-0.9)

ssh2_auth_password — SSH 上でプレーンなパスワードを使用した認証を行う

説明

bool **ssh2_auth_password** (resource $session , string $username , string $password)

SSH 上でプレーンなパスワードを使用した認証を行います。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

username

リモートのユーザ名。

password

username のパスワード。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 パスワードを用いた認証

```
<?php
$connection = ssh2_connect('shell.example.com', 22);

if (ssh2_auth_password($connection, 'username', 'secret')) {
  echo "Authentication Successful!¥n";
} else {
  die('Authentication Failed...');
}
?>
```

---

ssh2_auth_pubkey_file

(PECL ssh2:0.10-0.9)

ssh2_auth_pubkey_file — 公開鍵を使用した認証を行う

説明

bool **ssh2_auth_pubkey_file** (resource $session , string $username , string $pubkeyfile , string $privkeyfile [, string $passphrase])

ファイルから読み込んだ公開鍵を使用した認証を行います。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

username

pubkeyfile

privkeyfile

passphrase

もし、privkeyfile が暗号化されている（そうあるべきです）場合、パスフレーズも引数に渡す必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 公開鍵を用いた認証

```
<?php
$connection = ssh2_connect('shell.example.com', 22, array('hostkey'=>'ssh-rsa'));

if (ssh2_auth_pubkey_file($connection, 'username',
                          '/home/username/.ssh/id_rsa.pub',
                          '/home/username/.ssh/id_rsa', 'secret')) {
  echo "Public Key Authentication Successful¥n";
} else {
  die('Public Key Authentication Failed');
}
?>
```

---

ssh2_connect

(PECL ssh2:0.10-0.9)

ssh2_connect — SSH サーバに接続する

説明

resource **ssh2_connect** (string $host [, int $port [, array $methods [, array $callbacks]]])

リモートの SSH サーバとの接続を確立します。

一度接続すると、クライアントは ssh2_fingerprint() を使用してサーバのホスト鍵を検証し、 パスワードもしくは公開鍵を使用して認証します。

パラメータ

host

port

methods

methods は以下に示された4つのパラメータを持つ連想配列です。

methods は以下のパラメータのいくつかあるいは全てを含む連想配列

インデックス	意味	サポートする値*
kex	通知する鍵交換メソッドのリスト。優先する順にカンマ区切りにする。	*diffie-hellman-group1-sha1*、 *diffie-hellman-group14-sha1* および *diffie-hellman-group-exchange-sha1*
hostkey	通知するホスト鍵メソッドのリスト。優先する順にカンマ区切りにする。	*ssh-rsa* および *ssh-dss*
client_to_server	クライアントからサーバに送信されるメッセージのために優先する暗号化、 圧縮、メッセージ認証コード (MAC) メソッドを含む連想配列。	
server_to_client	サーバからクライアントに送信されるメッセージのために優先する暗号化、 圧縮、メッセージ認証コード (MAC) メソッドを含む連想配列。	

** - サポートする値は、 構成するライブラリがサポートしているメソッドに依存します。 追加情報については » libssh2 ドキュメントを参照ください。*

client_to_server と *server_to_client* は以下のパラメータのいくつかあるいは全てを含む連想配列

インデックス	意味	サポートする値*
crypt	通知する暗号化メソッドのリスト。優先する順にカンマ区切りにする。	*rijndael-cbc@lysator.liu.se*、 *aes256-cbc*、 *aes192-cbc*、 *aes128-cbc*、 *3des-cbc*、 *blowfish-cbc*、 *cast128-cbc*、 *arcfour* および *none***
comp	通知する圧縮メソッドのリスト。 優先する順にカンマ区切りにする。	*zlib* および *none*
mac	通知する MAC メソッドのリスト。優先する順にカンマ区切りにする。	*hmac-sha1*、 *hmac-sha1-96*、 *hmac-ripemd160*、 *hmac-ripemd160@openssh.com* および *none***

***注意*: 暗号化、MAC メソッドの "none"** *セキュリティ上の問題で、none は ビルド時に適切な ./configure オプションを使用して明示的に有効にしない限り、構成している » libssh2 によって無効にされます。 詳細は構成するライブラリのドキュメントを参照ください。*

callbacks

callbacks は以下のパラメータのいくつかあるいは全てを含む連想配列

コールバックパラメータ

インデックス	意味	プロトタイプ
ignore	**SSH2_MSG_IGNORE** パケットを受信したときにコールする関数名	void ignore_cb($message)
debug	**SSH2_MSG_DEBUG** パケットを受信したときにコールする関数名	void debug_cb($message, $language, $always_display)
macerror	パケットを受信したがメッセージ認証コードに失敗した場合にコールされる関数名。 もしコールバックが **TRUE** を返す場合、不整合は無視されます。 そうでない場合、接続は終了します。	bool macerror_cb($packet)
disconnect	**SSH2_MSG_DISCONNECT** パケットを受信したときにコールする関数名	void disconnect_cb($reason, $message, $language)

返り値

*成功した場合にリソース、エラー時に **FALSE** を返します。*

例

Example#1 ssh2_connect() の例

パケット送信時に 3des-cbc 、 パケット受信時に任意の強度の aes cipher、 両方向で無圧縮、 Group1 での鍵交換という設定で強制的に接続をオープンします。

```
<?php
/* もしサーバが接続を終了した場合、ユーザーに通知する */
function my_ssh_disconnect($reason, $message, $language) {
    printf("Server disconnected with reason code [%d] and message: %s¥n",
           $reason, $message);
}

$methods = array(
  'kex' => 'diffie-hellman-group1-sha1',
  'client_to_server' => array(
    'crypt' => '3des-cbc',
    'comp' => 'none'),
  'server_to_client' => array(
    'crypt' => 'aes256-cbc,aes192-cbc,aes128-cbc',
    'comp' => 'none'));

$callbacks = array('disconnect' => 'my_ssh_disconnect');

$connection = ssh2_connect('shell.example.com', 22, $methods, $callbacks);
if (!$connection) die('Connection failed');
?>
```

参考

- ssh2_fingerprint()
- ssh2_auth_none()
- ssh2_auth_password()
- ssh2_auth_pubkey_file()

ssh2_exec

(PECL ssh2:0.10-0.9)

ssh2_exec — リモートサーバ上でコマンドを実行する

説明

resource **ssh2_exec** (resource $session , string $command [, string $pty [, array $env [, int $width [, int $height [, int $width_height_type]]]]])

コマンドをリモートエンドで実行し、チャネルを割り当てます。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

command

pty

env

env には、 対象となる環境で設定する名前/値のペアを連想配列で渡します。

width

仮想端末の幅。

height

仮想端末の高さ。

width_height_type

width_height_type は、 **SSH2_TERM_UNIT_CHARS** あるいは **SSH2_TERM_UNIT_PIXELS** のいずれかです。

返り値

成功時にストリームを返し、失敗時に **FALSE** を返します。

例

Example#1 コマンドの実行

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stream = ssh2_exec($connection, '/usr/local/bin/php -i');
?>
```

参考

- ssh2_connect()
- ssh2_shell()
- ssh2_tunnel()

---

ssh2_fetch_stream

(PECL ssh2:0.10-0.9)

ssh2_fetch_stream — 拡張データストリームを取得する

説明

resource **ssh2_fetch_stream** (resource $channel , int $streamid)

SSH2 チャネルストリームと関連付けられたサブストリームを取得します。 SSH2 プロトコルは現在ただ 1 つのサブストリーム STDERR を定義しています。これはサブストリーム ID として **SSH2_STREAM_STDERR** (1 として定義) を持ちます。

パラメータ

channel

streamid

SSH2 チャネルストリーム。

返り値

ストリームリソースを返します。

例

Example#1 シェルをオープンし、それと関連付けられた stderr ストリームを処理する

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stdio_stream = ssh2_shell($connection);
$stderr_stream = ssh2_fetch_stream($stdio_stream, SSH2_STREAM_STDERR);
?>
```

参考

- ssh2_shell()
- ssh2_exec()
- ssh2_connect()

---

ssh2_fingerprint

(PECL ssh2:0.10-0.9)

ssh2_fingerprint — リモートサーバのフィンガープリントを処理する

説明

string **ssh2_fingerprint** (resource $session [, int $flags])

アクティブなセッションからサーバホスト鍵のハッシュを返します。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

flags

flags は **SSH2_FINGERPRINT_MD5**、 **SSH2_FINGERPRINT_HEX** で論理和された **SSH2_FINGERPRINT_SHA1**、 **SSH2_FINGERPRINT_RAW** のいずれかです。 デフォルトは **SSH2_FINGERPRINT_MD5** | **SSH2_FINGERPRINT_HEX** です。

返り値

ホストキーのハッシュを文字列で返します。

例

Example#1 既知の値に対するフィンガープリントのチェック

```
<?php
$known_host = '6F89C2F0A719B30CC38ABDF90755F2E4';

$connection = ssh2_connect('shell.example.com', 22);

$fingerprint = ssh2_fingerprint($connection,
               SSH2_FINGERPRINT_MD5 | SSH2_FINGERPRINT_HEX);

if ($fingerprint != $known_host) {
   die("HOSTKEY MISMATCH!¥n" .
       "Possible Man-In-The-Middle Attack?");
}
?>
```

---

ssh2_methods_negotiated

(PECL ssh2:0.10-0.9)

ssh2_methods_negotiated — ネゴシエートされたメソッドのリストを返す

説明

array **ssh2_methods_negotiated** (resource $session)

ネゴシエートされたメソッドのリストを返します。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

返り値

例

Example#1 どのメソッドでネゴシエートされたかの判定

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
$methods = ssh2_methods_negotiated($connection);

echo "Encryption keys were negotiated using: {$methods['kex']}¥n";
echo "Server identified using an {$methods['hostkey']} with ";
echo "fingerprint: " . ssh2_fingerprint($connection) . "¥n";

echo "Client to Server packets will use methods:¥n";
echo "¥tCrypt: {$methods['client_to_server']['crypt']}¥n";
echo "¥tComp: {$methods['client_to_server']['comp']}¥n";
echo "¥tMAC: {$methods['client_to_server']['mac']}¥n";

echo "Server to Client packets will use methods:¥n";
echo "¥tCrypt: {$methods['server_to_client']['crypt']}¥n";
echo "¥tComp: {$methods['server_to_client']['comp']}¥n";
echo "¥tMAC: {$methods['server_to_client']['mac']}¥n";

?>
```

参考

- ssh2_connect()

---

ssh2_publickey_add

(PECL ssh2:0.10)

ssh2_publickey_add — 認証済み公開鍵を追加する

説明

bool **ssh2_publickey_add** (resource $pkey , string $algoname , string $blob [, bool $overwrite [, array $attributes]])

> **注意**: この公開鍵サブシステムは、クライアントが認証 済 の公開鍵をサーバ上で管理するために使用されます。公開鍵認証によりリモートシステムで認証を行うには、かわりに ssh2_auth_pubkey_file() 関数を使用してください。

パラメータ

pkey

ssh2_publickey_init() が作成した、 公開鍵サブシステムのリソース。

algoname

公開鍵のアルゴリズム。例: ssh-dss、ssh-rsa

blob

生のバイナリデータとしての blob 形式の公開鍵。

overwrite

指定したキーがすでに存在する場合に、それを上書きしますか?

attributes

この公開鍵に代入する属性の連想配列。サポートされる属性の一覧は、 ietf-secsh-publickey-subsystem を参照ください。 必須属性を設定するには、属性名の先頭にアスタリスクをつけてください。 サーバが必須属性をサポートしていない場合、追加処理は異常終了します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ssh2_publickey_add() で公開鍵を追加する

```
<?php
$ssh2 = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($ssh2, 'jdoe', 'password');
$pkey = ssh2_publickey_init($ssh2);

$keyblob = base64_decode('
AAAAB3NzaC1yc2EAAAABIwAAAIEA5HVt6VqSGd5PTrLRdjNONxXH1tVFGn0
Bd26BF0aCP9qyJRlvdJ3j4WBeX4ZmrveGrjMgkseSYc4xZ26sDHwfL351xj
zaLpipu¥BGRrw17mWVBhuCExo476ri5tQFzbTc54VEHYckxQ16CjSTibI5X
69GmnYC9PNqEYq/1TP+HF10=');

ssh2_publickey_add($ssh2, 'ssh-rsa', $keyblob, false, array('comment'=>"John's Key"));
?>
```

参考

- ssh2_publickey_init()
- ssh2_publickey_remove()
- ssh2_publickey_list()

---

ssh2_publickey_init

(PECL ssh2:0.10)

ssh2_publickey_init — 公開鍵サブシステムを初期化する

説明

resource **ssh2_publickey_init** (resource $session)

すでに接続している SSH2 サーバから、公開鍵サブシステムを要求します。

公開鍵サブシステムを使用すると、 すでに接続・認証済みのクライアントが、 対象のサーバに保存されている認証済み公開鍵の一覧を管理できるようになります。 管理方法は、サーバの実装に依存しません。 リモートサーバが公開鍵サブシステムをサポートしていない場合、 **ssh2_publickey_init()** 関数は **FALSE** を返します。

パラメータ

session

返り値

他のすべての ssh2_publickey_*() メソッドで使用する、 SSH2 公開鍵サブシステムリソースを返します。 失敗した場合は **FALSE** を返します。

注意

> **注意**: この公開鍵サブシステムは、クライアントが認証 済 の公開鍵をサーバ上で管理するために使用されます。公開鍵認証によりリモートシステムで認証を行うには、かわりに ssh2_auth_pubkey_file() 関数を使用してください。

参考

- ssh2_publickey_add()
- ssh2_publickey_remove()
- ssh2_publickey_list()

---

ssh2_publickey_list

(PECL ssh2:0.10)

ssh2_publickey_list — 現在認証済みの公開鍵の一覧を表示する

説明

array **ssh2_publickey_list** (resource $pkey)

現在認証済みの公開鍵の一覧を表示します。

パラメータ

pkey

公開鍵サブシステムのリソース。

返り値

鍵を、数値添字の配列で返します。 個々の要素は連想配列となっており、その内容は name、blob、および attrs です。

公開鍵の要素

配列のキー	意味
name	この公開鍵で使用しているアルゴリズムの名前。例: *ssh-dss* あるいは *ssh-rsa*。
blob	生のバイナリデータとしての blob 形式の公開鍵。
attrs	この公開鍵に割り当てる属性。 もっとも一般的な属性であり、バージョン 1 の公開鍵サーバが唯一サポートしている属性は *comment* です。これは任意の書式の文字列です。

例

Example#1 認証済みの鍵の一覧を ssh2_publickey_list() で表示する

```
<?php
$ssh2 = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($ssh2, 'jdoe', 'secret');
$pkey = ssh2_publickey_init($ssh2);

$list = ssh2_publickey_list($pkey);

foreach($list as $key) {
  echo "Key:  {$key['name']}¥n";
  echo "Blob:  " . chunk_split(base64_encode($key['blob']), 40, "¥n") . "¥n";
  echo "Comment:  {$key['attrs']['comment']}¥n¥n";
```

```
}
?>
```

上の例の出力は以下となります。

```
Key:  ssh-rsa
Blob:  AAAAB3NzaC1yc2EAAAABIwAAAIEA5HVt6VqSGd5P
TrLRdjNONxXH1tVFGn0Bd26BF0aCP9qyJRlvdJ3j
4WBeX4ZmrveGrjMgkseSYc4xZ26sDHwfL351xjza
Lpipu¥BGRrw17mWVBhuCExo476ri5tQFzbTc54VE
HYckxQ16CjSTibI5X69GmnYC9PNqEYq/1TP+HF10
Comment:  John's  Key

Key:  ssh-rsa
Blob:  AAAAB3NzaHVt6VqSGd5C1yc2EAAAABIwA232dnJA
AIEA5HVt6VqSGd5PTrLRdjNONxX/1TP+HF1HVt6V
qSGd50H1tVFGn0BB3NzaC1yc2EAd26BF0aCP9qyJ
RlvdJ3j4WBeX4ZmrveGrjMgkseSYc4xZ26HVt6Vq
SGd5sDHwfL351xjzaLpipu¥BGB3NzaC1yc2EA/1T
Comment:  Alice's  Key
```

注意

注意: この公開鍵サブシステムは、クライアントが認証 済 の公開鍵をサーバ上で管理するために使用されます。公開鍵認証によりリモートシステムで認証を行うには、かわりに ssh2_auth_pubkey_file() 関数を使用してください。

参考

- ssh2_publickey_init()
- ssh2_publickey_add()
- ssh2_publickey_remove()

---

ssh2_publickey_remove

(PECL ssh2:0.10)

ssh2_publickey_remove — 認証済み公開鍵を取り除く

説明

bool **ssh2_publickey_remove** (resource $pkey , string $algoname , string $blob)

認証済み公開鍵を取り除きます。

パラメータ

pkey

公開鍵サブシステムのリソース。

algoname

公開鍵のアルゴリズム。例: ssh-dss、ssh-rsa

blob

生のバイナリデータとしての blob 形式の公開鍵。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

注意: この公開鍵サブシステムは、クライアントが認証 済 の公開鍵をサーバ上で管理するために使用されます。公開鍵認証によりリモートシステムで認証を行うには、かわりに ssh2_auth_pubkey_file() 関数を使用してください。

参考

- ssh2_publickey_init()
- ssh2_publickey_add()
- ssh2_publickey_list()

---

ssh2_scp_recv

(PECL ssh2:0.10-0.9)

ssh2_scp_recv — SCP 経由でファイルを要求する

説明

bool **ssh2_scp_recv** (resource $session , string $remote_file , string $local_file)

リモートサーバからローカルファイルシステムに SCP プロトコルを使用してコピーします。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

remote_file

リモートファイルへのパス。

local_file

ローカルファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 SCP 経由でのファイルのダウンロード

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

ssh2_scp_recv($connection, '/remote/filename', '/local/filename');
?>
```

参考

- ssh2_scp_send()
- copy()

---

ssh2_scp_send

(PECL ssh2:0.10-0.9)

ssh2_scp_send — SCP 経由でファイルを送信する

説明

bool **ssh2_scp_send** (resource $session , string $local_file , string $remote_file [, int $create_mode])

ローカルファイルシステムからリモートサーバに SCP プロトコルを使用してコピーします。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

local_file

ローカルファイルへのパス。

remote_file

リモートファイルへのパス。

create_mode

ファイルは create_mode で指定されたモードで作成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 SCP 経由でのファイルのアップロード

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

ssh2_scp_send($connection, '/local/filename', '/remote/filename', 0644);
?>
```

参考

- ssh2_scp_recv()
- copy()

ssh2_sftp_lstat

(PECL ssh2:0.10-0.9)

ssh2_sftp_lstat — シンボリックリンクの情報を取得する

説明

array **ssh2_sftp_lstat** (resource $sftp , string $path)

リンクを辿らないで リモートファイルシステムのシンボリックリンクの情報を取得します。

PHP 5 では、この関数は ssh2.sftp:// ラッパーを使用した場合の lstat() 関数と同様です。 返り値も同じです。

パラメータ

sftp

path

リモートのシンボリックリンクへのパス。

返り値

返り値の詳細については stat() のドキュメントを参照ください。

例

Example#1 SFTP 経由でのシンボリックリンクの情報の取得

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);
$statinfo = ssh2_sftp_lstat($sftp, '/path/to/symlink');

$filesize = $statinfo['size'];
$group = $statinfo['gid'];
$owner = $statinfo['uid'];
$atime = $statinfo['atime'];
$mtime = $statinfo['mtime'];
$mode = $statinfo['mode'];
?>
```

参考

- ssh2_sftp_stat()
- lstat()
- stat()

---

ssh2_sftp_mkdir

(PECL ssh2:0.10-0.9)

ssh2_sftp_mkdir — ディレクトリを作成する

説明

bool **ssh2_sftp_mkdir** (resource $sftp , string $dirname [, int $mode [, bool $recursive]])

リモートファイルサーバに mode で指定された権限でディレクトリを作成します。

この関数は ssh2.sftp:// ラッパーを使用した場合の mkdir() と同様です。

パラメータ

sftp

ssh2_sftp() でオープンした SSH2 SFTP リソース。

dirname

新しいディレクトリへのパス。

mode

新しいディレクトリのパーミッション。

recursive

もし recursive が **TRUE** の場合、 dirname に必要とされる全ての親ディレクトリは自動的に作成されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 リモートサーバ上へのディレクトリの作成

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_mkdir($sftp, '/home/username/newdir');
/* もしくは次の通り:   mkdir("ssh2.sftp://$sftp/home/username/newdir");  */
?>
```

参考

- mkdir()
- ssh2_sftp_rmdir()

---

ssh2_sftp_readlink

(PECL ssh2:0.10-0.9)

ssh2_sftp_readlink — シンボリックリンクのターゲットを返す

説明

string **ssh2_sftp_readlink** (resource $sftp , string $link)

シンボリックリンクのターゲットを返します。

パラメータ

sftp

ssh2_sftp() でオープンした SSH2 SFTP リソース。

link

シンボリックリンクへのパス。

返り値

シンボリックリンク link のターゲットを返します。

例

Example#1 シンボリックリンクの読み込み

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

$target = ssh2_sftp_readlink($sftp, '/tmp/mysql.sock');
/* $target は例えば次のような感じ: '/var/run/mysql.sock' */
?>
```

参考

- readlink()
- ssh2_sftp_symlink()

---

ssh2_sftp_realpath

(PECL ssh2:0.10-0.9)

ssh2_sftp_realpath — 指定されたパス文字列の実パスを解決する

説明

string **ssh2_sftp_realpath** (resource $sftp , string $filename)

filename をリモートファイルシステム上の有効な実パスに変換します。

パラメータ

sftp

ssh2_sftp() でオープンした SSH2 SFTP リソース。

filename

返り値

実際のパスを表す文字列を返します。

例

Example#1 パス名の解決

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

$realpath = ssh2_sftp_realpath($sftp, '/home/username/../../../.././/./usr/../etc/passwd');
/* $realpath は次の通り: '/etc/passwd' */
?>
```

参考

- realpath()
- ssh2_sftp_symlink()
- ssh2_sftp_readlink()

---

ssh2_sftp_rename

(PECL ssh2:0.10-0.9)

ssh2_sftp_rename — リモートファイルを改名する

説明

bool **ssh2_sftp_rename** (resource $sftp , string $from , string $to)

リモートファイルシステム上のファイルを改名します。

パラメータ

sftp

ssh2_sftp() でオープンした SSH2 SFTP リソース。

from

to

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 sftp 経由でのファイルの改名

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_rename($sftp, '/home/username/oldname', '/home/username/newname');
?>
```

参考

- rename()

---

ssh2_sftp_rmdir

(PECL ssh2:0.10-0.9)

ssh2_sftp_rmdir — ディレクトリを削除する

説明

bool **ssh2_sftp_rmdir** (resource $sftp , string $dirname)

リモートのファイルサーバからディレクトリを削除します。

この関数は ssh2.sftp:// ラッパーを使用した場合の rmdir() と同様です。

パラメータ

sftp

ssh2_sftp() でオープンした SSH2 SFTP リソース。

dirname

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 リモートサーバ上のディレクトリの削除

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_rmdir($sftp, '/home/username/deltodel');
/* もしくは次の通り:   rmdir("ssh2.sftp://$sftp/home/username/dirtodel");  */
?>
```

参考

- rmdir()
- ssh2_sftp_mkdir()

---

ssh2_sftp_stat

(PECL ssh2:0.10-0.9)

ssh2_sftp_stat — リモートファイルシステム上のファイルの情報を取得する

説明

array **ssh2_sftp_stat** (resource $sftp , string $path)

シンボリックリンクを辿って リモートファイルシステム上のファイルの情報を取得します。

PHP 5 では、この関数は ssh2.sftp:// ラッパーを使用した場合の stat() 関数と同様です。 返される値も同じです。

パラメータ

sftp

ssh2_sftp() でオープンした SSH2 SFTP リソース。

path

返り値

どのような値が返されるのかについての詳細は、 stat() を参照ください。

例

Example#1 SFTP 経由でのファイルの情報の取得

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);
$statinfo = ssh2_sftp_stat($sftp, '/path/to/file');

$filesize = $statinfo['size'];
$group = $statinfo['gid'];
$owner = $statinfo['uid'];
$atime = $statinfo['atime'];
$mtime = $statinfo['mtime'];
$mode = $statinfo['mode'];
?>
```

参考

- ssh2_sftp_lstat()
- lstat()
- stat()

---

ssh2_sftp_symlink

(PECL ssh2:0.10-0.9)

ssh2_sftp_symlink — シンボリックリンクを作成する

説明

bool ***ssh2_sftp_symlink*** *(resource $sftp , string $target , string $link)*

リモートファイルシステム上に *target* を指す *link* という名称のシンボリックリンクを作成します。

パラメータ

sftp

ssh2_sftp() でオープンした *SSH2 SFTP* リソース。

target

シンボリックリンクのリンク対象。

link

返り値

成功した場合に ***TRUE*** を、失敗した場合に ***FALSE*** を返します。

例

Example#1 シンボリックリンクの作成

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_symlink($sftp, '/var/run/mysql.sock', '/tmp/mysql.sock');
?>
```

参考

- *ssh2_sftp_readlink()*
- *symlink()*

---

ssh2_sftp_unlink

(PECL ssh2:0.10-0.9)

ssh2_sftp_unlink — ファイルを削除する

説明

bool ***ssh2_sftp_unlink*** *(resource $sftp , string $filename)*

リモートファイルシステム上のファイルを削除します。

パラメータ

sftp

ssh2_sftp() でオープンした *SSH2 SFTP* リソース。

filename

返り値

成功した場合に ***TRUE*** を、失敗した場合に ***FALSE*** を返します。

例

Example#1 ファイルの削除

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_unlink($sftp, '/home/username/stale_file');
?>
```

参考

- *unlink()*

---

ssh2_sftp

(PECL ssh2:0.10-0.9)

ssh2_sftp — *SFTP* サブシステムを初期化する

説明

resource **ssh2_sftp** (resource $session)

すでに接続された SSH2 サーバから SFTP サブシステムを要求します。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

返り値

このメソッドは全ての他の ssh2_sftp_*() や ssh2.sftp:// fopen ラッパーで使用する SSH2 SFTP リソースを返します。

例

Example#1 SFTP 経由でのファイルのオープン

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);

$stream = fopen("ssh2.sftp://$sftp/path/to/file", 'r');
?>
```

参考

- ssh2_scp_recv()
- ssh2_scp_send()

---

ssh2_shell

(PECL ssh2:0.10-0.9)

ssh2_shell — 対話式のシェルを要求する

説明

resource **ssh2_shell** (resource $session [, string $term_type [, array $env [, int $width [, int $height [, int $width_height_type]]]]])

リモートエンド上のシェルをオープンし、そのためのストリームを割り当てます。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

term_type

term_type は対象システムの /etc/termcap に記載されたエントリの一つに相当し、 デフォルトは vanilla です。

env

env には、 対象の環境に設定する名前/値の組の連想配列を渡すことができます。

width

仮想端末の幅。

height

仮想端末の高さ。

width_height_type

width_height_type は **SSH2_TERM_UNIT_CHARS** あるいは **SSH2_TERM_UNIT_PIXELS** のいずれかです。

返り値

例

Example#1 コマンドの実行

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stream = ssh2_shell($connection, 'vt102', null, 80, 24, SSH2_TERM_UNIT_CHARS);
?>
```

参考

- ssh2_exec()
- ssh2_tunnel()

- ssh2_fetch_stream()

---

ssh2_tunnel

(PECL ssh2:0.10-0.9)

ssh2_tunnel — リモートサーバを経由するトンネルをオープンする

説明

resource **ssh2_tunnel** (resource $session , string $host , int $port)

現在接続している SSH サーバを経由して、 任意のホスト/ポートへのソケットストリームをオープンします。

パラメータ

session

ssh2_connect() のコールによって取得した SSH 接続リンク ID。

host

port

返り値

例

Example#1 任意のホストへのトンネルのオープン

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_pubkey_file($connection, 'username', 'id_dsa.pub', 'id_dsa');

$tunnel = ssh2_tunnel($connection, '10.0.0.101', 12345);
?>
```

参考

- ssh2_connect()
- fsockopen()

目次

統計関数

導入

統計用の拡張モジュールです。統計計算のために有用な関数を、 何十個も含んでいます。この拡張モジュールは、B. Brown & J. Lavato の DCDFLIB (Library of C routines for Cumulative Distributions Functions, Inverses, and Other parameters) および Barry Brown, James Lavato & Kathy Russell の RANDLIB という 2 つの科学ライブラリのラッパーです。 CD 関数および PD 関数を含みます。

インストール手順

この » PECL 拡張モジュールは PHP にバンドルされていません。

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/stats.

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

要件

外部ライブラリは不要です。使用するライブラリは、 この拡張モジュールにバンドルされています。

---

stats_absolute_deviation

(PECL stats:1.0.0-1.0.2)

stats_absolute_deviation — 値の配列の絶対偏差を返す

説明

float **stats_absolute_deviation** (array $a)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

a

返り値

---

stats_cdf_beta

(PECL stats:1.0.0-1.0.2)

stats_cdf_beta — ベータ分布用の CDF 関数。ベータ分布のパラメータのいずれかを、 その他のパラメータの値から計算する

説明

float **stats_cdf_beta** (float $par1 , float $par2 , float $par3 , int $which)

Method Cumulative distribution function (P) is calculated directly by code associated with the following reference. DiDinato, A. R. and Morris, A. H. Algorithm 708: Significant Digit Computation of the Incomplete Beta Function Ratios. ACM Trans. Math. Softw. 18 (1993), 360-373. Computation of other parameters involve a search for a value that produces the desired value of P. The search relies on the monotinicity of P with the other parameter. Note The beta density is proportional to t^(A-1) * (1-t)^(B-1) Arguments P -- The integral from 0 to X of the chi-square distribution. Input range: [0, 1]. Q -- 1-P. Input range: [0, 1]. P + Q = 1.0. X -- Upper limit of integration of beta density. Input range: [0,1]. Search range: [0,1] Y -- 1-X. Input range: [0,1]. Search range: [0,1] X + Y = 1.0. A -- The first parameter of the beta density. Input range: (0, +infinity). Search range: [1D-100,1D100] B -- The second parameter of the beta density. Input range: (0, +infinity). Search range: [1D-100,1D100] STATUS -- 0 if calculation completed correctly -I if input parameter number I is out of range 1 if answer appears to be lower than lowest search bound 2 if answer appears to be higher than greatest search bound 3 if P + Q .ne. 1 4 if X + Y .ne. 1 BOUND -- Undefined if STATUS is 0 Bound exceeded by parameter number I if STATUS is negative. Lower search bound if STATUS is 1. Upper search bound if STATUS is 2.

パラメータ

par1

par2

par3

which

Integer indicating which of the next four argument values is to be calculated from the others. Legal range: 1..4 iwhich = 1 : Calculate P and Q from X,Y,A and B iwhich = 2 : Calculate X and Y from P,Q,A and B iwhich = 3 : Calculate A from P,Q,X,Y and B iwhich = 4 : Calculate B from P,Q,X,Y and A

返り値

STATUS -- 0 if calculation completed correctly -I if input parameter number I is out of range 1 if answer appears to be lower than lowest search bound 2 if answer appears to be higher than greatest search bound 3 if P + Q .ne. 1 4 if X + Y .ne. 1

stats_cdf_binomial

(PECL stats:1.0.0-1.0.2)

stats_cdf_binomial — 二項分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float **stats_cdf_binomial** (float $par1 , float $par2 , float $par3 , int $which)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

stats_cdf_cauchy

(PECL stats:1.0.0-1.0.2)

stats_cdf_cauchy — 未ドキュメント化

説明

float **stats_cdf_cauchy** (float $par1 , float $par2 , float $par3 , int $which)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

stats_cdf_chisquare

(PECL stats:1.0.0-1.0.2)

stats_cdf_chisquare — カイ二乗分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float **stats_cdf_chisquare** (float $par1 , float $par2 , int $which)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

which

返り値

stats_cdf_exponential

(PECL stats:1.0.0-1.0.2)

stats_cdf_exponential — 未ドキュメント化

説明

float ***stats_cdf_exponential*** *(float $par1 , float $par2 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

which

返り値

---

stats_cdf_f

(PECL stats:1.0.0-1.0.2)

stats_cdf_f — F 分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float ***stats_cdf_f*** *(float $par1 , float $par2 , float $par3 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_cdf_gamma

(PECL stats:1.0.0-1.0.2)

stats_cdf_gamma — ガンマ分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float ***stats_cdf_gamma*** *(float $par1 , float $par2 , float $par3 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_cdf_laplace

(PECL stats:1.0.0-1.0.2)

stats_cdf_laplace — 未ドキュメント化

説明

float ***stats_cdf_laplace*** *(float $par1 , float $par2 , float $par3 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_cdf_logistic

(PECL stats:1.0.0-1.0.2)

stats_cdf_logistic — *未ドキュメント化*

説明

float ***stats_cdf_logistic*** *(float $par1 , float $par2 , float $par3 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_cdf_negative_binomial

(PECL stats:1.0.0-1.0.2)

stats_cdf_negative_binomial — *負の二項分布のパラメータのいずれかを、その他のパラメータの値から計算する*

説明

float ***stats_cdf_negative_binomial*** *(float $par1 , float $par2 , float $par3 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_cdf_noncentral_chisquare

(PECL stats:1.0.0-1.0.2)

stats_cdf_noncentral_chisquare — *非心カイ二乗分布のパラメータのいずれかを、その他のパラメータの値から計算する*

説明

float ***stats_cdf_noncentral_chisquare*** *(float $par1 , float $par2 , float $par3 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_cdf_noncentral_f

(PECL stats:1.0.0-1.0.2)

stats_cdf_noncentral_f — 非心 F 分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float **stats_cdf_noncentral_f** (float $par1 , float $par2 , float $par3 , float $par4 , int $which)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

par4

which

返り値

---

stats_cdf_poisson

(PECL stats:1.0.0-1.0.2)

stats_cdf_poisson — ポアソン分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float **stats_cdf_poisson** (float $par1 , float $par2 , int $which)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

which

返り値

---

stats_cdf_t

(PECL stats:1.0.0-1.0.2)

stats_cdf_t — T 分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float **stats_cdf_t** (float $par1 , float $par2 , int $which)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

which

返り値

---

stats_cdf_uniform

(PECL stats:1.0.0-1.0.2)

stats_cdf_uniform — 未ドキュメント化

説明

float ***stats_cdf_uniform*** *(float $par1 , float $par2 , float $par3 , int $which)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_cdf_weibull

(PECL stats:1.0.0-1.0.2)

stats_cdf_weibull — 未ドキュメント化

説明

float ***stats_cdf_weibull*** *(float $par1 , float $par2 , float $par3 , int $which)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_covariance

(PECL stats:1.0.0-1.0.2)

stats_covariance — ふたつのデータセットの共分散を計算する

説明

float ***stats_covariance*** *(array $a , array $b)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

a

b

返り値

---

stats_den_uniform

(PECL stats:1.0.0-1.0.2)

stats_den_uniform — 未ドキュメント化

説明

float ***stats_den_uniform*** *(float $x , float $a , float $b)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

a

b

返り値

---

stats_dens_beta

(PECL stats:1.0.0-1.0.2)

stats_dens_beta — *未ドキュメント化*

説明

float ***stats_dens_beta*** *(float $x , float $a , float $b)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

a

b

返り値

---

stats_dens_cauchy

(PECL stats:1.0.0-1.0.2)

stats_dens_cauchy — *未ドキュメント化*

説明

float ***stats_dens_cauchy*** *(float $x , float $ave , float $stdev)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

ave

stdev

返り値

---

stats_dens_chisquare

(PECL stats:1.0.0-1.0.2)

stats_dens_chisquare — *未ドキュメント化*

説明

float ***stats_dens_chisquare*** *(float $x , float $dfr)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

dfr

返り値

---

stats_dens_exponential

(PECL stats:1.0.0-1.0.2)

stats_dens_exponential — 未ドキュメント化

説明

float **stats_dens_exponential** (float $x , float $scale)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

scale

返り値

---

stats_dens_f

(PECL stats:1.0.0-1.0.2)

stats_dens_f —

説明

float **stats_dens_f** (float $x , float $dfr1 , float $dfr2)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

dfr1

dfr2

返り値

---

stats_dens_gamma

(PECL stats:1.0.0-1.0.2)

stats_dens_gamma — 未ドキュメント化

説明

float **stats_dens_gamma** (float $x , float $shape , float $scale)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

shape

scale

返り値

---

stats_dens_laplace

(PECL stats:1.0.0-1.0.2)

stats_dens_laplace — 未ドキュメント化

説明

float **stats_dens_laplace** (float $x , float $ave , float $stdev)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

ave

stdev

返り値

---

stats_dens_logistic

(PECL stats:1.0.0-1.0.2)

stats_dens_logistic — *未ドキュメント化*

説明

float ***stats_dens_logistic*** *(float $x , float $ave , float $stdev)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

ave

stdev

返り値

---

stats_dens_negative_binomial

(PECL stats:1.0.0-1.0.2)

stats_dens_negative_binomial — *未ドキュメント化*

説明

float ***stats_dens_negative_binomial*** *(float $x , float $n , float $pi)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

n

pi

返り値

---

stats_dens_normal

(PECL stats:1.0.0-1.0.2)

stats_dens_normal — *未ドキュメント化*

説明

float ***stats_dens_normal*** *(float $x , float $ave , float $stdev)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

ave

stdev

返り値

---

stats_dens_pmf_binomial

(PECL stats:1.0.0-1.0.2)

stats_dens_pmf_binomial — *未ドキュメント化*

説明

float ***stats_dens_pmf_binomial*** *(float $x , float $n , float $pi)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

n

pi

返り値

---

stats_dens_pmf_hypergeometric

(PECL stats:1.0.0-1.0.2)

stats_dens_pmf_hypergeometric —

説明

float ***stats_dens_pmf_hypergeometric*** *(float $n1 , float $n2 , float $N1 , float $N2)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

n1

n2

N1

N2

返り値

---

stats_dens_pmf_poisson

(PECL stats:1.0.0-1.0.2)

stats_dens_pmf_poisson — 未ドキュメント化

説明

float ***stats_dens_pmf_poisson*** *(float $x , float $lb)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

lb

返り値

---

stats_dens_t

(PECL stats:1.0.0-1.0.2)

stats_dens_t — 未ドキュメント化

説明

float ***stats_dens_t*** *(float $x , float $dfr)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

dfr

返り値

---

stats_dens_weibull

(PECL stats:1.0.0-1.0.2)

stats_dens_weibull — 未ドキュメント化

説明

float **stats_dens_weibull** (float $x , float $a , float $b)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

a

b

返り値

---

stats_harmonic_mean

(PECL stats:1.0.0-1.0.2)

stats_harmonic_mean — 値の配列の調和平均を返す

説明

number **stats_harmonic_mean** (array $a)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

a

返り値

---

stats_kurtosis

(PECL stats:1.0.0-1.0.2)

stats_kurtosis — 配列内のデータの尖度を計算する

説明

float **stats_kurtosis** (array $a)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

a

返り値

---

stats_rand_gen_beta

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_beta — 無作為な値を生成する

説明

float **stats_rand_gen_beta** (float $a , float $b)

パラメータ A および B で表されるベータ分布から、無作為な値を返します。 ベータ密度は、< x < に対して x^(a-1) * (1-x)^(b-1) / B(a,b) となります。これは R. C. H. Cheng の手法です。

パラメータ

a

b

返り値

---

stats_rand_gen_chisquare

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_chisquare — 自由度 "df" の乱数で表されるカイ二乗分布から、無作為な値を返す

説明

float ***stats_rand_gen_chisquare*** *(float $df)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

df

返り値

---

stats_rand_gen_exponential

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_exponential — 平均値 "av" の指数分布から、無作為な値を返す

説明

float ***stats_rand_gen_exponential*** *(float $av)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

av

返り値

---

stats_rand_gen_f

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_f — 無作為な値を返す

説明

float ***stats_rand_gen_f*** *(float $dfn , float $dfd)*

分子の自由度が "dfn"、分母の自由度が "dfd" の F（分散比）分布から、無作為な値を返します。カイ二乗の変量の比を直接生成します。

パラメータ

dfn

dfd

返り値

---

stats_rand_gen_funiform

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_funiform — low（それ自身は含まない）と high（それ自身は含まない）の間の一様な浮動小数点数値を生成する

説明

float ***stats_rand_gen_funiform*** *(float $low , float $high)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

low

high

返り値

---

stats_rand_gen_gamma

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_gamma — ガンマ分布から無作為な値を生成する

説明

float **stats_rand_gen_gamma** (float $a , float $r)

密度 (A**R)/Gamma(R) * X**(R-1) * Exp(-A*X) のガンマ分布から、 無作為な値を生成します。

パラメータ

a

ガンマ分布の位置母数 (a > 0)。

r

ガンマ分布の形状母数 (r > 0)。

返り値

---

stats_rand_gen_ibinomial_negative

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_ibinomial_negative — 負の二項分布から無作為な値を生成する。引数: n - 無作為な値を生成するために行う負の二項分布の試行回数 (n > 0)、p - 事象の発生する確率 (0 < p < 1))

説明

int **stats_rand_gen_ibinomial_negative** (int $n , float $p)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

n

p

返り値

---

stats_rand_gen_ibinomial

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_ibinomial — 二項分布から無作為な値を生成する。二項分布の試行回数を "n" (n >= 0)、各試行で事象の発生する確率を "pp" ([0;1]) とし、BTPE アルゴリズムを使用する

説明

int **stats_rand_gen_ibinomial** (int $n , float $pp)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

n

pp

返り値

---

stats_rand_gen_int

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_int — 1 から 2147483562 までの間の無作為な整数値を生成する

説明

int **stats_rand_gen_int** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

返り値

---

stats_rand_gen_ipoisson

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_ipoisson — 平均 "mu" (mu >= 0.0) のポアソン分布から無作為な値を生成する

説明

int **stats_rand_gen_ipoisson** (float $mu)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

mu

返り値

---

stats_rand_gen_iuniform

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_iuniform — LOW（それ自身を含む）と HIGH（それ自身を含む）の間の一様分布から整数値を生成する

説明

int **stats_rand_gen_iuniform** (int $low , int $high)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

low

high

返り値

---

stats_rand_gen_noncenral_chisquare

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_noncenral_chisquare — 自由度 "df"、非心母数 "xnonc" の非心カイ二乗分布から無作為な値を生成する。 d は >= 1.0、xnonc は >= 0.0 でなければならない

説明

float **stats_rand_gen_noncenral_chisquare** (float $df , float $xnonc)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

df

xnonc

返り値

---

stats_rand_gen_noncentral_f

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_noncentral_f — 分子の自由度が "dfn"、分母の自由度が "dfd"、非心母数が "xnonc" の非心 F (分散比) 分布から、無作為な値を返す。 非心カイ二乗変量の分子とカイ二乗変量の分母の比を直接生成する

説明

float **stats_rand_gen_noncentral_f** (float $dfn , float $dfd , float $xnonc)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

dfn

dfd

xnonc

返り値

---

stats_rand_gen_noncentral_t

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_noncentral_t — 非心 T 分布から無作為な値を生成する

説明

float **stats_rand_gen_noncentral_t** (float $df , float $xnonc)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

df

xnonc

返り値

---

stats_rand_gen_normal

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_normal — mean、av および標準偏差 sd (sd >= 0) によって表される正規分布から無作為な値を生成する。 Method : Renames SNORM from TOMS as slightly modified by BWB to use RANF instead of SUNIF.

説明

float **stats_rand_gen_normal** (float $av , float $sd)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

av

sd

返り値

---

stats_rand_gen_t

(PECL stats:1.0.0-1.0.2)

stats_rand_gen_t — T 分布から無作為な値を生成する

説明

float **stats_rand_gen_t** (float $df)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

df

返り値

---

stats_rand_get_seeds

(PECL stats:1.0.0-1.0.2)

stats_rand_get_seeds — 未ドキュメント化

説明

array ***stats_rand_get_seeds*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

返り値

---

stats_rand_phrase_to_seeds

(PECL stats:1.0.0-1.0.2)

stats_rand_phrase_to_seeds — 乱数ジェネレータ用のふたつのシードを生成する

説明

array ***stats_rand_phrase_to_seeds*** *(string $phrase)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

phrase

返り値

---

stats_rand_ranf

(PECL stats:1.0.0-1.0.2)

stats_rand_ranf — 0 から 1 (区間の両端は含まない) までの一様分布から、 現在のジェネレータを使用して無作為な浮動小数点数値を返す

説明

float ***stats_rand_ranf*** *(void)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

返り値

---

stats_rand_setall

(PECL stats:1.0.0-1.0.2)

stats_rand_setall — 未ドキュメント化

説明

void ***stats_rand_setall*** *(int $iseed1 , int $iseed2)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

iseed1

iseed2

返り値

---

stats_skew

(PECL stats:1.0.0-1.0.2)

stats_skew — 配列内のデータの歪度を計算する

説明

float **stats_skew** (array $a)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

a

返り値

---

stats_standard_deviation

(PECL stats:1.0.0-1.0.2)

stats_standard_deviation — 標準偏差を返す

説明

float **stats_standard_deviation** (array $a [, bool $sample])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

a

sample

返り値

---

stats_stat_binomial_coef

(PECL stats:1.0.0-1.0.2)

stats_stat_binomial_coef — 未ドキュメント化

説明

float **stats_stat_binomial_coef** (int $x , int $n)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

x

n

返り値

---

stats_stat_correlation

(PECL stats:1.0.0-1.0.2)

stats_stat_correlation — 未ドキュメント化

説明

float **stats_stat_correlation** (array $arr1 , array $arr2)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

arr1

arr2

返り値

---

stats_stat_gennch

(PECL stats:1.0.0-1.0.2)

stats_stat_gennch — 未ドキュメント化

説明

float ***stats_stat_gennch*** *(int $n)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

n

返り値

---

stats_stat_independent_t

(PECL stats:1.0.0-1.0.2)

stats_stat_independent_t — 未ドキュメント化

説明

float ***stats_stat_independent_t*** *(array $arr1 , array $arr2)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

arr1

arr2

返り値

---

stats_stat_innerproduct

(PECL stats:1.0.0-1.0.2)

stats_stat_innerproduct —

説明

float ***stats_stat_innerproduct*** *(array $arr1 , array $arr2)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

arr1

arr2

返り値

---

stats_stat_noncentral_t

(PECL stats:1.0.0-1.0.2)

stats_stat_noncentral_t — 非心 t 分布のパラメータのいずれかを、その他のパラメータの値から計算する

説明

float ***stats_stat_noncentral_t*** *(float $par1 , float $par2 , float $par3 , int $which)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

par1

par2

par3

which

返り値

---

stats_stat_paired_t

(PECL stats:1.0.0-1.0.2)

stats_stat_paired_t — 未ドキュメント化

説明

float ***stats_stat_paired_t*** *(array $arr1 , array $arr2)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

arr1

arr2

返り値

---

stats_stat_percentile

(PECL stats:1.0.0-1.0.2)

stats_stat_percentile — 未ドキュメント化

説明

float ***stats_stat_percentile*** *(float $df , float $xnonc)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

df

xnonc

返り値

---

stats_stat_powersum

(PECL stats:1.0.0-1.0.2)

stats_stat_powersum — 未ドキュメント化

説明

float ***stats_stat_powersum*** *(array $arr , float $power)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

arr

power

返り値

---

stats_variance

(PECL stats:1.0.0-1.0.2)

stats_variance — 母分散を返す

説明

float **stats_variance** (array $a [, bool $sample])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

a

sample

返り値

目次

- stats_rand_gen_iuniform — LOW（それ自身を含む）と HIGH（それ自身を含む）の間の一様分布から整数値を生成する
- stats_rand_gen_noncenral_chisquare — 自由度 "df"、非心母数 "xnonc" の非心カイ二乗分布から無作為な値を生成する。 d は >= 1.0、xnonc は >= 0.0 でなければならない
- stats_rand_gen_noncentral_f — 分子の自由度が "dfn"、分母の自由度が "dfd"、非心母数が "xnonc" の非心 F（分散比）分布から、無作為な値を返す。 非心カイ二乗変量の分子とカイ二乗変量の分母の比を直接生成する
- stats_rand_gen_noncentral_t — 非心 T 分布から無作為な値を生成する
- stats_rand_gen_normal — mean、av および標準偏差 sd（sd >= 0）によって表される正規分布から無作為な値を生成する。 Method : Renames SNORM from TOMS as slightly modified by BWB to use RANF instead of SUNIF.
- stats_rand_gen_t — T 分布から無作為な値を生成する
- stats_rand_get_seeds — 未ドキュメント化
- stats_rand_phrase_to_seeds — 乱数ジェネレータ用のふたつのシードを生成する
- stats_rand_ranf — 0 から 1（区間の両端は含まない）までの一様分布から、 現在のジェネレータを使用して無作為な浮動小数点数値を返す
- stats_rand_setall — 未ドキュメント化
- stats_skew — 配列内のデータの歪度を計算する
- stats_standard_deviation — 標準偏差を返す
- stats_stat_binomial_coef — 未ドキュメント化
- stats_stat_correlation — 未ドキュメント化
- stats_stat_gennch — 未ドキュメント化
- stats_stat_independent_t — 未ドキュメント化
- stats_stat_innerproduct — 説明
- stats_stat_noncentral_t — 非心 t 分布のパラメータのいずれかを、その他のパラメータの値から計算する
- stats_stat_paired_t — 未ドキュメント化
- stats_stat_percentile — 未ドキュメント化
- stats_stat_powersum — 未ドキュメント化
- stats_variance — 母分散を返す

ストリーム関数

導入

ストリームは、PHP 4.3.0 に、 ファイル、ネットワーク、データ圧縮などに関する、 共通した一連の関数群と利用法を持つ操作の一般化の手法として 導入されました。もっとも単純な定義では、ストリーム というのは、ストリーミング可能な動作を体現する resource オブジェクトといえます。つまり、ストリームには線的に読み出したり、 あるいは書き込んだりすることが可能で、かつ、 ストリーム上の任意の場所に fseek() できる場合もあります。

ラッパー というのは、ストリームにおいてどのように特定の プロトコル/エンコーディングを扱うかを扱うかを指示する付加的なコード です。たとえば、http ラッパーは、どのようにして URL を、リモートサーバのファイルに対する HTTP/1.0 リクエストに転換するかを知っています。PHP には、デフォルトで組み込まれているラッパーが多数存在しますが（サポートされるプロトコル/ラッパー を参照ください）、 それに加え、カスタムラッパーをstream_wrapper_register() を利用して PHP スクリプトの内部から、あるいはストリーム API を用いて、 直接拡張モジュールの内部から追加できます。（ストリーム API については ストリームの作成 を参照ください）あらゆる種類のラッパーが PHP に追加できるので、 特にラッパーでできることの限界はありません。 現在登録されているラッパーの種類を知るには、 stream_get_wrappers() を使います。

リソースは次のような形でストリームとして参照されます: scheme ://target

- scheme (string) - 使用されるラッパの名称です。例として、 file, http, https, ftp, ftps, compress.zlib, compress.bz2, php などが含まれます。 PHP 組み込みのラッパについては、サポートされるプロトコル/ラッパー を参照ください。もしラッパが指定されていない場合は、使用している関数の デフォルトが利用されます(通常は file://)。
- target - 使用するラッパによって解釈が異なります。 ファイルシステムに関連したストリームの場合、 一般的にこのパラメータは対象となるファイルの ファイル名とパスを表します。 ネットワークに関連したストリームの場合、一般的に このパラメータはホスト名と、(多くの場合付加されるはずの）パス名です。 PHP 組み込みのラッパに、どのようにターゲットが解釈されるかは、 サポートされるプロトコル/ラッパー を参照ください。

ストリームフィルタ

フィルタ は、ストリームから読み込まれたり、 あるいは、ストリームから書き込まれたするデータに対して何らかの 操作を行う、最終段階にあるコードです。 PHP スクリプトの内部で、 stream_filter_register() を使って、 あるいは拡張モジュールの内部で ストリームの作成 に示された 関連する API を使って、カスタムフィルタを作成することができます。 現在登録されているフィルタの一覧を取得するには、stream_get_filters() をお使いください。

ストリームコンテキスト

コンテキスト は、ストリームの挙動を変えたり、 拡張したりすることのできる パラメータ と ラッパ固有の オプション の集合です。 コンテキスト は、 stream_context_create() を使って生成しますが、 これは、ほとんどのファイルシステム関連のストリーム生成関数に 渡すことができます。(例えば fopen(), file(), file_get_contents() など)

オプション は、 stream_context_create() の呼び出し時に指定しますが、 後で stream_context_set_option() を使って 指定することもできます。 ラッパ固有の オプション については、 組み込みラッパの一覧に説明があります。（サポートされるプロトコル/ラッパー を参照ください）

さらに、stream_context_set_params() を使うことで、 パラメータ を、コンテキスト に対し指定することができます。なお、現在のところ、PHP にサポートされているコンテキストパラメータ は、 notification のみです。このパラメータの値は、 ストリームに発生した何らかのイベントを通知するために呼び出される関数の 名前でなくてはなりません。 通知関数は次のような 6 個のパラメータを取ります:

```
void my_notifier ( int $notification_code , int $severity , string $message , int $message_code , int
$bytes_transferred , int $bytes_max )
```

*notification_code と severity は、下記の **STREAM_NOTIFY_*** 定数に該当する数値です。 もし、通知内容についての詳細メッセージがある場合、 message と message_code に適切な値が入ります。なお、これらの値の意味するところは使われている ラッパによって異なります。 bytes_transferred と bytes_max には、場合に応じて値が埋められます。*

インストール手順

ストリームは、バージョン 4.3.0 より、PHP の一部として 統合されています。有効にするために特別な手順を踏む必要はありません。

ストリームのクラス

ユーザ定義のラッパは、stream_wrapper_register() で、同関数の説明ページに記載されたクラス定義を使うことで、 登録することができます。

ユーザ定義のフィルタ用に、class php_user_filter が、 ベースとなる抽象クラスとしてあらかじめ定義されています。 ユーザ定義のフィルタの実装に関する詳細は、 stream_filter_register() を参照ください。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

定数	説明
STREAM_FILTER_READ *	stream_filter_append() または stream_filter_prepend() と共に使われます。 指定されたフィルタが *読み込みのときのみ* 適用されることを示します。
STREAM_FILTER_WRITE *	stream_filter_append() または stream_filter_prepend() と共に使われます。 指定されたフィルタが *書き込みのときのみ* 適用されることを示します。
STREAM_FILTER_ALL *	この定数は右の値と等価です: *STREAM_FILTER_READ ¦ STREAM_FILTER_WRITE*
PSFS_PASS_ON *	*返値*: ユーザ空間のフィルタが バケットを *$out* に返したことを 示します。
PSFS_FEED_ME *	*返値*: ユーザ空間のフィルタが *$out* にバケットを返さなかったことを 示します。(つまり、変換されたデータを返す用意ができていないという ことです。)
PSFS_ERR_FATAL *	*返値*: ユーザ空間のフィルタにおいて、 復旧不可能なエラーが発生したことを示します。 (つまり、不正なデータを受け取ったということです。)
STREAM_USE_PATH	*フラグ*: *stream* が include_path を使ったかどうかを示します。
STREAM_REPORT_ERRORS	*フラグ*: ストリームを開く際に、 ユーザ空間にある *wrapper* の側で、 trigger_error() を使ってエラーを発生させるかどうかを 示します。もし、このフラグがセットされていない場合は、 ユーザの側でエラーを投げてはいけません。
STREAM_CLIENT_ASYNC_CONNECT *	非同期的にソケットを開きます。このオプションは、 **STREAM_CLIENT_CONNECT** フラグとともに使用する 必要があります。 stream_socket_client() と共に使われます。
STREAM_CLIENT_CONNECT *	クライアントソケット接続を開きます。クライアントソケットは 常にこのフラグを含んでいる必要があります。 stream_socket_client() と共に使われます。
STREAM_CLIENT_PERSISTENT *	stream_socket_client()を使って 開かれたソケットが、セッションをまたいでも保持されるよう 指示します。
STREAM_SERVER_BIND *	stream_socket_server() で開かれたソケットが 特定のターゲットに関連付けられることを示します。 サーバーソケットには常にこのフラグが与えられているはずです。
STREAM_SERVER_LISTEN *	stream_socket_server() で開かれ、 **STREAM_SERVER_BIND** によってターゲットに 関連付けられたストリームに、ソケットに対して接続待ちを開始するよう 指示します。 接続指向の転送（TCP のような）はこのフラグを使用する必要があります。 それ以外の場合はサーバソケットが有効になりません。 このフラグを非接続指向の転送（UDP のような）に使用するとエラーになります。
STREAM_NOTIFY_RESOLVE *	該当するストリームにおいて、リクエストされたリモートのドメイン名の 解決に成功したか、失敗したことを示します。 このとき、何が起こっているかについては、 *severity* を参照ください。
STREAM_NOTIFY_CONNECT	外部のリソースへの接続が確立されたことを示します。
STREAM_NOTIFY_AUTH_REQUIRED	指定されたリソースにアクセスするためには、さらに認証情報が必要です。 通常に、 **STREAM_NOTIFY_SEVERITY_ERR** の *severity* (深刻度) と共に通知されます。
STREAM_NOTIFY_MIME_TYPE_IS	リソースの *mime タイプ* が確認されました。 どのタイプと判定されたかについては、*message* を見てください。
STREAM_NOTIFY_FILE_SIZE_IS	確認されたリソースの *サイズ*を表します。
STREAM_NOTIFY_REDIRECTED	外部リソースへのアクセスは、別の場所にリダイレクトされました。 詳細を知るには、*message* を調べてみてください。
STREAM_NOTIFY_PROGRESS	ストリームにおける転送の進捗が、 *bytes_transferred* または、場合によっては *bytes_max* に現れていることを示します。
STREAM_NOTIFY_COMPLETED	これ以上ストリームにデータがないことを示します。
STREAM_NOTIFY_FAILURE	ストリームにおいて一般的なエラーが発生したことを示します。 エラーの詳細については、 *message* と *message_code* を調べてみてください。
STREAM_NOTIFY_AUTH_RESULT	認証が完了した(成功または失敗した)ことを示します。
STREAM_NOTIFY_SEVERITY_INFO	正常な状態における、エラーでない通知があることを示します。
STREAM_NOTIFY_SEVERITY_WARN	致命的でないエラーが発生したことを示します。処理は継続されます。
STREAM_NOTIFY_SEVERITY_ERR	致命的なエラーが発生したことを示します。処理は中断されます。

定数	説明
STREAM_IPPROTO_ICMP +	ICMP ソケットを提供します。
STREAM_IPPROTO_IP +	IP ソケットを提供します。
STREAM_IPPROTO_RAW +	RAW ソケットを提供します。
STREAM_IPPROTO_TCP +	TCP ソケットを提供します。
STREAM_IPPROTO_UDP +	UDP ソケットを提供します。
STREAM_PF_INET +	Internet Protocol バージョン 4（IPv4）です。
STREAM_PF_INET6 +	Internet Protocol バージョン 6（IPv6）です。
STREAM_PF_UNIX +	Unix システムの内部プロトコルです。
STREAM_SOCK_DGRAM +	データグラムを提供します。これはコネクションレスのメッセージ（例: UDP）です。
STREAM_SOCK_RAW +	raw ソケットを提供します。これは内部のネットワークプロトコルや インターフェースへのアクセス機能を提供します。通常、この形式の ソケットは root ユーザが使用します。
STREAM_SOCK_RDM +	RDM（Reliably-delivered messages）ソケットを提供します。
STREAM_SOCK_SEQPACKET +	シーケンシャルパケットストリームソケットを提供します。
STREAM_SOCK_STREAM +	帯域外データを転送するための、シーケンシャルで双方向の バイトストリーム（例: TCP）を提供します。
STREAM_SHUT_RD	stream_socket_shutdown() で使用し、 それ以降の受信を無効にします。 PHP 5.2.1 で追加されました。
STREAM_SHUT_WR	stream_socket_shutdown() で使用し、 それ以降の送信を無効にします。 PHP 5.2.1 で追加されました。
STREAM_SHUT_RDWR	stream_socket_shutdown() で使用し、 それ以降の送受信を無効にします。 PHP 5.2.1 で追加されました。

注意: ** がつけられている定数は PHP 5.0.0 以降で使用可能です。*

注意: *+ がつけられている定数は PHP 5.1.0 以降で使用可能で、 stream_socket_pair() とともに使用することが想定されています。 これらの定数の中にはあなたのシステムでは使用できないものがあるかもしれない ことに注意してください。*

ストリームのエラー

*ファイルやソケット関連の関数で処理に失敗するのと同じように、 ストリームにおける操作も様々な予測できる理由で失敗することがあります。（例: リモートホストに接続できない、ファイルが見つからない、等）また、ストリーム関連の関数呼び出しは、 スクリプトの実行されるシステムにおいて、 利用しようとしているストリームが登録されていないことから失敗することも あります。どのタイプのストリームがインストールされた PHP でサポートされているかは、 stream_get_wrappers() の返す配列の内容を参照 してください。なお、他の PHP の内部関数と同様、 **E_WARNING** メッセージがエラー時に出力され、 エラーの原因を報告します。*

例

Example#1 file_get_contents() を使って さまざまな場所からデータを取得する

```
<?php
/* /home/bar にあるローカルファイルを読み出す */
$localfile = file_get_contents("/home/bar/foo.txt");

/* 上と同一だが、明示的に FILE スキームを指定している */
$localfile = file_get_contents("file:///home/bar/foo.txt");

/* HTTP を利用し、www.example.com にあるリモートのファイルを読み出す */
$httpfile  = file_get_contents("http://www.example.com/foo.txt");

/* HTTPS を利用し、www.example.com にあるリモートのファイルを読み出す */
$httpsfile = file_get_contents("https://www.example.com/foo.txt");

/* FTP を利用し、ftp.example.com にあるリモートのファイルを読み出す */
$ftpfile   = file_get_contents("ftp://user:pass@ftp.example.com/foo.txt");

/* FTPS を利用し、ftp.example.com にあるリモートのファイルを読み出す */
$ftpsfile  = file_get_contents("ftps://user:pass@ftp.example.com/foo.txt");
?>
```

Example#2 https のサーバに対して POST リクエストを行う

```
<?php
/* https://secure.example.com/form_action.php に対して POST リクエストを送信
 * ダミー値を持つ "foo" と "bar" というフォーム要素が含まれます。
 */

$sock = fsockopen("ssl://secure.example.com", 443, $errno, $errstr, 30);
if (!$sock) die("$errstr ($errno)¥n");

$data = "foo=" . urlencode("Fooの値") . "&bar=" . urlencode("Barの値");

fwrite($sock, "POST /form_action.php HTTP/1.0¥r¥n");
fwrite($sock, "Host: secure.example.com¥r¥n");
fwrite($sock, "Content-type: application/x-www-form-urlencoded¥r¥n");
fwrite($sock, "Content-length: " . strlen($data) . "¥r¥n");
fwrite($sock, "Accept: */*¥r¥n");
fwrite($sock, "¥r¥n");
fwrite($sock, "$data¥r¥n");
fwrite($sock, "¥r¥n");

$headers = "";
while ($str = trim(fgets($sock, 4096)))
  $headers .= "$str¥n";

echo "¥n";
```

```
$body = "";
while (!feof($sock))
  $body .= fgets($sock, 4096);

fclose($sock);
?>
```

Example#3 データを圧縮ファイルに書き込む

```
<?php
/* 任意の文字列を含む圧縮ファイルを作成
 * ファイルは、compress.zlib ストリームを使っても、または単に
 * コマンドラインで 'gzip -d foo-bar.txt.gz' を使っても読み出せます。
 */
$fp = fopen("compress.zlib://foo-bar.txt.gz", "wb");
if (!$fp) die("ファイルが作成できませぬ.");

fwrite($fp, "これはテストです.\n");

fclose($fp);
?>
```

---

stream_bucket_append

(PHP 5)

stream_bucket_append — bucket を brigade に追加する

説明

void **stream_bucket_append** (resource $brigade , resource $bucket)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

stream_bucket_make_writeable

(PHP 5)

stream_bucket_make_writeable — 操作する brigade から bucket オブジェクトを返す

説明

object **stream_bucket_make_writeable** (resource $brigade)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

stream_bucket_new

(PHP 5)

stream_bucket_new — 現在のストリームで使用する新しい bucket を作成する

説明

object **stream_bucket_new** (resource $stream , string $buffer)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

stream_bucket_prepend

(PHP 5)

stream_bucket_prepend — bucket を brigade に追加する

説明

void **stream_bucket_prepend** (resource $brigade , resource $bucket)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

stream_context_create

(PHP 4 >= 4.3.0, PHP 5)

stream_context_create — ストリームコンテキストを作成する

説明

resource **stream_context_create** ([array $options [, array $params]])

options に、 指定されたオプションが予め設定されたストリームコンテキストを作成し、 それを返します。

options は、 次のような形式の連想配列からなる連想配列でなくてはなりません。 $arr['wrapper']['option'] = $value. デフォルトは空の配列です。

params は、 $arr['parameter'] = $value 形式の連想配列でなくてばなりません。 標準のストリームパラメータについては stream_context_set_params() を参照ください。

注意: 引数 params は PHP 6.0.0 で追加されました。

Example#1 stream_context_create() を利用する

```
<?php
$opts = array(
  'http'=>array(
    'method'=>"GET",
    'header'=>"Accept-language: en¥r¥n" .
              "Cookie: foo=bar¥r¥n"
  )
);

$context = stream_context_create($opts);

/* 上のヘッダと共に http リクエストを www.example.com に対して
   送出します */
$fp = fopen('http://www.example.com', 'r', false, $context);
fpassthru($fp);
fclose($fp);
?>
```

stream_context_set_option() と コンテキストオプションをサポートするラッパのリスト（サポートされるプロトコル/ラッパー）も参照ください。

---

stream_context_get_default

(PHP 5 >= 5.1.0)

stream_context_get_default — デフォルトのストリームコンテキストを取得する

説明

resource **stream_context_get_default** ([array $options])

（fopen()、file_get_contents() のような） ファイル操作関数がコンテキストパラメータなしでコールされた場合に使用される デフォルトのストリームコンテキストを返します。デフォルトコンテキストに 関するオプションは、stream_context_create() と同じ 構文で任意に指定することが可能です。

options は、 $arr['wrapper']['option'] = $value のような形式の、連想配列の連想配列である必要があります。

Example#1 stream_context_get_default() の使用

```
<?php
$default_opts = array(
  'http'=>array(
    'method'=>"GET",
    'header'=>"Accept-language: en¥r¥n" .
              "Cookie: foo=bar",
    'proxy'=>"tcp://10.54.1.39:8000"
  )
);


$alternate_opts = array(
  'http'=>array(
    'method'=>"POST",
    'header'=>"Content-type: application/x-www-form-urlencoded¥r¥n" .
              "Content-length: " . strlen("baz=bomb"),
    'content'=>"baz=bomb"
  )
);

$default = stream_context_get_default($default_opts);
$alternate = stream_context_create($alternate_opts);

/* www.example.com に対する通常の GET リクエストを、プロキシサーバ 10.54.1.39
 * に対して送信します。コンテキストオプション $default_opts を使用します。
 */
readfile('http://www.example.com');

/* POST リクエストを、直接 www.example.com に送信します。
 * コンテキストオプション $alternate_opts を使用します。
 */
readfile('http://www.example.com', false, $alternate);

?>
```

stream_context_create() および サポートされるラッパ（サポートされるプロトコル/ラッパー） も参照ください。

---

stream_context_get_options

(PHP 4 >= 4.3.0, PHP 5)

stream_context_get_options — ストリーム / ラッパ / コンテキストに設定されているオプションを取得する

説明

array **stream_context_get_options** (resource $stream_or_context)

指定されたストリームまたはコンテキストに設定されたオプションを配列 として返します。

---

stream_context_set_option

(PHP 4 >= 4.3.0, PHP 5)

stream_context_set_option — ストリーム / ラッパ / コンテキストのオプションを設定する

説明

bool **stream_context_set_option** (resource $stream_or_context , string $wrapper , string $option , mixed $value)
bool **stream_context_set_option** (resource $stream_or_context , array $options)

指定されたストリームまたはコンテキストのオプションを設定します。 value の内容が wrapper に対する option として設定されます。

---

stream_context_set_params

(PHP 4 >= 4.3.0, PHP 5)

stream_context_set_params — ストリーム / ラッパ / コンテキストのパラメータを設定する

説明

bool **stream_context_set_params** (resource $stream_or_context , array $params)

params は、次のような形式の連想配列でなくてはなりません: $params['paramname'] = "paramvalue";.

設定できるパラメータの一覧

パラメータ	目的
notification	ストリームから何らかの通知があった時に呼ばれる、ユーザ定義のコールバック関数を指定します。
options	stream_context_create() へのオプションの配列を指定します。

---

stream_copy_to_stream

(PHP 5)

stream_copy_to_stream — データをあるストリームから別のストリームにコピーする

説明

int **stream_copy_to_stream** (resource $source , resource $dest [, int $maxlength [, int $offset]])

現在の位置（あるいはもし指定されていれば offset の位置）から最大 maxlength バイトのデータを source から dest にコピーします。もし maxlength が指定されていない 場合は、source にある残りすべてのデータが コピーされます。

パラメータ

source

コピー元のストリーム。

dest

コピー先のストリーム。

maxlength

コピーする最大バイト数。

offset

コピーを開始する位置。

返り値

コピーされたバイト数を返します。

変更履歴

バージョン	説明
5.1.0	*offset* パラメータが追加されました。

例

Example#1 stream_copy_to_stream() の例

```
<?php
$src = fopen('http://www.example.com', 'r');
$dest1 = fopen('first1k.txt', 'w');
$dest2 = fopen('remainder.txt', 'w');

echo stream_copy_to_stream($src, $dest1, 1024) . " バイトが first1k.txt にコピーされました\n";
echo stream_copy_to_stream($src, $dest2) . " バイトが remainder.txt にコピーされました\n";

?>
```

参考

- copy()

---

stream_encoding

(No version information available, might be only in CVS)

stream_encoding — ストリームのエンコード用の文字セットを設定する

説明

bool **stream_encoding** (resource $stream [, string $encoding])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

stream_filter_append

(PHP 4 >= 4.3.0, PHP 5)

stream_filter_append — ストリームにフィルタを付加する

説明

resource **stream_filter_append** (resource $stream , string $filtername [, int $read_write [, mixed $params]])

filtername で指定されたフィルタを、 stream に付加されているフィルタのリストに加えます。 このフィルタは、指定された params と共に、リストの末尾に追加され、ストリームに対する操作の中で最後に呼び出されます。 フィルタをリストの先頭に加えたいときは、stream_filter_prepend() を使ってください。

デフォルトでは、 **stream_filter_append()** は ストリームが読み込み用に開かれている場合は (つまり、オープンモードが r あるいは + を伴う場合は)、 フィルタを リードフィルタチェイン に追加し、 ストリームが書き出し用に開かれている場合は(つまり、オープンモードが w か aか、あるいは+ を伴う場合は)、 ライトフィルターチェィンにも追加します。 **STREAM_FILTER_READ**・ **STREAM_FILTER_WRITE**・ **STREAM_FILTER_ALL** を read_write パラメータに渡すことで、この挙動を変えることができます。

PHP 5.1.0 以降では、この関数はリソースを返します。 このリソースは、stream_filter_remove() をコールする際にこのフィルタインスタンスを参照するために使用可能です。 PHP 5.1.0 より前のバージョンでは、この関数は成功した場合に **TRUE**、 失敗した場合に **FALSE** を返します。

Example#1 フィルタの適用される場所を制御する

```
<?php
/* ファイルを読み書き用に開く */
$fp = fopen('test.txt', 'w+');

/* ROT13 フィルタをライトフィルタチェインに付加する。
 * リードフィルタチェインには付加しない。*/
stream_filter_append($fp, "string.rot13", STREAM_FILTER_WRITE);

/* 単純な文字列をファイルに書き出す。
 * この文字列には、出口で ROT13 変換が適用される。
 */
fwrite($fp, "This is a test¥n");

/* ファイルの最初に戻る */
rewind($fp);

/* 書き出した内容を読み戻す。
 * もし、フィルタがリードフィルタチェインにも
 * 付加されていれば、再び読み出し時に ROT13 が適用され、
 * テキストは元の状態に戻るはず。 */
fpassthru($fp);

fclose($fp);

/* 期待される出力
   --------------

Guvf vf n grfg

 */
```

```
?>
```

注意: **カスタム(ユーザ)フィルタを使うには** カスタムフィルタを filtername に指定するためには、 まず stream_filter_register() 関数でそれを登録しておく 必要があります。

注意: ストリームデータは（ローカルおよびリモートの）リソースからチャンク単位で 読み込まれ、内部バッファに保持されます。新しいフィルタがストリームに 追加されると、内部バッファのデータがその時点でフィルタリングされます。 これは stream_filter_prepend() の挙動とは違います。

stream_filter_register()、 stream_filter_prepend() および stream_get_filters() も参照ください。

---

stream_filter_prepend

(PHP 4 >= 4.3.0, PHP 5)

stream_filter_prepend — フィルタをストリームに付加する

説明

resource **stream_filter_prepend** (resource $stream , string $filtername [, int $read_write [, mixed $params]])

filtername で指定されたフィルタを、 stream に付加されているフィルタのリストに加えます。 このフィルタは、指定された params と共に、リストの先頭に追加され、ストリームに対する操作の中で最初に呼び出されます。 フィルタをリストの末尾に加えたいときは、stream_filter_append() を使ってください。

デフォルトでは、 **stream_filter_prepend()** は ストリームが読み込み用に開かれている場合は（つまり、オープンモードが r あるいは + を伴う場合は)、 フィルタを リードフィルタチェイン に追加し、 ストリームが書き出し用に開かれている場合は(つまり、オープンモードが w か aか、あるいは+ を伴う場合は)、 ライトフィルターチェィンにも追加します。 **STREAM_FILTER_READ**・ **STREAM_FILTER_WRITE**・ **STREAM_FILTER_ALL** を read_write パラメータに渡すことで、この挙動を変えることができます。 このパラメータを使った例については、__stream_filter_append() を参照ください。

PHP 5.1.0 以降、この関数はリソースを返します。このリソースは stream_filter_remove() のコール時にこの フィルタのインスタンスを指定するために使用可能です。 PHP 5.1.0 より前のバージョンでは、この関数は 成功した場合に **TRUE**、失敗した場合に **FALSE** を返します。

注意: **カスタム(ユーザ)フィルタを使うには** カスタムフィルタを filtername に指定するためには、 まず stream_filter_register() 関数でそれを登録しておく 必要があります。

注意: ストリームデータは（ローカルおよびリモートの）リソースからチャンク単位で 読み込まれ、内部バッファに保持されます。新しいフィルタがストリームに 追加されても、内部バッファにあるデータのうち既に他のフィルタの処理が 終わっているものは新しいフィルタでは処理されません。 これは **stream_filter_apppend()** の挙動とは違います。

stream_filter_register() および stream_filter_append() も参照ください。

---

stream_filter_register

(PHP 5)

stream_filter_register — php_user_filter に由来するクラスとして実装されたストリームフィルタを登録する

説明

bool **stream_filter_register** (string $filtername , string $classname)

stream_filter_register() は、fopen() や fread() などのファイルシステムの関数で利用可能な 登録されているどの種類のストリームとも一緒に使うことのできる カスタムフィルタを登録します。

フィルタを実装するには、まず、php_user_fitler を継承したクラスの下記に示されたメンバ関数を実装しなくてはなりません。 PHP は、書き込みまたは読み出し操作を カスタムフィルタの付加されたストリームに対して行う際に、まず データをそのフィルタに(そして、付加されている他のすべてのフィルタにも）渡し、指示どおりストリームのデータが変更されるようにします。 下記のメソッドは、説明の通り実装されなくてはなりません。さもないと、定義されていない動作をします。

stream_filter_register() は、 指定された filtername がすでに定義されている場合、 **FALSE** を返します。

int **filter** (resource $in , resource $out , int &$consumed , bool $closing)

このメソッドは、fread() や fwrite() 等の関数で、付加されたストリームからデータを読み出す時や、 データを書き込む度に呼ばれます。 in は、bucket brigade (バケツリレー隊）を指すポインタです。 これは、フィルタの対象になるデータを含む複数の bucket オブジェクトから成っています。 out は、変更されたバケットが渡される bucket brigade を示しています。 consumed は、参照渡しされるパラメータで、 ここには、フィルタが実際に処理した元のデータ長を加算します。 ほとんどの場合、それぞれのバケットについて、そのデータ長 $bucket->datalen を、そこに足すだけでいいはずです。 もし、ストリームが閉じようとしている時（すなわちフィルタ連鎖の中の 最後の呼び出しだった時)、closing パラメータは、 **TRUE** となります。 **filter** パラメータは、さらに、 次の値のいずれかを終了時に返さなくてはなりません。

返り値	意味
PSFS_PASS_ON	フィルタの処理に成功し、 out の バケツ隊 に データが渡されました。
PSFS_FEED_ME	フィルタの処理に成功しましたが、データは返されませんでした。 ストリームあるいは前段階のフィルタに対してさらにデータが必要です。
PSFS_ERR_FATAL（デフォルト）	フィルタにおいて復旧不可能なエラーが発生し、処理を継続できません。

bool **onCreate** (void)

このメソッドは、フィルタクラスのオブジェクトが実体化されるときに 呼び出されます。もし、フィルタがバッファなど他のリソースを 確保したり初期化しなくてはならない場合、この時点で行ってください。 このメソッドを実装する際には、失敗した場合に **FALSE**、成功した場合に **TRUE** を返すようにする必要があります。

最初にフィルタのインスタンスが作成され yourfilter->onCreate() がコールされた際に、 以下の表に示す多くのプロパティが使用可能となります。

プロパティ	内容
FilterClass->filtername	インスタンス化されたフィルタの名前を含む文字列。 フィルタは複数の名前で登録されりワイルドカードで登録されたり することもありえます。どのような名前が使われたのか、 このプロパティで調べます。
FilterClass->params	stream_filter_append() あるいは stream_filter_prepend() に渡すパラメータの内容。

void ***onClose*** *(void)*

このメソッドは、フィルタが遮断される時(通常、ストリームが遮断 される時)に呼ばれます。また、同様に、flush メソッドが 呼ばれた後に呼び出されます。 もし、何らかのリソースが、oncreate メソッドの 呼び出し時に確保されていた場合は、それらをここで廃棄するのが いいでしょう。

下記の例は、読み込まれたり書き出されたりするデータに含まれるすべての英文字を 大文字化する strtoupper ストリームを実装し、foo-bar.txt ストリームに適用するものです。

Example#1 foo-bar.txt ストリームの文字を大文字化するフィルタ

```php
<?php

/* フィルタクラスを定義する */
class strtoupper_filter extends php_user_filter {
  function filter($in, $out, &$consumed, $closing)
  {
    while ($bucket = stream_bucket_make_writeable($in)) {
      $bucket->data = strtoupper($bucket->data);
      $consumed += $bucket->datalen;
      stream_bucket_append($out, $bucket);
    }
    return PSFS_PASS_ON;
  }
}

/* PHP にフィルタを登録する */
stream_filter_register("strtoupper", "strtoupper_filter")
    or die("Failed to register filter");

$fp = fopen("foo-bar.txt", "w");

/* フィルタを開いたストリームに付加する */
stream_filter_append($fp, "strtoupper");

fwrite($fp, "Line1¥n");
fwrite($fp, "Word - 2¥n");
fwrite($fp, "Easy As 123¥n");

fclose($fp);

/* ファイルを読み出し出力する。
 */
readfile("foo-bar.txt");

?>
```

上の例の出力は以下となります。

```
LINE1
WORD - 2
EASY AS 123
```

Example#2 複数のフィルタ名に対応する一般的なフィルタクラスを登録する

```php
<?php

/* フィルタクラスを定義する */
class string_filter extends php_user_filter {
  var $mode;

  function filter($in, $out, &$consumed, $closing)
  {
    while ($bucket = stream_bucket_make_writeable($in)) {
      if ($this->mode == 1) {
        $bucket->data = strtoupper($bucket->data);
      } elseif ($this->mode == 0) {
        $bucket->data = strtolower($bucket->data);
      }

      $consumed += $bucket->datalen;
      stream_bucket_append($out, $bucket);
    }
    return PSFS_PASS_ON;
  }

  function onCreate()
  {
    if ($this->filtername == 'str.toupper') {
      $this->mode = 1;
    } elseif ($this->filtername == 'str.tolower') {
      $this->mode = 0;
    } else {
     /* その他の str.* フィルタが問い合わせられた場合は
          失敗を報告し、PHP が検索を続けられるようにする */
      return false;
    }

    return true;
  }
```

```
}

/* PHP にフィルタを登録する */
stream_filter_register("str.*", "string_filter")
    or die("Failed to register filter");

$fp = fopen("foo-bar.txt", "w");

/* フィルタを開いたストリームに付加する
   ここで str.tolower をバインドすることも可能 */
stream_filter_append($fp, "str.toupper");

fwrite($fp, "Line1¥n");
fwrite($fp, "Word - 2¥n");
fwrite($fp, "Easy As 123¥n");

fclose($fp);

/* ファイルを読み出し出力する。
 */
readfile("foo-bar.txt");
?>
```

上の例の出力は以下となります。

```
LINE1
WORD - 2
EASY AS 123
```

stream_wrapper_register()、 stream_filter_prepend() および stream_filter_append() も参照ください。

---

stream_filter_remove

(PHP 5 >= 5.1.0)

stream_filter_remove — ストリームからフィルタを取り除く

説明

bool **stream_filter_remove** (resource $stream_filter)

事前に stream_filter_prepend() あるいは stream_filter_append() でストリームに追加した フィルタを削除します。フィルタの内部バッファに残っているデータは 削除前にフラッシュされ、次のフィルタに渡されます。

Example#1 ストリームの再フィルタリングを動的に行う

```
<?php
/* テストファイルを読み込み/書き込みモードでオープンします */
$fp = fopen("test.txt", "rw");

$rot13_filter = stream_filter_append($fp, "string.rot13", STREAM_FILTER_WRITE);
fwrite($fp, "This is ");
stream_filter_remove($rot13_filter);
fwrite($fp, "a test¥n");

rewind($fp);
fpassthru($fp);
fclose($fp);

/* 想定される出力
     --------------

Guvf vf a test

 */
?>
```

stream_filter_register()、 stream_filter_append() および stream_filter_prepend() も参照ください。

---

stream_get_contents

(PHP 5)

stream_get_contents — 残りのストリームを文字列に読み込む

説明

string **stream_get_contents** (resource $handle [, int $maxlength [, int $offset]])

file_get_contents() と似ていますが、 **stream_get_contents()** は既にオープンしている ストリームリソースに対して操作を行います。そして、指定した offset から始まる最大 maxlength バイトのデータを取得して文字列に 保存します。

パラメータ

handle (resource)

ストリームリソース（例: fopen() の返す値）。

maxlength (integer)

読み込む最大バイト数。デフォルトは -1 （バッファの残りのデータをすべて読み込む）。

offset (integer)

読み込みを開始する前に移動する位置。PHP 5.1.0 で追加されました。

返り値

文字列を返します。失敗した場合は FALSE を返します。

例

Example#1 stream_get_contents() の例

```
<?php

if ($stream = fopen('http://www.example.com', 'r')) {
    // オフセット 10 から開始して、残りのすべてのページを表示します
    echo stream_get_contents($stream, -1, 10);

    fclose($stream);
}


if ($stream = fopen('http://www.example.net', 'r')) {
    // 最初の 5 バイトを表示します
    echo stream_get_contents($stream, 5);

    fclose($stream);
}

?>
```

参考

- fgets()
- fread()
- fpassthru()

注意: この関数はバイナリデータに対応しています。

---

stream_get_filters

(PHP 5)

stream_get_filters — 登録されているフィルタのリストを取得する

説明

array **stream_get_filters** (void)

すべてのランタイムにおいて有効なストリームフィルタの名前を含む 配列を返します。

Example#1 stream_get_filters()を使用する

```
<?php
$streamlist = stream_get_filters();
print_r($streamlist);
?>
```

出力は以下のようになります。 注意: PHP のバージョンによって、フィルタの数は増減します。

```
Array (
   [0] => string.rot13
   [1] => string.toupper
   [2] => string.tolower
   [3] => string.base64
   [4] => string.quoted-printable
)
```

stream_filter_register() および stream_get_wrappers() も参照ください。

---

stream_get_line

(PHP 5)

stream_get_line — 指定されたデリミタの位置までのデータを一行分としてストリームから読み込む

説明

string **stream_get_line** (resource $handle , int $length [, string $ending])

最大 length バイトの、 handle で指定されたリソースから読み込んだデータ を返します。読み込みは、length バイト読まれたか、 ending で指定された文字列がストリームに見つかった か（この文字列は返値に 含まれません）、あるいは EOF に達したとき、いずれかの条件のもとで停止します。

エラーが発生した際には、**FALSE** を返します。

この関数は fgets() とほとんど同一ですが、 \n や \r 、\r\n といった一般的な文字列以外を行末を示すデリミタ として指定できる点で、またデリミタ自体を返値に 含まない 点で異なります。

fread()、 fgets() および fgetc() も参照ください。

---

stream_get_meta_data

(PHP 4 >= 4.3.0, PHP 5)

stream_get_meta_data — ヘッダーあるいはメタデータをストリームまたはファイルポインタから取得する

説明

array **stream_get_meta_data** (resource $stream)

既存の stream に関する情報を取得します。 ストリームは fopen() か、 fsockopen() か、pfsockopen() で 作成されたいずれのものも指定できます。 結果の配列は次のような項目を含みます。

- timed_out (bool) - 最後に fread() または fgets() でデータを待っている時にタイムアウトした場合 **TRUE** を返します。
- blocked (bool) - ストリームがブロック I/O モードの場合に **TRUE** となります。 stream_set_blocking() を参照ください。
- eof (bool) - ストリームが EOF に 達した時 **TRUE** となります。 ストリームがソケットベースの場合、このメンバーは、 たとえ unread_bytes が 0 でなくても **TRUE** になる場合があることに注意してください。 まだデータがあるかどうかを調べるには、このパラメータではなく、 feof() を使ってください。
- unread_bytes (int) - PHP の 内部バッファにあるデータのバイト数。

  注意: スクリプト中でこの値を使用すべきではありません。

次の項目は PHP 4.3.0 で追加されました。

- stream_type (string) - ストリームの下層にある実装を表すラベル
- wrapper_type (string) - ストリームを覆うプロトコルラッパを表すラベル。 ラッパについては サポートされるプロトコル/ラッパー を参照ください。
- wrapper_data (mixed) - ストリームに付随しているラッパの固有のデータ。 ラッパとその固有の情報については、サポートされるプロトコル/ラッパー を参照ください。
- filters (array) - ストリームに付加されているフィルタの名称を格納した配列。 フィルタに関するドキュメントは 利用できるフィルタのリスト にあります。

  注意: この関数は PHP 4.3.0 で初めて導入されましたが、 それ以前には ソケットベースのストリーム専用の stream_get_status() を使い、 最初に挙げた 4 つの項目に関しては取得できました。
  PHP 4.3.0 またはそれ以降では、 stream_get_status() はこの関数のエイリアスと なっています。

  注意: この関数は ソケット関数 で作られたストリームに対しては機能しません。

次の項目は PHP 5.0.0 で追加されました。

- mode (string) - このストリームに要求される アクセスモード（fopen() リファレンスの表 1 を参照ください）。
- seekable (bool) - 現在のストリーム内で 移動が可能かどうか。
- uri (string) - このストリームに関連付けられた URI / ファイル名。

---

stream_get_transports

(PHP 5)

stream_get_transports — 登録されたソケットのトランスポートの一覧を取得する

説明

array **stream_get_transports** (void)

スクリプトを走らせているシステムにおいて利用可能なソケットのトランスポートのリストを配列として返します。

Example#1 stream_get_transports() を使った例

```
<?php
$xportlist = stream_get_transports();
print_r($xportlist);
?>
```

出力は以下のようになります。 注意: PHP のバージョンによって、トランスポートの数が増減することがあります。

```
Array (
   [0] => tcp
   [1] => udp
   [2] => unix
   [3] => udg
)
```

stream_get_filters(), stream_get_wrappers() も参照ください。

---

stream_get_wrappers

(PHP 5)

stream_get_wrappers — 登録されているストリームのラッパのリストを取得する

説明

array **stream_get_wrappers** (void)

スクリプトを走らせているシステム上で使うことのできるすべてのストリーム ラッパの名前を配列の形で返します。

Example#1 stream_get_wrappers() の例

```
<?php
print_r(stream_get_wrappers());
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => php
     [1] => file
     [2] => http
     [3] => ftp
      [4] => compress.bzip2
      [5] => compress.zlib
)
```

Example#2 ストリームラッパのが存在するかどうかを調べる

```
<?php
// bzip2 ストリームラッパが存在するかどうかを調べます
if (in_array('compress.bzip2', stream_get_wrappers())) {
    echo 'compress.bzip2://  サポートが有効です。';
} else {
    echo 'compress.bzip2://  サポートは有効ではありません。';
}
?>
```

stream_wrapper_register() も参照ください。

---

stream_register_wrapper

(PHP 4 >= 4.3.0, PHP 5)

stream_register_wrapper — stream_wrapper_register() へのエイリアス

説明

この関数は stream_wrapper_register() へのエイリアスです。 PHP 4.3.0 と PHP 4.3.1 のみに 対する互換性の保持のために存在します。 通常は stream_wrapper_register() を使用するべきです。

---

stream_resolve_include_path

(No version information available, might be only in CVS)

stream_resolve_include_path — fopen() に相対パスを指定してコールされたときに、 どのファイルをオープンするかを決める

説明

string **stream_resolve_include_path** (string $filename [, resource $context])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

stream_select

(PHP 4 >= 4.3.0, PHP 5)

stream_select — select() システムコールと同等の操作を、 ストリームの配列に対して tv_sec と tv_usec で指定されたタイムアウト時間をもって行う

説明

int **stream_select** (array &$read , array &$write , array &$except , int $tv_sec [, int $tv_usec])

stream_select() はストリームの配列を受け取ると、 それらの状態が変化するまで待ちます。機能としては、ストリームに対して 働くという点以外では socket_select() と同一です。

read 配列に列挙されたストリームに対しては、 何らかのデータがそのストリーム内で読み出せる状態にあるかどうか 監視が行われます（より正確にいえば、ブロックしないで読み出せる状態かどうか - 特にストリームが EOF に達したかどうか、です。このとき、 fread() は長さ 0 の文字列を返します)。

write 配列に列挙されたストリームに対しては、 ブロックしないで書き込みができるかどうかの監視が行われます。

except 配列に列挙されたストリームに対しては、 重大な例外（"帯域外の"）データが発生したかどうかの監視が行われます。

> **注意**: **stream_select()** の終了時には、 どのストリームの状態が実際に変化したのかが分かるよう、 配列 read 、write および except に変更が加えられます。

tv_sec と tv_usec は、一体となって、timeout パラメータを表現します。 tv_sec は秒数を指定し、一方 tv_usec はマイクロ秒数を指定します。 timeout は、**stream_select()** の実行から戻るまでの時間の上限です。 tv_sec および tv_usec の両方に 0 を指定すると **stream_select()** はデータを待たずに一瞬で戻ります。 これは現在のストリームの状態を示します。 tv_sec が **NULL**（タイムアウトなし）の場合、 **stream_select()** はブロックしつづけ、調べている ストリームのひとつでイベントが発生する（あるいはシグナルがシステム コールを中断する）まで終了しません。

成功した場合 **stream_select()** は、変更された 配列に何個のストリームリソースが格納されたかを示す数を返します。 もしタイムアウトの時間内に何も規定された事象が起こらなかった場合は 0 になることもあります。エラーの際は **FALSE** を返し、警告を発生させます （システムコールが別のシグナルによって中断された場合などに起こりえます）。

警告

タイムアウト値に 0 を指定すると、ストリームの 状態を即時に取得することが可能です。しかし、ループ内でタイムアウト 0 を指定するのは良い考えではありません。 そうすると大量の CPU 時間を消費してしまいます。

タイムアウト値を数秒にするとかなりましになります。しかし、どうしても 他のコードを同時に実行させながらチェックをする必要がある場合には、少なくとも 200000 マイクロ秒以上のタイムアウトを 設定するようにしましょう。これであなたのスクリプトの CPU 使用量を 抑えることができます。

タイムアウト値は、あくまでも経過時間の最大値であることを覚えておきましょう。 **stream_select()** は、指定したストリームが使用可能に なるとすぐに結果を返します。

stream_select() のすべてのパラメータに配列を 渡す必要はありません。代わりに そのままにしておくことも、空の配列を渡すことも、**NULL** を渡すことも できます。このとき、それらの変数は参照渡しで渡されるため、 **stream_select()** から戻った時点で変更されている 可能性があることに注意してください。

この例では、$stream1 あるいは $stream2 のどちらかに読み込めるデータが 到達したかどうかを調べます。 タイムアウトが 0 なので、すぐに結果を返します。

```
<?php
/* read 配列を用意 */
$read   = array($stream1, $stream2);
$write  = NULL;
$except = NULL;
if (false === ($num_changed_streams = stream_select($read, $write, $except, 0))) {
    /* エラー処理 */
} elseif ($num_changed_streams > 0) {
    /* 少なくとも 1 つのストリームに何らかの事象が起こりました。*/
}
?>
```

> **注意**: 現在の Zend Engine の実装上の制約により、**NULL** のような定数を この関数の参照渡しが行われるパラメータに直接指定することはできません。 代わりに一時的な変数を指定するか、一番左の変数が一時的な変数になるような 式を指定してください:
>
> ```
> <?php
> $e = NULL;
> stream_select($r, $w, $e, 0);
> ?>
> ```

> **注意**: エラーかどうかをチェックするには === を使ってください。 **stream_select()** は 0 を返すことがあるため、その場合 **FALSE** と == 演算子で比較すると **TRUE** と評価されてしまうからです:
>
> ```
> <?php
> $e = NULL;
> if (false === stream_select($r, $w, $e, 0)) {
>     echo "stream_select() に失敗しました\n";
> }
> ?>
> ```

> **注意**: 配列に返されたストリームに対して読み込みまたは書き込み操作を行う際に、 必ずしもあなたの希望しただけデータが読まれたり書かれたりはしないことに 注意してください。たった 1 バイトしか読み出せない場合も、書き込めない 場合もあるのです。

> **注意**: Windows の互換性: Windows 98 では、 proc_open() が返すパイプに対して **stream_select()** を使用すると データが失われる可能性があります。
> Windows では、proc_open() が返すファイル記述子に 対して **stream_select()** を使用すると 失敗して **FALSE** を返すことがあります。

stream_set_blocking() も参照ください。

---

stream_set_blocking

(PHP 4 >= 4.3.0, PHP 5)

stream_set_blocking — ストリームのブロックモードを有効にする / 解除する

説明

bool **stream_set_blocking** (resource $stream , int $mode)

mode が 0 の時、ストリームは 非ブロックモードに切り替えられ、1 の場合は、 ブロックモードに切り替えられます。このモードの違いは、fgets() や fread() といった、ストリームからデータを読む関数に影響します。 非ブロックモードにおいては fgets() を呼び出すと どんな場合でもただちに呼び出し元に戻りますが、フロッグモードの場合では、 ストリームがデータを読み出せる状態になるまで待ちつづけます。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

以前、この関数は、 **set_socket_blocking()** または、後に socket_set_blocking() と呼ばれていましたが、 これらの関数は廃止されました。

> **注意**: PHP 4.3 より前のバージョンでは、この関数は ソケットベースのストリームにしか機能しませんでした。 PHP 4.3 より、この関数は、非ブロックモードを サポートするすべてのストリームにおいて利用可能となりました。(現在サポートするのは、通常のファイルストリームとソケットストリームのみです。)

stream_select() も参照ください。

---

stream_set_timeout

(PHP 4 >= 4.3.0, PHP 5)

stream_set_timeout — ストリームにタイムアウトを設定する

説明

bool **stream_set_timeout** (resource $stream , int $seconds [, int $microseconds])

stream にタイムアウトの値を設定します。 この値は、seconds と microseconds の和で表されます。成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

ストリームがタイムアウトとなった場合は、 stream_get_meta_data() が返す配列のキー 'timed_out' の値が **TRUE** に設定されます。エラーや警告が発生していなくても 同様になります。

Example#1 stream_set_timeout() の例

```
<?php
$fp = fsockopen("www.example.com", 80);
if (!$fp) {
    echo "開けません\n";
} else {

    fwrite($fp, "GET / HTTP/1.0¥r¥n¥r¥n");
    stream_set_timeout($fp, 2);
    $res = fread($fp, 2000);

    $info = stream_get_meta_data($fp);
    fclose($fp);

    if ($info['timed_out']) {
        echo 'Connection timed out!';
    } else {
        echo $res;
    }

}
?>
```

> **注意**: PHP 4.3 より、この関数は、(潜在的には)どの種類の ストリームに対しても機能するようになりました。 PHP 4.3 では、ソケットベースのストリームが、 唯一この関数でサポートされている種類でしたが、他の拡張モジュール由来の モジュールはこの機能をサポートしているかもしれません。
>
> **注意**: この関数では、stream_socket_recvfrom() のような 高度な操作はできません。そのかわりに、timeout パラメータを指定して stream_select() を使用してください。

この関数は、以前は **set_socket_timeout()** 、その後は socket_set_timeout() と呼ばれたこともありましたが、 これらの利用は推奨されません。

fsockopen() と fopen() も参照ください。

---

stream_set_write_buffer

(PHP 4 >= 4.3.0, PHP 5)

stream_set_write_buffer — 指定されたストリームのファイルバッファリングを有効にする

説明

int **stream_set_write_buffer** (resource $stream , int $buffer)

fwrite() による出力は、通常では 8K バイトがバッファ されます。 これは、もし同じストリームに対し出力を行おうとするプロセスが2つあったとき、 いずれかのプロセスは、他方のプロセスが出力できるように 8K バイト分 データを書き出したところで停止することを示しています。 **stream_set_write_buffer()** は、stream で指定されたファイルポインタに buffer で表されたバイト数分だけ出力バッファを設定します。 buffer が 0 であれば、書き込み操作はバッファされなく なります。 これにより、fwrite() による書き込み操作が、他の プロセスが同じ出力ストリームに対して何らかの書き込み操作を行う前に 完了することが保証されます。

この関数は、成功時に 0 を、要求通りに設定できなかった場合は EOF を返します。

次の例は、バッファされていないストリームを **stream_set_write_buffer()** によって 作成する方法を示したものです。

Example#1 stream_set_write_buffer() の例

```
<?php
$fp = fopen($file, "w");
if ($fp) {
  stream_set_write_buffer($fp, 0);
  fwrite($fp, $output);
  fclose($fp);
}
?>
```

fopen(), fwrite() も参照ください。

---

stream_socket_accept

(PHP 5)

stream_socket_accept — stream_socket_server() で作られたソケットの接続を受け入れる

説明

resource **stream_socket_accept** (resource $server_socket [, float $timeout [, string &$peername]])

以前に stream_socket_server() によって作られた ソケットの接続を受け入れます。もし、timeout が指定されていると、通常の接続待ちタイムアウトは、秒数で表された その値で上書きされます。接続元のクライアントの名前（アドレス）は パラメータが指定されていて、かつトランスポートにおいて有効であった場合のみ、 peername に返します。

peername は、 後で、stream_socket_get_name() を使っても 取得できます。

実行に失敗すると、**FALSE** を返します。

警告

この関数は UDP サーバソケットとともに使用すべきではありません。 代わりに stream_socket_recvfrom() および stream_socket_sendto() を使用します。

stream_socket_server(), stream_socket_get_name(), stream_set_blocking(), stream_set_timeout(), fgets(), fgetss(), fwrite(), fclose(), feof(), Curl 拡張モジュール も参照ください。

---

stream_socket_client

(PHP 5)

stream_socket_client — インターネットドメインまたは Unix ドメインのソケット接続を開く

説明

resource **stream_socket_client** (string $remote_socket [, int &$errno [, string &$errstr [, float $timeout [, int $flags [, resource $context]]]]])

remote_socket で指定された接続先との、 ストリームまたはデータグラム接続を確立します。 作成されるソケットのタイプは、[トランスポート]://[ターゲット] という形式の URL フォーマットによって指定された トランスポートによって決定されます。 TCP や UDP といったインターネットドメインのソケット (AF_INET) には、remote_socket パラメータの ターゲット の部分は、ホスト名または IP アドレスと、 それに続くコロンで区切られたポート番号から構成されていなければなりません。 Unix ドメインのソケットの場合は、ターゲット の部分は、ファイルシステムにおけるソケットのファイルを指定しなくては いけません。 省略可能な timeout パラメータを使うことで、 接続するためのシステムコールにおけるタイムアウトを設定することができます。 flags は、接続設定フラグの任意の組み合わせ を指定できるビットフィールドです。 現在、接続設定フラグとして選択できる値は、 **STREAM_CLIENT_CONNECT**（デフォルト）、 **STREAM_CLIENT_ASYNC_CONNECT** と **STREAM_CLIENT_PERSISTENT** のみです。

> **注意**: ソケット上のデータの読み書きに関してタイムアウトを設定する必要がある場合は、 stream_set_timeout() を使ってください。 **stream_socket_client()** に渡される timeout は、ソケットの接続時にのみ適用されます。
>
> **注意**: パラメータ timeout は、非同期通信を試みている場合には適用されません。

stream_socket_client() は、 fgets(), fgetss(), fwrite(), fclose(), feof() といった、ファイル関数と共に 使うことのできるストリームリソースを返します。

もし、呼び出しに失敗すると、**FALSE** を返し、 さらに省略可能な errno 引数と errstr 引数がある場合は、そこに システムレベルの connect() 関数の実行時に 発生したシステムレベルのエラーを表す値を返します。 もし、errno に返された値が 0 で、かつ、 この関数が **FALSE** を返した時は、 connect() システムコールの前に何らかのエラーが 発生したことを示しています。これは、多くの場合、ソケットの初期化 に失敗したことで起こります。errno と errstr パラメータは常に参照渡しされることに 留意してください。

環境により、Unix ドメインや接続タイムアウトが利用できない場合があります。 有効なトランスポートのリストは、stream_get_transports() で取得できます。 組み込むのポートの一覧については、サポートされるソケットトランスポートのリスト を 参照ください。

ストリームはデフォルトではブロックモードで開かれますが、 stream_set_blocking() を使うことで非ブロックモードに 変更することができます。

Example#1 stream_socket_client() の例

```
<?php
$fp = stream_socket_client("tcp://www.example.com:80", $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br />¥n";
} else {
    fwrite($fp, "GET / HTTP/1.0¥r¥nHost: www.example.com¥r¥nAccept: */*¥r¥n¥r¥n");
    while (!feof($fp)) {
        echo fgets($fp, 1024);
    }
    fclose($fp);
}
?>
```

下記の例は、PHP のスクリプトで、どのようにローカル上で動いている UDP サービスの "daytime"（ポート 13）によって日時の情報を 取得できるかを表しています。

Example#2 UDP 接続を使う

```
<?php
$fp = stream_socket_client("udp://127.0.0.1:13", $errno, $errstr);
if (!$fp) {
    echo "ERROR: $errno - $errstr<br />¥n";
} else {
    fwrite($fp, "¥n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

警告

UDP ソケットは、リモートホストに到達できなくても エラーを発生せず、開いているかような状態になることがあります。 このエラーは、実際にソケットに対して、読み込みや書き込み動作を 行ってみないと判断できません。原因としては、UDP が非接続型の プロトコルであることが挙げられます。つまり、実際にデータを 送受信する段階になるまで、OS の側では接続を確立しようとしないという ことです。

> **注意**: 数値で IPv6 アドレスを指定するときは、（例 fe80::1）アドレスを角カッコでくくらなくてはなりません。たとえば、tcp://[fe80::1]:80.

stream_socket_server(), stream_set_blocking(), stream_set_timeout(), stream_select(), fgets(), fgetss(), fwrite(), fclose(), feof(), Curl extension も参照ください。

---

stream_socket_enable_crypto

(PHP 5 >= 5.1.0)

stream_socket_enable_crypto — 接続済みのソケットについて暗号化の on/off を切り替える

説明

mixed **stream_socket_enable_crypto** (resource $stream , bool $enable [, int $crypto_type [, resource $session_stream]])

crypto_type パラメータを指定してコールすると、 **stream_socket_enable_crypto()** は 指定した方法でストリームの暗号化を設定します。

crypto_type に使用可能な値

- **STREAM_CRYPTO_METHOD_SSLv2_CLIENT**
- **STREAM_CRYPTO_METHOD_SSLv3_CLIENT**
- **STREAM_CRYPTO_METHOD_SSLv23_CLIENT**
- **STREAM_CRYPTO_METHOD_TLS_CLIENT**
- **STREAM_CRYPTO_METHOD_SSLv2_SERVER**
- **STREAM_CRYPTO_METHOD_SSLv3_SERVER**
- **STREAM_CRYPTO_METHOD_SSLv23_SERVER**
- **STREAM_CRYPTO_METHOD_TLS_SERVER**

暗号化設定が確立されると、それ以降は enable パラメータに **TRUE** あるいは **FALSE** を指定することで暗号化の on/off を動的に切り替えられます。

既に確立されている暗号化ストリームの設定をもとにする場合は、 ストリームのリソース変数を 4 番目のパラメータに指定します。

成功した場合に **TRUE**、ネゴシエーションに失敗した場合に **FALSE**、 十分なデータがないために再試行が必要な場合（非ブロックモード時のみ） に 0 を返します。

Example#1 stream_socket_enable_crypto() の例

```
<?php
$fp = stream_socket_client("tcp://myproto.example.com:31337", $errno, $errstr, 30);
if (!$fp) {
  die("接続できません: $errstr ($errno)");
}
/* ログイン時の暗号化を有効にします */
stream_socket_enable_crypto($fp, true, STREAM_CRYPTO_METHOD_SSLv23_CLIENT);
fwrite($fp, "USER god¥r¥n");
fwrite($fp, "PASS secret¥r¥n");
/* それ以外では暗号化を無効にします */
stream_socket_enable_crypto($fp, false);
while ($motd = fgets($fp)) {
  echo $motd;
}
fclose($fp);
?>
```

OpenSSL および サポートされるソケットトランスポートのリスト

---

stream_socket_get_name

(PHP 5)

stream_socket_get_name — ローカルまたはリモートのソケットの名前を取得する

説明

string ***stream_socket_get_name*** *(resource $handle , bool $want_peer)*

与えられたソケット接続のローカルまたはリモートでの名前を取得します。 want_peer が ***TRUE*** *にセットされている時は、 リモートでのソケットの名前が、* ***FALSE*** *の時は、ローカルでのソケットの名前が 返されます。*

stream_socket_accept() も参照ください。

---

stream_socket_pair

(PHP 5 >= 5.1.0)

stream_socket_pair — 接続された、区別できないソケットストリームの組を作成する

説明

array ***stream_socket_pair*** *(int $domain , int $type , int $protocol)*

stream_socket_pair() *は、互いに接続されており区別できない ソケットストリームの組を作成します。この関数は、一般に IPC (Inter-Process Communication: プロセス間通信) で使用します。*

パラメータ

domain

使用するプロトコルファミリー。 ***STREAM_PF_INET***、 ***STREAM_PF_INET6*** *あるいは* ***STREAM_PF_UNIX***

type

使用する接続の型。 ***STREAM_SOCK_DGRAM***、 ***STREAM_SOCK_RAW***、 ***STREAM_SOCK_RDM***、 ***STREAM_SOCK_SEQPACKET*** *あるいは* ***STREAM_SOCK_STREAM***

protocol

使用するプロトコル。 ***STREAM_IPPROTO_ICMP***、 ***STREAM_IPPROTO_IP***、 ***STREAM_IPPROTO_RAW***、 ***STREAM_IPPROTO_TCP*** *あるいは* ***STREAM_IPPROTO_UDP***

> ***注意***: *各定数についての詳細な情報は ストリーム定数の一覧 を参照ください。*

返り値

成功した場合に 2 つのソケットリソースの配列、失敗した場合に ***FALSE*** *を返します。*

例

Example#1 stream_socket_pair() の例

この例では、プロセス間通信に ***stream_socket_pair()*** *を使用する基本的な方法を示します。*

```
<?php

$sockets = stream_socket_pair(STREAM_PF_UNIX, STREAM_SOCK_STREAM, STREAM_IPPROTO_IP);
$pid     = pcntl_fork();

if ($pid == -1) {
     die('フォークできません');

} else if ($pid) {
     /* 親プロセス */
    fclose($sockets[0]);

    fwrite($sockets[1], "子プロセスの PID: $pid\n");
     echo fgets($sockets[1]);

    fclose($sockets[1]);

} else {
    /* 子プロセス */
    fclose($sockets[1]);

    fwrite($sockets[0], "子プロセスからのメッセージ\n");
     echo fgets($sockets[0]);

    fclose($sockets[0]);
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
子プロセスの PID: 1378
子プロセスからのメッセージ
```

---

stream_socket_recvfrom

(PHP 5)

stream_socket_recvfrom — 接続されているかどうかにかかわらず、ソケットからのデータを受信する

説明

```
string stream_socket_recvfrom ( resource $socket , int $length [, int $flags [, string &$address ]] )
```

関数 **stream_socket_recvfrom()** は、 リモートソケットから最大 length バイトのデータを 受け取ります。address が指定された場合、 そこにはリモートソケットのアドレスが保存されます。

flags は以下の値の組み合わせです。

flags でとりうる値

STREAM_OOB	OOB (out-of-band) データを処理します。
STREAM_PEEK	ソケットからデータを取得しますが、バッファを消費しません。 fread() あるいは stream_socket_recvfrom() を続けてコールした際には、同じデータが読み込まれます。

Example#1 stream_socket_recvfrom() の例

```
<?php
/* localhost のポート 1234 へのサーバソケットをオープンします */
$server = stream_socket_server('tcp://127.0.0.1:1234');

/* 接続を受け付けます */
$socket = stream_socket_accept($server);

/* OOB データのパケットを取得します (1500 は典型的な MTU のサイズです) */
echo "Received Out-Of-Band: '" . stream_socket_recvfrom($socket, 1500, STREAM_OOB) . "'¥n";

/* 通常の帯域内のデータを読み込みますが、バッファを消費しません */
echo "Data: '" . stream_socket_recvfrom($socket, 1500, STREAM_PEEK) . "'¥n";

/* 同じパケットをもう一度読み込みます。今度はバッファからそれを削除します */
echo "Data: '" . stream_socket_recvfrom($socket, 1500) . "'¥n";

/* ソケットを閉じます */
fclose($socket);
fclose($server);
?>
```

> **注意**: 受信したメッセージが length パラメータより 長かった場合、ソケットの型によっては (例えば UDP など) 超過分の バイトデータが捨てられてしまう可能性があります。
>
> **注意**: バッファベースのストリーム関数 (fread() あるいは stream_get_line() など) をコールした後に ソケットベースのストリームで **stream_socket_recvfrom()** をコールすると、ストリームバッファを経由せず、データをソケットから 直接読み込みます。

stream_socket_sendto()、 stream_socket_client() および stream_socket_server() も参照ください。

---

stream_socket_sendto

(PHP 5)

stream_socket_sendto — 接続されているかどうかにかかわらず、ソケットにデータを送信する

説明

```
int stream_socket_sendto ( resource $socket , string $data [, int $flags [, string $address ]] )
```

関数 **stream_socket_sendto()** は、 data で指定したデータを socket で指定したソケットに送信します。 address で別のアドレスが指定されていない限り、 ソケットストリームが作成された際のアドレスを使用します。

flags は以下の値の組み合わせです。

flags でとりうる値

STREAM_OOB	OOB (out-of-band) データを処理します。

Example#1 stream_socket_sendto() の例

```
<?php
/* localhost のポート 1234 へのソケットをオープンします */
$socket = stream_socket_client('tcp://127.0.0.1:1234');

/* 普通のデータを普通のチャネルで送信します */
fwrite($socket, "Normal data transmit.");

/* 帯域外のデータを送信します */
stream_socket_sendto($socket, "Out of Band data.", STREAM_OOB);

/* ソケットを閉じます */
fclose($socket);
?>
```

stream_socket_recvfrom()、 stream_socket_client() および stream_socket_server() も参照ください。

---

stream_socket_server

(PHP 5)

stream_socket_server — インターネットドメインまたは Unix ドメインのサーバソケットを作成する

説明

resource **stream_socket_server** (string $local_socket [, int &$errno [, string &$errstr [, int $flags [, resource $context]]]])

local_socket で指定された接続ポイントに、 ストリームまたはデータグラムソケットによる接続を作成します。 作成されるソケットのタイプは、 [トランスポート]://[ターゲット] という形式の URL フォーマットによって指定された トランスポートによって決定されます: TCP や UDP といったインターネットドメインのソケット (AF_INET) には、remote_socket パラメータの ターゲット の部分は、ホスト名または IP アドレスと、 それに続くコロンで区切られたポート番号から構成されていなければなりません。 Unix ドメインのソケットの場合は、ターゲット の部分は、ファイルシステムにおけるソケットのファイルを指定しなくては いけません。 flags は、ソケット作成フラグの任意の組み合わせ を指定できるビットフィールドです。 デフォルトの値は、**STREAM_SERVER_BIND** | **STREAM_SERVER_LISTEN** です。

> **注意**: UDP ソケットに対しては、**STREAM_SERVER_BIND** を flags パラメータとして使用する必要があります。

この関数は、ソケットのみを作成します。接続待ちの状態に入るには、stream_socket_accept() 関数を使います。

もし失敗した場合は、**FALSE** を返し、その時オプションの errno と errstr パラメータが存在するときは、そこにシステムレベルの socket()、bind() および listen() のコールにおいて発生した 実際のシステムレベルのエラーを返します。 もし、errno に返された値が 0 で、かつ **FALSE** が返された場合、bind() コールを行う前にエラーが発生したことを示しており、これは多くの場合 ソケットの初期化に失敗したことを示しています。 errno と errstr パラメータは常に参照渡しとなることに留意してください。

システムの種類によって、Unix ドメインのソケットが利用できない場合が あります。 利用できるトランスポートの種類は、stream_get_transports() によって知ることができます。 組み込みのトランスポートのリストは、サポートされるソケットトランスポートのリストを参照ください。

Example#1 TCP サーバソケットの使用

```
<?php
$socket = stream_socket_server("tcp://0.0.0.0:8000", $errno, $errstr);
if (!$socket) {
   echo "$errstr ($errno)<br />¥n";
} else {
   while ($conn = stream_socket_accept($socket)) {
     fwrite($conn, 'The local time is ' . date('n/j/Y g:i a') . "¥n");
     fclose($conn);
   }
   fclose($socket);
}
?>
```

下記の例は、PHP のスクリプトが、どうやって stream_socket_client() で示したような、 問い合わせに応答するタイムサーバとして機能するかを示したものです。

> **注意**: 1024 番よりも小さいポート番号のサーバソケットを作成する場合、 多くのシステムでは root 権限が必要となります。

Example#2 UDP サーバソケットを利用する

```
<?php
$socket = stream_socket_server("udp://127.0.0.1:1113", $errno, $errstr, STREAM_SERVER_BIND);
if (!$socket) {
    die("$errstr ($errno)");
}

do {
    $pkt = stream_socket_recvfrom($socket, 1, 0, $peer);
    echo "$peer¥n";
    stream_socket_sendto($socket, date("D M j H:i:s Y¥r¥n"), 0, $peer);
} while ($pkt !== false);

?>
```

> **注意**: 数値で IPv6 アドレスを指定するときは、 (例 fe80::1) アドレスを角カッコでくくらなくてはなりません。たとえば、tcp://[fe80::1]:80.

stream_socket_client(), stream_set_blocking(), stream_set_timeout(), fgets(), fgetss(), fwrite(), fclose(), feof(), と Curl 拡張モジュール も参照ください。

---

stream_socket_shutdown

(PHP 5 >= 5.2.1)

stream_socket_shutdown — 全二重接続を終了する

説明

bool **stream_socket_shutdown** (resource $stream , int $how)

全二重接続を (一時的あるいはそうでなく) 終了します。

パラメータ

stream

オープンしているストリーム (たとえば stream_socket_client() でオープンしたものなど)。

how

以下の定数のいずれか。**STREAM_SHUT_RD** (それ以降の受信を無効にする)、**STREAM_SHUT_WR** (それ以降の送信を無効にする) あるいは **STREAM_SHUT_RDWR** (それ以降の送受信を無効にする)。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 stream_socket_shutdown() の例

```
<?php

$server = stream_socket_server('tcp://127.0.0.1:1337');
$client = stream_socket_client('tcp://127.0.0.1:1337');

var_dump(fputs($client, "hello"));

stream_socket_shutdown($client, STREAM_SHUT_WR);
var_dump(fputs($client, "hello")); //  ここでは動作しません

?>
```

上の例の出力は、たとえば以下のようになります。

```
int(5)

Notice: fputs(): send of 5 bytes failed with errno=32 Broken pipe in test.php on line 9
int(0)
```

参考

- fclose()

---

stream_wrapper_register

(PHP 4 >= 4.3.2, PHP 5)

stream_wrapper_register — PHP のクラスとして実装された URL ラッパを登録する

説明

bool **stream_wrapper_register** (string $protocol , string $classname)

stream_wrapper_register() は、自分で作った プロトコルハンドラとストリームを実装し、それを fopen() や fread() といったファイルシステムの関数と利用することを可能にします。

ラッパを実装するには、下記のようないくつかのメンバ関数を持った クラスを定義しなくてはなりません。もし、誰かがあなたの作った ストリームを fopen() したとき、 PHP は classname のインスタンスを作り 以後そのインスタンスと共にメソッドを呼び出そうとします。 これらのメソッドは、下記に示したとおり、正確に実装されなければなりません。 さもないと、定義されていない動作をします。

PHP 5.0.0 より新たに、classname のインスタンス内の context プロパティが、 コンテキストリソース を参照する形で 初期化されるようになります。なお、 この コンテキストリソースは、 stream_context_get_options() によっても 取得することができます。 もし、コンテキストがストリーム生成関数に対し渡されていない場合は、 context の値は **NULL** にセットされます。

stream_wrapper_register() は、 protocol というハンドラが既にある場合、 **FALSE** を返します。

bool **stream_open** (string $path , string $mode , int $options , string $opened_path)

このメソッドは、ストリームオブジェクトが生成された直後に呼び出されます。 path には、fopen() に 与えられ、データの取得元となる URL が入ります。 parse_url() を使えば、この URL を解釈できます。

mode は、ファイルを開く際に使われるモードです。 fopen() で詳しく述べられているように、 この関数の側で、mode が要求された path に適切かどうかを調べなくてはなりません。

options は、ストリーム API によってセットされる 付加的なフラグです。この値は次に挙げた値のいずれかか、 それらを二つ以上 OR 演算した値となります。

フラグ	説明
STREAM_USE_PATH	もし path が相対パスだった場合、 include_path で指定されたパス内を探索します。
STREAM_REPORT_ERRORS	このフラグがセットされている場合は、ストリームのオープン時に ラッパ側で trigger_error() 関数を使い、エラーを発生させる必要があります。逆に、このフラグがセットされていない場合は、 ラッパ側でエラーを発生させてはいけません。

もし path に対するストリームのオープンに成功し、 STREAM_USE_PATH が options にセットされていた 際には、opened_path に、実際に開かれた ファイルまたはリソースへの絶対パスをセットしなくてはなりません。

もし要求されたリソースを正常に開けた場合、**TRUE** を、そうでなければ、 **FALSE** を返さなくてはなりません。

void **stream_close** (void)

このメソッドは、fclose()によってストリームが閉じられるときに呼び出されます。 この時点で、カスタムストリームの中でロックされたか確保されたリソースを 開放しなくてはなりません。

string **stream_read** (int $count)

このメソッドはカスタムストリームに対する fread() または fgets() に対応して呼び出されます。 count バイトを超えない長さのデータが返される ようにする必要があります。 count バイトよりも少ないデータしか準備できていない 場合は、その時点にあるだけを返します。もし渡すべきデータがもうなければ、 **FALSE** または空文字列を返してください。 また、正常に読み込めた分だけ、ストリームの読み込み/書き出し操作における 内部位置を更新しなくてはなりません。

int **stream_write** (string $data)

このメソッドは、カスタムストリームに対する fwrite() などの 書き出し動作に対応して呼び出されます。data には、 カスタムストリームで使われている下層にあるストレージに書き出されるべき データが入っています。渡されたデータをすべて書き出せない場合は、 書き出せるだけ書き出しま

す。このとき、正常にデータを書き出せた場合は、 その書き出せたバイト数を、書き出せなかった場合は 0 を返さなくては なりません。 また、正常に読み込めた分だけ、ストリームの読み込み/書き出し操作における 内部位置を更新しなくてはなりません。

bool **stream_eof** (void)

このメソッドは、ストリームに対する feof() 関数の実行に 対応して呼び出されます。 もし、ストリームの読み込み/書き出し操作における内部位置が、 ストリームの終端にある場合は、あるいは、もう読み込むべきデータがない 場合は **TRUE** を、その他の場合には **FALSE** をそれぞれ返さなくては なりません。

int **stream_tell** (void)

このメソッドは、ストリームに対する ftell() 関数の 実行に対応して呼び出されます。 ストリームの読み込み/書き出し操作における内部位置を返します。

bool **stream_seek** (int $offset , int $whence)

このメソッドは、ストリームに対する fseek() 関数の 実行に対応して呼び出されます。 ストリームの読み込み/書き出し操作における内部位置を、offset パラメータと whence パラメータの値に従って更新しなければなりません。 なお、これらの値の取り方については、fseek() を参照ください。 正常に位置が更新できた場合には **TRUE** を、失敗した場合は **FALSE** を返します。

bool **stream_flush** (void)

このメソッドは、ストリームに対する fflush() 関数の 実行に対応して呼び出されます。 もしカスタムストリームがデータをキャッシュしていて、それをまだ下層にあるストレージへ記録していない場合は、 このメソッドが呼ばれた時に、それを書き出します。 もしキャッシュされたデータが正常に書き出された（あるいは、もう書くべきデータ がない）場合は、**TRUE** を、もしくは、データを書き出すことができなかった 場合は **FALSE** を返します。

array **stream_stat** (void)

このメソッドは、ストリームに対する fstat() 関数の 実行に対応して呼び出されます。 fstat() が返すのと同じような要素をもつ配列に、 適切な値を入れて返さなくてはなりません。

bool **unlink** (string $path)

このメソッドは、ラッパに関連付けられた URL のパスに対する unlink() 関数の実行に対応して呼び出されます。 path で指定されたパスにあるアイテムを 削除しようと試みなければなりません。その時、削除に成功したら **TRUE** を、失敗したら **FALSE** を返してください。 適切なエラーメッセージがユーザに返されるよう、ラッパ側で削除を サポートしない場合は、このメソッドを定義しないでください。

注意: ユーザ定義のラッパにおける unlink メソッドは、 PHP 5.0.0 以前ではサポートされていません。

bool **rename** (string $path_from , string $path_to)

このメソッドは、ラッパに関連付けられた URL のパスに対する rename() 関数の実行に対応して呼び出され、 path_from で指定されたパスにある項目を path_to にリネームすることを試みなければなりません。 成功した場合に **TRUE**、失敗した場合に **FALSE** を返さなければなりません。 適切なエラーメッセージが返されるよう、ラッパ側でリネームをサポートしない 場合はこのメソッドを定義しないでください。

注意: ユーザ定義のラッパにおける rename メソッドは、 PHP 5.0.0 以前ではサポートされていません。

bool **mkdir** (string $path , int $mode , int $options)

このメソッドは、ラッパに関連付けられた URL のパスに対する mkdir() 関数の実行に対応して呼び出され、 path で指定されたディレクトリの作成を 試みなければなりません。 成功した場合に **TRUE**、失敗した場合に **FALSE** を返さなければなりません。 適切なエラーメッセージが返されるよう、ラッパ側でディレクトリの作成を サポートしない場合はこのメソッドを定義しないでください。 options でとりうる値には **STREAM_REPORT_ERRORS** および **STREAM_MKDIR_RECURSIVE** が含まれます。

注意: ユーザ定義のラッパにおける mkdir メソッドは、 PHP 5.0.0 以前ではサポートされていません。

bool **rmdir** (string $path , int $options)

このメソッドは、ラッパに関連付けられた URL のパスに対する rmdir() 関数の実行に対応して呼び出され、 path で指定されたディレクトリの削除を 試みなければなりません。 成功した場合に **TRUE**、失敗した場合に **FALSE** を返さなければなりません。 適切なエラーメッセージが返されるよう、ラッパ側でディレクトリの削除を サポートしない場合はこのメソッドを定義しないでください。 options でとりうる値には **STREAM_REPORT_ERRORS** が含まれます。

注意: ユーザ定義のラッパにおける rmdir メソッドは、 PHP 5.0.0 以前ではサポートされていません。

bool **dir_opendir** (string $path , int $options)

このメソッドは、ディレクトリの内容を走査するために、 ストリームオブジェクトが opendir()によって 生成されるその時に呼び出されます。path は、opendir() に 渡された、このオブジェクトが走査すべき場所を示す URL を指定します。 このとき、parse_url() を使えば、この URL を 細分することができます。

array **url_stat** (string $path , int $flags)

このメソッドは、ラッパに関連付けられた URL のパスに対する stat() 関数の実行に対応して呼び出され、 システムの関数が返す要素とできるだけ同じ内容を返さなければなりません。 不明な、あるいは取得できない値には、妥当な値を設定しておくべきです （通常は **0** です）。

flags には、ストリーム API によって設定された 追加のフラグを保持します。以下のうちのひとつまたは複数を OR で結合した 値を保持します。

フラグ	説明
STREAM_URL_STAT_LINK	他のリソースへのリンク機能があるリソース（たとえば HTTP の Location: forward やファイルシステムのシンボリック リンクなど）。このフラグはリンクそのものについての情報のみを 返し、リンクが指している先のリソースについては返しません。 このフラグは lstat()、 is_link() あるいは filetype() のコールへの応答として設定されます。
STREAM_URL_STAT_QUIET	このフラグが設定されている場合、ラッパは一切エラーを 報告してはいけません。このフラグが設定されていない場合は、 パスを調べている間に発生したエラーについて trigger_error() で報告する義務があります。

string **dir_readdir** (void)

このメソッドは、readdir() 関数の呼び出しに 対応して呼ばれます。**dir_opendir()** メソッドにおいて 指定された場所にあるファイルのリスト上の、次のファイル名を表す 文字列を返さなくてはなりません。

bool **dir_rewinddir** (void)

このメソッドは、rewinddir() 関数の呼び出しに 対応して呼ばれます。このとき、**dir_readdir()** メソッドにおいて変更された状態をリセットしなくてはなりません。 つまり、このメソッドの呼び出しに続いて **dir_readdir()** が 呼ばれると、**dir_opendir()** によって開かれた ファイルのリスト上の、最初のエントリを返すことになります。

bool **dir_closedir** (void)

このメソッドは、closedir() 関数の呼び出しに 対応して呼ばれます。このとき、ディレクトリストリームの利用に際して 確保されたりロックされたリソースを全て開放しなければなりません。

下記の例は、var:// プロトコルハンドラを実装し、 fread() などの標準のファイルシステム関数 を利用して、指定されたグローバル変数に対するアクセスができるように するものです。 var:// プロトコルは、"var://foo" として与えられた URL に対応して、 $GLOBALS["foo"] からデータを読み込んだり、そこにデータを書き込んだり します。

Example#1 グローバル変数を読み書きするストリーム

```
<?php

class VariableStream {
    var $position;
    var $varname;

    function stream_open($path, $mode, $options, &$opened_path)
    {
        $url = parse_url($path);
        $this->varname = $url["host"];
        $this->position = 0;

        return true;
    }

    function stream_read($count)
    {
        $ret = substr($GLOBALS[$this->varname], $this->position, $count);
        $this->position += strlen($ret);
        return $ret;
    }

    function stream_write($data)
    {
        $left = substr($GLOBALS[$this->varname], 0, $this->position);
        $right = substr($GLOBALS[$this->varname], $this->position + strlen($data));
        $GLOBALS[$this->varname] = $left . $data . $right;
        $this->position += strlen($data);
        return strlen($data);
    }

    function stream_tell()
    {
        return $this->position;
    }

    function stream_eof()
    {
        return $this->position >= strlen($GLOBALS[$this->varname]);
    }

    function stream_seek($offset, $whence)
    {
        switch ($whence) {
            case SEEK_SET:
                if ($offset < strlen($GLOBALS[$this->varname]) && $offset >= 0) {
                    $this->position = $offset;
                    return true;
                } else {
                    return false;
                }
                break;

            case SEEK_CUR:
                if ($offset >= 0) {
                    $this->position += $offset;
                    return true;
                } else {
                    return false;
                }
                break;

            case SEEK_END:
                if (strlen($GLOBALS[$this->varname]) + $offset >= 0) {
                    $this->position = strlen($GLOBALS[$this->varname]) + $offset;
                    return true;
                } else {
                    return false;
                }
                break;

            default:
                return false;
        }
    }
}

stream_wrapper_register("var", "VariableStream")
    or die("プロトコルの登録に失敗しました");

$myvar = "";

$fp = fopen("var://myvar", "r+");

fwrite($fp, "line1¥n");
fwrite($fp, "line2¥n");
fwrite($fp, "line3¥n");

rewind($fp);
while (!feof($fp)) {
```

```
    echo fgets($fp);
}
fclose($fp);
var_dump($myvar);
?>
```

---

stream_wrapper_restore

(PHP 5 >= 5.1.0)

stream_wrapper_restore — 事前に登録を解除された組み込みラッパを復元する

説明

bool **stream_wrapper_restore** (string $protocol)

stream_wrapper_unregister() で事前に登録を解除した 組み込みラッパを復元します。

---

stream_wrapper_unregister

(PHP 5 >= 5.1.0)

stream_wrapper_unregister — URL ラッパの登録を解除する

説明

bool **stream_wrapper_unregister** (string $protocol)

stream_wrapper_unregister() は、すでに定義 されているストリームラッパを無効にします。ラッパが無効になった後は、stream_wrapper_register() を使用してユーザ定義の ラッパで上書きしたり stream_wrapper_restore() で再度使用可能にしたりすることが可能となります。

目次

- stream_socket_server — インターネットドメインまたは Unix ドメインのサーバソケットを作成する
- stream_socket_shutdown — 全二重接続を終了する
- stream_wrapper_register — PHP のクラスとして実装された URL ラッパを登録する
- stream_wrapper_restore — 事前に登録を解除された組み込みラッパを復元する
- stream_wrapper_unregister — URL ラッパの登録を解除する

Strings(文字列関数)

導入

以下の関数はすべて、文字列をいろいろな方法で操作します。正規表現や URL 処理の節にも関連する記述があります。

文字列の動作に関する情報、特にシングルクオート、ダブルクオート、 エスケープシーケンスについては、マニュアルの 型 の節にある 文字列 エントリを 参照ください。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

CRYPT_SALT_LENGTH integer
CRYPT_STD_DES integer
CRYPT_EXT_DES integer
CRYPT_MD5 integer
CRYPT_BLOWFISH integer
HTML_SPECIALCHARS (integer)
HTML_ENTITIES (integer)
ENT_COMPAT (integer)
ENT_QUOTES (integer)
ENT_NOQUOTES (integer)
CHAR_MAX (integer)
LC_CTYPE (integer)
LC_NUMERIC (integer)
LC_TIME (integer)
LC_COLLATE (integer)
LC_MONETARY (integer)
LC_ALL (integer)
LC_MESSAGES (integer)
STR_PAD_LEFT (integer)
STR_PAD_RIGHT (integer)
STR_PAD_BOTH (integer)

参考

より強力な文字列処理および処理関数については、 POSIX 正規表現関数および Perl 互換正規表現関数 を参照ください。

---

addcslashes

(PHP 4, PHP 5)

addcslashes — C 言語と同様にスラッシュで文字列をクォートする

説明

string **addcslashes** (string $str , string $charlist)

charlist パラメータに羅列された文字の前にバックスラッシュを付けた文字列を返します。

パラメータ

str

エスケープしたい文字列。

charlist

エスケープの対象となる文字を並べたもの。 charlist が ¥n, ¥r 等の文字を含んでいる場合、 C言語と同様の手法によりエスケープされます。 アスキーコードが32未満または126より大きい文字は、8進表現に変換されます。

charlist 引数の文字の列びを定義する際には、 範囲の最初と最後で指定する文字集合に含まれる文字の種類を把握するようにしてください。

```
<?php
echo addcslashes('foo[ ]', 'A..z');
// 出力:  \f\o\o\[ \]
// 全ての大文字と小文字はエスケープされます。
// ... しかし、[\]^_` とタブ、改行、復帰文字等もエスケープされてしまいます。
```

```
?>
```
また、ある範囲を指定する最初の文字がその範囲の2番目の文字よりも大きな ASCII 値を有している場合、範囲は定義されません。 最初と最後の文字とピリオド文字のみがエスケープされます。 ある文字の ASCII 値を見つけるには、 ord() 関数を使用してください。

```
<?php
echo addcslashes("zoo['.']", 'z..A');
// 出力:   \zoo['\.']
?>
```

エスケープ文字を0, a, b, f, n, r, t , vとする場合には注意してください。 これらは、\0, \a, \b, \f, \n, \r, \t , \v に変換されます。PHPでは、 \0 (NULL), \r (復改文字), \n (改行文字) , \t (タブ) が定義済みのエスケープシーケンスですが、C言語では、 これら全てが定義済みのエスケープシーケンスです。

返り値

エスケープされた文字列を返します。

例

"\0..\37" のように charlist に範囲を指定可能です。この場合、アスキーコードが 0 から 31 の範囲の文字は全てエスケープされます。

Example#1 addcslashes() の例

```
<?php
$escaped = addcslashes($not_escaped, "¥0..¥37!@¥177..¥377");
?>
```

参考

- stripcslashes()
- stripslashes()
- addslashes()
- htmlspecialchars()
- quotemeta()

---

addslashes

(PHP 4, PHP 5)

addslashes — 文字列をスラッシュでクォートする

説明

```
string addslashes ( string $str )
```

データベースへの問い合わせなどに際してクォートされるべき文字の前に バックスラッシュを挿入した文字列を返します。 クォートされるべき文字とは、シングルクォート('), ダブルクォート("),バックスラッシュ (¥) ,NUL (**NULL** バイト) です。

addslashes() の使用例は、 データベースにデータを登録するときです。例えば、 O'reilly という名前をデータベースに挿入するには、 エスケープする必要があります。ほとんどのデータベースでは O¥'reilly という具合に ¥ を使用します。これはデータベースにデータを格納する場合のみ必要です。 追加される ¥ は挿入されません。 PHP のディレクティブ magic_quotes_sybase を ¥ にすると、' はもうひとつの ' でエスケープされます。

PHP ディレクティブ magic_quotes_gpc はデフォルトでは on で、 全ての GET、POST、COOKIE データについて基本的に **addslashes()** を実行します。 magic_quotes_gpc によってすでにエスケープされた文字列に対して **addslashes()** を実行しないでください。 さもないと、重複してエスケープされてしまいます。 関数 get_magic_quotes_gpc() はこれを確認するのに役立つかも知れません。

パラメータ

str

エスケープしたい文字列。

返り値

エスケープされた文字列を返します。

例

Example#1 addslashes() の例

```
<?php
$str = "Is your name O'reilly?";

// 出力: Is your name O\'reilly?
echo addslashes($str);
?>
```

参考

- stripcslashes()
- stripslashes()
- addcslashes()
- htmlspecialchars()
- quotemeta()
- get_magic_quotes_gpc()

bin2hex

(PHP 4, PHP 5)

bin2hex — バイナリデータを16進表現に変換する

説明

string **bin2hex** (string $str)

str を16進表現に変換したASCII文字列を返します。 変換は、上位ニブルからバイト毎に行われます。

パラメータ

str

文字。

返り値

指定した文字列を16進表現に変換したものを返します。

参考

- pack()
- unpack()

---

chop

(PHP 4, PHP 5)

chop — rtrim() のエイリアス

説明

この関数は次の関数のエイリアスです。 rtrim().

注意

> **注意**: **chop()** は、文字列の末尾の文字を削除する Perl の chop()関数とは異なります。

---

chr

(PHP 4, PHP 5)

chr — 特定の文字を返す

説明

string **chr** (int $ascii)

ascii で指定された、1文字からなる文字列を返します。

この関数はord()の逆の動作をします。

パラメータ

ascii

ascii コード。

返り値

指定した文字を返します。

例

Example#1 chr() の例

```
<?php
$str = "この文字列はエスケープで終了します: ";
$str .= chr(27); /* $str の最後にエスケープ文字を付加する */

/* こちらの方がより便利なことが多い */

$str = sprintf("この文字列はエスケープで終了します: %c", 27);
?>
```

参考

- sprintf() のフォーマット文字列 %c
- » ASCII テーブル

---

chunk_split

(PHP 4, PHP 5)

chunk_split — 文字列をより小さな部分に分割する

説明

string **chunk_split** (string $body [, int $chunklen [, string $end]])

文字列をより小さな部分に分割する際に使用され、 base64_encode() の出力を RFC 2045の規約に基づいた出力に変換するといった用途に適しています。 この関数は、chunklen 文字毎に文字列 end を挿入します。

パラメータ

body

分割したい文字列。

chunklen

各部分の長さ。デフォルトは 76 です。

end

行末の区切り。デフォルトは "\r\n" です。

返り値

分割した文字列を返します。

例

Example#1 chunk_split() の例

```
<?php
// RFC 2045 に基づき $data をフォーマットします
$new_string = chunk_split(base64_encode($data));
?>
```

参考

- str_split()
- explode()
- split()
- wordwrap()
- » RFC 2045

---

convert_cyr_string

(PHP 4, PHP 5)

convert_cyr_string — キリル文字セットを他のものに変換する

説明

string **convert_cyr_string** (string $str , string $from , string $to)

キリル文字セットを、別の文字セットに変換します。

パラメータ

str

変換したい文字列。

from

変換元のキリル文字セットを一文字で表したもの。

to

変換先のキリル文字セットを一文字で表したもの。

サポートされている文字セットは次のとおりです。

- k - koi8-r
- w - windows-1251
- i - iso8859-5

- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

返り値

変換した文字列を返します。

注意

注意: この関数はバイナリデータに対応しています。

---

convert_uudecode

(PHP 5)

convert_uudecode — uuencode された文字列をデコードする

説明

string **convert_uudecode** (string $data)

convert_uudecode() は、uuencode された文字列をデコードします。

パラメータ

data

uuencode されたデータ。

返り値

デコードしたデータを文字列で返します。

例

Example#1 convert_uudecode() の例

```
<?php
/* 何が表示されるか、想像できますか? (^o^) */
echo convert_uudecode("+22!L;W9E(%!(4¥"$`¥n`");
?>
```

参考

- convert_uuencode()

---

convert_uuencode

(PHP 5)

convert_uuencode — 文字列を uuencode する

説明

string **convert_uuencode** (string $data)

convert_uuencode() は、uuencode アルゴリズムを使用して文字列をエンコードします。

uuencode はすべての文字(バイナリを含む)を表示可能な文字に変換し、 ネットワーク上での転送を可能にします。uuencode されたデータは、 元のデータより約 35% 大きくなります。

パラメータ

data

エンコードしたいデータ。

返り値

uuencode されたデータを返します。

例

Example#1 convert_uuencode() の例

```
<?php
$some_string = "test¥ntext  text¥r¥n";

echo convert_uuencode($some_string);
?>
```

参考

- convert_uudecode()
- base64_encode()

---

count_chars

(PHP 4, PHP 5)

count_chars — 文字列で使用されている文字に関する情報を返す

説明

mixed **count_chars** (string $string [, int $mode])

string において各バイト値 (0..255) が存在する数をかぞえ、様々な手法で返します。

パラメータ

string

調べたい文字列。

mode

オプションのパラメータ mode のデフォルトは 0 です。

返り値

mode の値により、 **count_chars()** は以下の値のどれかを返します。

- 0 - 各バイト値をキー、各バイトの出現回数を値とする配列。
- 1 - 0と同じですが、各バイト値の出現回数がゼロより大きいものの一覧となります。
- 2 - 0と同じですが、各バイト値の出現回数がゼロであるものの一覧となります。
- 3 - 使用されている全てのバイト値を有する文字列が返されます。
- 4 - 使用されていない全てのバイト値を有する文字列が返されます。

例

Example#1 count_chars() の例

```
<?php
$data = "Two Ts and one F.";

foreach (count_chars($data, 1) as $i => $val) {
    echo "There were $val instance(s) of ¥"" , chr($i) , "¥" in the string.¥n";
}
?>
```

上の例の出力は以下となります。

```
There were 4 instance(s) of " " in the string.
There were 1 instance(s) of "." in the string.
There were 1 instance(s) of "F" in the string.
There were 2 instance(s) of "T" in the string.
There were 1 instance(s) of "a" in the string.
There were 1 instance(s) of "d" in the string.
There were 1 instance(s) of "e" in the string.
There were 2 instance(s) of "n" in the string.
There were 2 instance(s) of "o" in the string.
There were 1 instance(s) of "s" in the string.
There were 1 instance(s) of "w" in the string.
```

参考

- strpos()
- substr_count()

---

crc32

(PHP 4 >= 4.0.1, PHP 5)

crc32 — 文字列の crc32 多項式計算を行う

説明

int **crc32** (string $str)

str の 32 ビット長の CRC (cyclic redundancy checksum) チェックサムを生成します。 これは通常、送信したデータの整合性を検証するために使用します。

PHP の整数型は符号付きで、多くの crc32 チェックサムは負の整数になるため、 符号なしの crc32 チェックサムの文字列表記を取得するには sprintf() もしくは printf() の "%u" フォーマッタを使用する必要があります。

パラメータ

str

データ。

返り値

例

Example#1 crc32 チェックサムの表示

この例は printf() 関数を用いた変換後のチェックサムの表示方法を示しています。

```
<?php
$checksum = crc32("The quick brown fox jumped over the lazy dog.");
printf("%u¥n", $checksum);
?>
```

参考

- md5()
- sha1()

---

crypt

(PHP 4, PHP 5)

crypt — 文字列の一方向の暗号化（ハッシュ化）を行う

説明

string **crypt** (string $str [, string $salt])

crypt() は暗号化した文字列を返します。 Unix 標準の DES ベースの暗号化方式か、 あるいはそのシステム上で使用できるその他のアルゴリズムを使用します。

複数の暗号化方式をサポートしているオペレーティングシステムもあります。 実際、標準の DES ベースの暗号化方式を MD5 ベースのものに置き換えることもあります。 暗号化方式は、salt 引数によって決まります。 PHP のインストール時に、そのシステムで使用できる暗号化関数を判別します。 salt を省略した場合は、デフォルトで 2 文字の salt を自動生成します。 ただし、そのシステムのデフォルトの暗号化方式が MD5 の場合は、MD5 互換のランダムな salt を生成します。 PHP の定数 **CRYPT_SALT_LENGTH** を確認すると、そのシステムで使用できる salt が通常の 2 文字のものなのか 12 文字のものが使用できるのかがわかります。

標準の DES ベースの暗号化の場合、**crypt()** は出力の最初の 2 文字を salt として使用します。また、 str の最初の 8 文字しか使用しません。つまり、最初の 8 文字が同じである長い文字列は、 同じ salt を使う限り同じ結果となります。

crypt() が複数の暗号化方式をサポートしているシステムでは、 その方式が使用可能かどうかによって次の定数群が 0 か 1 に設定されます。

- **CRYPT_STD_DES** - 標準の DES ベースの暗号化方式で、2 文字の salt を使用するもの
- **CRYPT_EXT_DES** - 拡張した DES ベースの暗号化方式で、9 文字の salt を使用するもの
- **CRYPT_MD5** - 1 ではじまる 12 文字の salt を使用する MD5 暗号化方式
- **CRYPT_BLOWFISH** - 2 あるいは $2a$ ではじまる 16 文字の salt を使用する Blowfish 暗号化方式

パラメータ

str

暗号化したい文字列。

salt

暗号化のもととなる salt 文字列。省略した場合は、 この関数がコールされるたびに PHP がランダムな値を生成します。

自動生成される salt を使用する場合は、 salt が生成されるのが一度だけであることに注意しましょう。 この関数を繰り返しコールすると、 出力だけでなくセキュリティにも影響を与えます。

返り値

暗号化した文字列を返します。

例

Example#1 crypt() の例

```
<?php
$password = crypt('mypassword'); //  saltを自動的に生成させます

/* 異なったハッシュアルゴリズムが使用された際の問題を避けるために
    crypt()の結果全体をパスワード比較用のsaltとして渡す必要があります。
    (上記のように標準DESに基づくパスワードハッシュは2文字のsaltを使用します
    が、MD5に基づくハッシュは12文字のsaltを使用します) */
if (crypt($user_input, $password) == $password) {
    echo "Password  verified!";
}
```

```
?>
```

Example#2 crypt() を htpasswd で使用する例

```
<?php
// パスワードを設定します
$password = 'mypassword';

// ハッシュを取得します。salt は自動生成させます
$hash = crypt($password);
?>
```

Example#3 異なる暗号化手法を用いた crypt() の例

```
<?php
if (CRYPT_STD_DES == 1) {
    echo 'Standard DES: ' . crypt('rasmuslerdorf', 'rl') . "¥n";
}

if (CRYPT_EXT_DES == 1) {
    echo 'Extended DES: ' . crypt('rasmuslerdorf', '_J9..rasm') . "¥n";
}

if (CRYPT_MD5 == 1) {
    echo 'MD5:          ' . crypt('rasmuslerdorf', '$1$rasmusle$') . "¥n";
}

if (CRYPT_BLOWFISH == 1) {
    echo 'Blowfish:     ' . crypt('rasmuslerdorf', '$2a$07$rasmuslerd..........$') . "¥n";
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Standard DES: rl.3StKT.4T8M
Extended DES: _J9..rasmBYk8r9AiWNc
MD5:          $1$rasmusle$rISCgZzpwk3UhDidwXvin0
Blowfish:     $2a$07$rasmuslerd............nIdrcHdxcUxWomQX9j6kvERCFjTg7Ra
```

注意

注意: 復号するための関数はありません。 **crypt()** が使用しているのは単方向アルゴリズムだからです。

参考

- md5()
- mcrypt 拡張モジュール
- 暗号化関数についての Unix man ページ

---

echo

(PHP 4, PHP 5)

echo — 1 つ以上の文字列を出力する

説明

```
void echo ( string $arg1 [, string $... ] )
```

すべてのパラメータを出力します。

echo() は実際には関数ではありません（言語構造です）。このため、使用する際に括弧は必要ありません。（いくつかの他の言語構造と異なり）**echo()** は関数のように動作しません。そのため、 関数のコンテキスト中では常に使用することができません。 加えて、複数のパラメータを指定して **echo()** をコールしたい場合、括弧の中にパラメータを記述してはいけません。

echo() には、開始タグの直後に等号を付ける短縮構文もあります。 この短縮構文は、設定オプションshort_open_tag が有効な場合のみ使用可能です。

```
I have <?=$foo?> foo.
```

パラメータ

arg1

出力したいパラメータ。

...

返り値

値を返しません。

例

Example#1 echo() の例

```
<?php
```

```
echo "Hello World";

echo "This spans
multiple lines. The newlines will be
output as well";

echo "This spans¥nmultiple lines. The newlines will be¥noutput as well.";

echo "Escaping characters is done ¥"Like this¥".";

// echo 命令の中で変数を使用することが可能です
$foo = "foobar";
$bar = "barbaz";

echo "foo is $foo"; // foo is foobar

// 配列を使用することもできます
$baz = array("value" => "foo");

echo "this is {$baz['value']} !"; // this is foo !

// 値ではなく変数名を出力するシングルクオートを使用します
echo 'foo is $foo'; // foo is $foo

// 他の文字を全く使用しない場合、echo 変数を使用可能です
echo $foo;           // foobar
echo $foo,$bar;      // foobarbarbaz

// 複数のパラメータを結合してechoに渡そうとする人もいます
echo 'This ', 'string ', 'was ', 'made ', 'with multiple parameters.', chr(10);
echo 'This ' . 'string ' . 'was ' . 'made ' . 'with concatenation.' . "¥n";

echo <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon. no extra whitespace!
END;

// echo は関数のように動作しないので、以下のコードは正しくありません
($some_var) ? echo 'true' : echo 'false';

// しかし、次の例は動作します
($some_var) ? print 'true' : print 'false'; // print も言語構造ですが、
                                            // 関数のように動作します。なので、
                                            // このコンテキスト中で使用できます
echo $some_var ? 'true': 'false'; // 命令を変更
?>
```

注意

print() と **echo()** の違いに関する簡単な議論については、FAQTs Knowledge Base Article: » http://www.faqts.com/knowledge_base/view.phtml/aid/1/fid/40 を参照してください。

> **注意**: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

参考

- print()
- printf()
- flush()

---

explode

(PHP 4, PHP 5)

explode — 文字列を文字列により分割する

説明

array **explode** (string $delimiter , string $string [, int $limit])

文字列の配列を返します。この配列の各要素は、 string を文字列 delimiter で区切った部分文字列となります。

パラメータ

delimiter

区切り文字列。

string

入力文字列。

limit

limit が指定された場合、返される配列には 最大 limit の要素が含まれ、その最後の要素には string の残りの部分が全て含まれます。

もし limit パラメータが負の場合、 最後の -limit 個の要素を除く全ての構成要素が返されます。

歴史的理由により、implode() はいずれのパラメータ順も受け入れることができますが、 **explode()** はそうできません。 string 引数の前に必ず delimiter 引数がくるように確認する必要があります。

返り値

空の文字列 ("") が delimiter として使用された場合、 **explode()** は **FALSE** を返します。delimiter に引数 string に含まれていない値が含まれている場合、 **explode()** は、引数 string を含む配列を返します。

変更履歴

バージョン	説明
5.1.0	limit に負の数を指定できるようになりました。
4.0.1	limit パラメータが追加されました。

例

Example#1 explode() の例

```
<?php
// 例 1
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
echo $pieces[1]; // piece2

// 例 2
$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user, $pass, $uid, $gid, $gecos, $home, $shell) = explode(":", $data);
echo $user; // foo
echo $pass; // *

?>
```

Example#2 limit パラメータの例

```
<?php
$str = 'one|two|three|four';

// 正の値を持つ limit
print_r(explode('|', $str, 2));

// 負の値を持つ limit (PHP 5.1 以降)
print_r(explode('|', $str, -1));
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => one
    [1] => two|three|four
)
Array
(
    [0] => one
    [1] => two
    [2] => three
)
```

注意

> **注意**: この関数はバイナリデータに対応しています。

参考

- preg_split()
- str_split()
- strtok()
- implode()

---

fprintf

(PHP 5)

fprintf — フォーマットされた文字列をストリームに書き込む

説明

int **fprintf** (resource $handle , string $format [, mixed $args [, mixed $...]])

format によって作成された文字列を handle で指定したストリームに書き込みます。

パラメータ

handle

format

format については、 sprintf() のドキュメントで説明されています。

args

...

返り値

出力された文字列の長さを返します。

例

Example#1 fprintf(): 数値のゼロ埋め

```
<?php
if (!($fp = fopen('date.txt', 'w')))
    return;

fprintf($fp, "%04d-%02d-%02d", $year, $month, $day);
// ISO 形式にフォーマットした日付を date.txt に書き込みます
?>
```

Example#2 fprintf(): 金額のフォーマット

```
<?php
if (!($fp = fopen('currency.txt', 'w')))
    return;

$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money は "123.1" を出力します
$len = fprintf($fp, '%01.2f', $money);
// "123.10" を currency.txt に書き込みます

echo "$len バイトを currency.txt に書き込みました";
// fprintf の返り値を使用して、書き込まれたバイト数を取得します
?>
```

参考

- printf()
- sprintf()
- sscanf()
- fscanf()
- vsprintf()
- number_format()

---

get_html_translation_table

(PHP 4, PHP 5)

get_html_translation_table — htmlspecialchars() および htmlentities() で使用される変換テーブルを返す

説明

array **get_html_translation_table** ([int $table [, int $quote_style]])

get_html_translation_table() は、 htmlspecialchars() および htmlentities() で内部的に使用される変換テーブルを返します。

> **注意**: 特殊文字はいくつかの方法でエンコードすることができます。 例えば、" は ", " もしくは &#x22 としてエンコードすることができます。 **get_html_translation_table()** はこれらに対して 最も一般的な形式を返します。

パラメータ

table

テーブルを指定できるように新規に定義が2つ (HTML_ENTITIES , HTML_SPECIALCHARS)追加されました。 table のデフォルト値は **HTML_SPECIALCHARS** です。

quote_style

htmlspecialchars() および htmlentities() 関数と同様にオプションで 処理する quote_style を指定することが可能です。デフォルトは、**ENT_COMPAT** モードです。これらのモードに関する説明は、 htmlspecialchars()を参照ください。

返り値

変換テーブルを配列で返します。

例

Example#1 変換テーブルの例

```
<?php
$trans = get_html_translation_table(HTML_ENTITIES);
$str = "Hallo & <Frau> & Krämer";
$encoded = strtr($str, $trans);
?>
```

変数 $encoded は次のようになります: "Hallo & <Frau> & Krämer"

参考

- htmlspecialchars()
- htmlentities()
- html_entity_decode()

---

hebrev

(PHP 4, PHP 5)

hebrev — 論理表記のヘブライ語を物理表記に変換する

説明

string **hebrev** (string $hebrew_text [, int $max_chars_per_line])

論理表記のヘブライ語を物理表記に変換します。

この関数は、単語の分断をできるだけ回避しようとします。

パラメータ

hebrew_text

ヘブライ語の入力文字列。

max_chars_per_line

このオプションのパラメータは、 出力される行毎の最大文字数を返します。

返り値

物理表記の文字列を返します。

参考

- hebrevc()

---

hebrevc

(PHP 4, PHP 5)

hebrevc — 論理表記のヘブライ語を、改行の変換も含めて物理表記に変換する

説明

string **hebrevc** (string $hebrew_text [, int $max_chars_per_line])

この関数は、hebrev() に似ていますが、 改行 (\n) を "
\n" に変換するところが異なります。

この関数は、単語の分断をできるだけ回避しようとします。

パラメータ

hebrew_text

ヘブライ語の入力文字列。

max_chars_per_line

このオプションのパラメータは、 出力される行毎の最大文字数を返します。

返り値

物理表記の文字列を返します。

参考

- hebrev()

---

html_entity_decode

(PHP 4 >= 4.3.0, PHP 5)

html_entity_decode — HTML エンティティを適切な文字に変換する

説明

string ***html_entity_decode*** *(string $string [, int $quote_style [, string $charset]])*

html_entity_decode() *は* htmlentities() *の反対で、string にあるすべての HTML エンティティを適切な文字に変換します。*

パラメータ

string

入力文字列。

quote_style

オプションの 2 番目のパラメータ quote_style は 'シングルクォート' および "ダブルクォート" をどのように扱うかを 指定します。以下の 3 つの定数のうちのひとつを指定でき、デフォルトは ***ENT_COMPAT*** *です。*

使用可能な *quote_style* 定数

定数名	説明
ENT_COMPAT	ダブルクォートを変換し、シングルクォートはそのままにします。
ENT_QUOTES	ダブルクォート、シングルクォートの両方を変換します。
ENT_NOQUOTES	ダブルクォート、シングルクォートの両方とも変換しません。

charset

オプションの 3 番目のパラメータ charset の デフォルトは ISO-8859-1 です。これは変換に使用する文字セットを指定します。

PHP 4.3.0 以降では、以下の文字セットがサポートされます。

サポートされる文字セット

文字セット	エイリアス	説明
ISO-8859-1	ISO8859-1	西欧、Latin-1
ISO-8859-15	ISO8859-15	西欧、Latin-9 。Latin-1(ISO-8859-1) に欠けている ユーロ記号やフランス・フィンランドの文字を追加したもの。
UTF-8		ASCII 互換のマルチバイト 8 ビット Unicode 。
cp866	ibm866, 866	DOS 固有のキリル文字セット。 4.3.2 以降でサポートされます。
cp1251	Windows-1251, win-1251, 1251	Windows 固有のキリル文字セット。 4.3.2 以降でサポートされます。
cp1252	Windows-1252, 1252	西欧のための Windows 固有の文字セット。
KOI8-R	koi8-ru, koi8r	ロシア語。4.3.2 以降でサポートされます。
BIG5	950	繁体字中国語。主に台湾で使用されます。
GB2312	936	簡体字中国語。国の標準文字セットです。
BIG5-HKSCS		Big5 に香港の拡張を含めたもの。繁体字中国語。
Shift_JIS	SJIS, 932	日本語。
EUC-JP	EUCJP	日本語。

***注意**: それ以外の文字セットは理解できず、かわりに ISO-8859-1 が使用されます。*

返り値

デコードされた文字列を返します。

変更履歴

バージョン	説明
5.0.0	マルチバイト文字セットをサポートするようになりました。

例

Example#1 HTML エンティティのデコード

```
<?php
$orig = "I'll ¥"walk¥" the <b>dog</b> now";

$a = htmlentities($orig);

$b = html_entity_decode($a);

echo $a; // I'll &quot;walk&quot; the &lt;b&gt;dog&lt;/b&gt; now

echo $b; // I'll "walk" the <b>dog</b> now


// PHP 4.3.0 より前のバージョンでは、このようにします
function unhtmlentities($string)
{
    // 数値エンティティの置換
    $string = preg_replace('~&#x([0-9a-f]+);~ei', 'chr(hexdec("¥¥1"))', $string);
    $string = preg_replace('~&#([0-9]+);~e', 'chr("¥¥1")', $string);
    // 文字エンティティの置換
    $trans_tbl = get_html_translation_table(HTML_ENTITIES);
    $trans_tbl = array_flip($trans_tbl);
    return strtr($string, $trans_tbl);
}

$c = unhtmlentities($a);

echo $c; // I'll "walk" the <b>dog</b> now

?>
```

注意

注意: trim(html_entity_decode(' ')); の結果が空の文字列に ならないことを疑問に思う人もいるでしょう。なぜそうなるのかというと、 デフォルトの文字セット ISO-8859-1 では ' ' エンティティが ASCII コード 32 (これは trim() で取り除かれる) ではなく ASCII コード 160 (0xa0) に変換されるからです。

参考

- htmlentities()
- htmlspecialchars()
- get_html_translation_table()
- urldecode()

---

htmlentities

(PHP 4, PHP 5)

htmlentities — 適用可能な文字を全て HTML エンティティに変換する

説明

string **htmlentities** (string $string [, int $quote_style [, string $charset [, bool $double_encode]]])

この関数はhtmlspecialchars()と同じですが、 HTML エンティティと等価な意味を有する文字をHTMLエンティティに変換します。

もしデコード (逆の処理) をしたい場合、 html_entity_decode() を使用することができます。

パラメータ

string

入力文字列。

quote_style

htmlspecialchars() と同様に、シングルまたは ダブルクオートに関する動作を示すオプションの第2の引数をとります。 これは 3 つの定数のうちの一つとなり、 デフォルトは ENT_COMPAT です。

利用可能な quote_style 定数

定数名	説明
ENT_COMPAT	ダブルクオートのみを変換し、 シングルクオートをそのままにします
ENT_QUOTES	ダブルおよびシングルクオートを共に変換します
ENT_NOQUOTES	ダブルクオートおよびシングルクオートを共に変換しません

charset

htmlspecialchars()と同様に、この関数はオプションの3番目の引数 charset をとり、変換に使用される文字セットを指定可能です。 現在のところ、ISO-8859-1 文字セットがデフォルトの文字エンコーディングとして使用されます。

PHP 4.3.0 以降では、以下の文字セットがサポートされます。

サポートされる文字セット

文字セット	エイリアス	説明
ISO-8859-1	ISO8859-1	西欧、Latin-1
ISO-8859-15	ISO8859-15	西欧、Latin-9 。Latin-1(ISO-8859-1) に欠けている ユーロ記号やフランス・フィンランドの文字を追加したもの。
UTF-8		ASCII 互換のマルチバイト 8 ビット Unicode 。
cp866	ibm866, 866	DOS 固有のキリル文字セット。 4.3.2 以降でサポートされます。
cp1251	Windows-1251, win-1251, 1251	Windows 固有のキリル文字セット。 4.3.2 以降でサポートされます。
cp1252	Windows-1252, 1252	西欧のための Windows 固有の文字セット。
KOI8-R	koi8-ru, koi8r	ロシア語。4.3.2 以降でサポートされます。
BIG5	950	繁体字中国語。主に台湾で使用されます。
GB2312	936	簡体字中国語。国の標準文字セットです。
BIG5-HKSCS		Big5 に香港の拡張を含めたもの。繁体字中国語。
Shift_JIS	SJIS, 932	日本語。
EUC-JP	EUCJP	日本語。

注意: それ以外の文字セットは理解できず、かわりに ISO-8859-1 が使用されます。

double_encode

double_encode をオフにすると、PHP は既存の html エンティティをエンコードしません。 デフォルトでは、既存のエンティティも含めてすべてを変換します。

返り値

エンコードした文字列を返します。

変更履歴

バージョン	説明

バージョン	説明
5.2.3	*double_encode* パラメータが追加されました。
4.1.0	*charset* パラメータが追加されました。
4.0.3	*quote_style* パラメータが追加されました。

例

Example#1 htmlentities() の例

```
<?php
$str = "A 'quote' is <b>bold</b>";

// 出力: A 'quote' is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str);

// 出力: A &#039;quote&#039; is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str, ENT_QUOTES);
?>
```

参考

- html_entity_decode()
- get_html_translation_table()
- htmlspecialchars()
- nl2br()
- urlencode()

---

htmlspecialchars_decode

(PHP 5 >= 5.1.0)

htmlspecialchars_decode — 特殊な HTML エンティティを文字に戻す

説明

string **htmlspecialchars_decode** (string $string [, int $quote_style])

この関数は htmlspecialchars() の反対です。 特殊な HTML エンティティを文字に戻します。

変換されるエンティティは次のものです。&、 " (ENT_NOQUOTES が設定されていない場合)、 ' (ENT_QUOTES が設定されている場合)、 < および >。

パラメータ

string

デコードする文字列。

quote_style

クォートの形式。以下の定数のいずれかです。

quote_style 定数

定数名	説明
ENT_COMPAT	ダブルクォートを変換し、シングルクォートはそのままに します(デフォルト)。
ENT_QUOTES	ダブルクォート、シングルクォートの両方を変換します。
ENT_NOQUOTES	ダブルクォート、シングルクォートの両方をそのままにします。

返り値

デコードされた文字列を返します。

例

Example#1 htmlspecialchars_decode() の例

```
<?php
$str = '<p>this -&gt; &quot;</p>';

echo htmlspecialchars_decode($str);

// クォートが変換されないことに注意しましょう
echo htmlspecialchars_decode($str, ENT_NOQUOTES);
?>
```

上の例の出力は以下となります。

```
<p>this -> "</p>
<p>this -> &quot;</p>
```

参考

---

htmlspecialchars

(PHP 4, PHP 5)

htmlspecialchars — 特殊文字を HTML エンティティに変換する

説明

string **htmlspecialchars** (string $string [, int $quote_style [, string $charset [, bool $double_encode]]])

文字の中には HTML において特殊な意味を持つものがあり、 それらの本来の値を表示したければ HTML の表現形式に変換してやらなければなりません。 この関数は、これらの変換を行った結果の文字列を返します。 これは、日常的な Web プログラミングにおいて最も有用な変換を行います。 全ての HTML 文字エンティティを変換する必要がある場合には、代わりに get_html_translation_table() を使用してください。

この関数は、掲示板やゲストブックなどでユーザが書きこんだテキストから HTML のマークアップ用文字を取り除く場合に有用です。

変換対象となる文字は以下の通りです。

- '&' (アンパサンド) は '&' になります。
- **ENT_NOQUOTES** が設定されていない場合、 '"' (ダブルクォート) は '"'になります。
- **ENT_QUOTES** が設定されている場合のみ、 ''' (シングルクオート) は '''になります。
- '<' (小なり) は '<' になります。
- '>' (大なり) は '>' になります。

パラメータ

string

変換される文字列。

quote_style

オプションの 2 番目の引数 quote_style は、 シングルおよびダブルクオートされた文字をどのように扱うかを指定します。 デフォルトの **ENT_COMPAT** は下位互換性のためのモードで、 ダブルクオートは変換しますがシングルクオートは変換しません。 **ENT_QUOTES** が設定されている場合は、シングルクオートとダブルクオートを共に変換します。 **ENT_NOQUOTES** が設定されている場合は、シングルクオートとダブルクオートは共に変換されません。

charset

変換に使用される文字セットを指定します。 デフォルトの文字セットは、ISO-8859-1 です。

PHP 4.3.0 以降では、以下の文字セットがサポートされます。

サポートされる文字セット

文字セット	エイリアス	説明
ISO-8859-1	ISO8859-1	西欧、Latin-1
ISO-8859-15	ISO8859-15	西欧、Latin-9 。Latin-1(ISO-8859-1) に欠けている ユーロ記号やフランス・フィンランドの文字を追加したもの。
UTF-8		ASCII 互換のマルチバイト 8 ビット Unicode 。
cp866	ibm866, 866	DOS 固有のキリル文字セット。 4.3.2 以降でサポートされます。
cp1251	Windows-1251, win-1251, 1251	Windows 固有のキリル文字セット。 4.3.2 以降でサポートされます。
cp1252	Windows-1252, 1252	西欧のための Windows 固有の文字セット。
KOI8-R	koi8-ru, koi8r	ロシア語。4.3.2 以降でサポートされます。
BIG5	950	繁体字中国語。主に台湾で使用されます。
GB2312	936	簡体字中国語。国の標準文字セットです。
BIG5-HKSCS		Big5 に香港の拡張を含めたもの。繁体字中国語。
Shift_JIS	SJIS, 932	日本語。
EUC-JP	EUCJP	日本語。

注意: それ以外の文字セットは理解できず、かわりに ISO-8859-1 が使用されます。

double_encode

double_encode をオフにすると、PHP は既存の html エンティティをエンコードしません。 デフォルトでは、既存のエンティティも含めてすべてを変換します。

返り値

変換後の文字列を返します。

変更履歴

バージョン	説明
5.2.3	double_encode パラメータが追加されました。
4.1.0	charset パラメータが追加されました。

バージョン	説明
4.0.3	*quote_style* パラメータが追加されました。

例

Example#1 htmlspecialchars() の例

```
<?php
$new = htmlspecialchars("<a  href='test'>Test</a>", ENT_QUOTES);
echo $new; //  &lt;a  href=&#039;test&#039;&gt;Test&lt;/a&gt;
?>
```

注意

> **注意**: この関数は上記のあげたもの以外に関しては一切の変換を行わないことに注意してください。 すべての変換を行うには *htmlentities()* を参照してください。

参考

- *get_html_translation_table()*
- *htmlspecialchars_decode()*
- *strip_tags()*
- *htmlentities()*
- *nl2br()*

---

implode

(PHP 4, PHP 5)

implode — 配列要素を文字列により連結する

説明

string **implode** (string $glue , array $pieces)

配列の要素を *glue* 文字列で連結します。

> **注意**: **implode()**は、歴史的な理由により、引数をどちら の順番でも受けつけることが可能です。しかし、 *explode()* との統一性の観点からは、 ドキュメントに記述された引数の順番を使用する方が混乱が少なくなるでしょう。

パラメータ

glue

デフォルトは空文字 ('') です。 これは **implode()** の好ましい使用法ではありません。 下位互換性のため、常に 2 つのパラメータを使用することが推奨されています。

pieces

返り値

すべての配列要素の順序を変えずに、各要素間に *glue* 文字列をはさんで 1 つの文字列にして返します。

変更履歴

バージョン	説明
4.3.0	*glue* パラメータがオプションとなりました。

例

Example#1 implode() の例

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);

echo $comma_separated; //  lastname,email,phone

?>
```

注意

> **注意**: この関数はバイナリデータに対応しています。

参考

- *explode()*
- *split()*

---

join

(PHP 4, PHP 5)

join — implode() のエイリアス

説明

この関数は次の関数のエイリアスです。 implode().

---

levenshtein

(PHP 4 >= 4.0.1, PHP 5)

levenshtein — 二つの文字列のレーベンシュタイン距離を計算する

説明

int **levenshtein** (string $str1 , string $str2 [, int $cost_ins], int $cost_rep , int $cost_del)

レーベンシュタイン距離は、str1 を str2 に変換するために置換、挿入、削除 しなければならない最小の文字数として定義されます。アルゴリズムの複雑さは、 O(m*n) です。 ここで、n および m はそれぞれ str1 および str2 の長さです (O(max(n,m)**3) となる similar_text() よりは良いですが、 まだかなりの計算量です)。

上記の最も簡単な形式では、この関数はパラメータとして引数を二つだけとり、 str1 から str2 に変換する際に必要な 挿入、置換、削除演算の数のみを計算します。

2 番目の形式では、挿入、置換、削除演算のコストを定義する 3 番目のパラメータが追加されます。この形式は 1 番目の形式より一般的で 汎用性が高いですが、効率的ではありません。

パラメータ

str1

レーベンシュタイン距離を計算する文字列のひとつ。

str2

レーベンシュタイン距離を計算する文字列のひとつ。

cost_ins

挿入のコストを定義します。

cost_rep

置換のコストを定義します。

cost_del

削除のコストを定義します。

返り値

この関数は、引数で指定した二つの文字列のレーベンシュタイン距離を返します。 引数文字列の一つが 255 文字の制限より長い場合に -1 を返します。

例

Example#1 levenshtein() の例

```php
<?php
// スペルミスした単語を入力します
$input = 'carrrot';

// チェックするための単語の配列
$words  = array('apple','pineapple','banana','orange',
                'radish','carrot','pea','bean','potato');

// まだ最短距離は見つかっていません
$shortest = -1;

// 最短距離を見つけるため単語をループします
foreach ($words as $word) {

    // 入力した単語と現在の単語の距離を
    // 計算します
    $lev = levenshtein($input, $word);

    // マッチするかどうかチェックします
    if ($lev == 0) {

        // 最短な単語はこれだ (マッチした)
        $closest = $word;
        $shortest = 0;

        // ループを抜ける; マッチしたものを見つけました
        break;
    }

    // もし距離が次に見つけた最短距離よりも短い場合、
```

```
    // もしくは次の最短の単語がまだ見つかっていない場合
    if ($lev <= $shortest || $shortest < 0) {
        // 最短のマッチと最短距離をセットします
        $closest  = $word;
        $shortest = $lev;
    }
}

echo "入力した単語: $input\n";
if ($shortest == 0) {
    echo "一致するものが見つかりました: $closest\n";
} else {
    echo "もしかして: $closest\n";
}

?>
```

上の例の出力は以下となります。

```
入力した単語: carrrot
もしかして: carrot
```

参考

- soundex()
- similar_text()
- metaphone()

---

localeconv

(PHP 4 >= 4.0.5, PHP 5)

localeconv — 数値に関するフォーマット情報を得る

説明

array **localeconv** (void)

ローカルな数値および通貨フォーマット情報を有する連想配列を返します。

返り値

localeconv() は、 setlocale() で設定された現在のロケールに基づきデータを返します。 返される連想配列は、次のフィールドを有します。

配列要素	説明
decimal_point	小数点文字
thousands_sep	千毎の区切り文字
grouping	数値集合を有する配列
int_curr_symbol	国際通貨記号 (すなわち、USD)
currency_symbol	ローカルな通貨記号 (すなわち、$)
mon_decimal_point	通貨用の小数点文字
mon_thousands_sep	通貨用の千毎の区切り文字
mon_grouping	通貨集合を有する配列
positive_sign	正の値を表す記号
negative_sign	負の値を表す記号
int_frac_digits	国際分割桁
frac_digits	ローカルな分割桁
p_cs_precedes	currency_symbol が正の値を前に置く場合に**TRUE**、後に置く場合に **FALSE**
p_sep_by_space	正の値から currency_symbol を1文字の空白で区切る場合に**TRUE**、 そうでない場合に**FALSE**
n_cs_precedes	currency_symbol が負の値を前に置く場合に**TRUE**、後に置く場合に **FALSE**
n_sep_by_space	負の値から currency_symbol を1文字の空白で区切る場合に**TRUE**、 そうでない場合に**FALSE**
p_sign_posn	• 0 - 量および通貨記号を括る括弧 • 1 - 量および通貨記号の前に置く符号文字列 • 2 - 量および通貨記号の後に置く符号文字列 • 3 - 通貨記号の直前に置く符号文字列 • 4 - 通貨記号の直後に置く符号文字列
n_sign_posn	• 0 - 量および通貨記号を括る括弧 • 1 - 量および通貨記号の前に置く符号文字列 • 2 - 量および通貨記号の後に置く符号文字列

配列要素	説明
	• 3 - 通貨記号の直前に置く符号文字列 • 4 - 通貨記号の直後に置く符号文字列

n_sign_posn や *n_sign_posn* は、フォーマッタオプションの文字列を含みます。それぞれの数字は 上に一覧されている条件の *1* つを表します。

*grouping*フィールドには、グループ化する方法を表す数字を定義する配 列が含まれます。例えば、*nl_NL* ロケール用の通貨 *grouping*フィールド *(UTF-8* モードでのユーロ記号*)* には、 *3*、*3*を値とする要素数*2*の配列が含まれます。この配列のより高い添字に は、より左側のグループに関するものが含まれます。 ある配列要素が、**CHAR_MAX** に等しい場合、 さらにグループは行われません。配列要素が*0*に等しい場合、 前の要素が使用されています。

例

Example#1 localeconv() の例

```
<?php
if (false !== setlocale(LC_ALL, 'nl_NL.UTF-8@euro')) {
    $locale_info = localeconv();
    print_r($locale_info);
}
?>
```

上の例の出力は以下となります。

```
Array
(
     [decimal_point] => .
     [thousands_sep] =>
     [int_curr_symbol] => EUR
     [currency_symbol] => €
     [mon_decimal_point] => ,
     [mon_thousands_sep] =>
     [positive_sign] =>
     [negative_sign] => -
     [int_frac_digits] => 2
     [frac_digits] => 2
     [p_cs_precedes] => 1
     [p_sep_by_space] => 1
     [n_cs_precedes] => 1
     [n_sep_by_space] => 1
     [p_sign_posn] => 1
     [n_sign_posn] => 2
     [grouping] => Array
        (
        )

     [mon_grouping] => Array
        (
            [0] => 3
            [1] => 3
        )

)
```

参考

- setlocale()

---

ltrim

(PHP 4, PHP 5)

ltrim — 文字列の最初から空白（もしくはその他の文字）を取り除く

説明

string **ltrim** *(string $str [, string $charlist])*

文字列の最初から空白（もしくはその他の文字）を取り除きます。

パラメータ

str

入力文字列。

charlist

charlist パラメータにより、削除する 文字を指定することも可能です。削除したい全ての文字をリストに してください。*..*を文字の範囲を指定する際に 使用可能です。

返り値

この関数は文字列の最初から空白文字を取り除き、 取り除かれた文字列を返します。*2*番目のパラメータを指定しない場合、 **ltrim()**は以下の文字を削除します。

- " " (ASCII 32 (0x20)), 通常の空白。
- "¥t" (ASCII 9 (0x09)), タブ。
- "¥n" (ASCII 10 (0x0A)), 改行。
- "¥r" (ASCII 13 (0x0D)), 復帰。
- "¥0" (ASCII 0 (0x00)), NUL バイト。
- "¥x0B" (ASCII 11 (0x0B)), 垂直タブ。

変更履歴

バージョン	説明
4.1.0	charlist パラメータが追加されました。

例

Example#1 ltrim() の使用例

```
<?php

$text = "¥t¥tThese are a few words :) ...  ";
$binary = "¥x09Example string¥x0A";
$hello  = "Hello World";
var_dump($text, $binary, $hello);

print "¥n";


$trimmed = ltrim($text);
var_dump($trimmed);

$trimmed = ltrim($text, " ¥t.");
var_dump($trimmed);

$trimmed = ltrim($hello, "Hdle");
var_dump($trimmed);

// ASCII 制御文字 (0 から 31 まで) を
// $binary の先頭から取り除きます
$clean = ltrim($binary, "¥x00..¥x1F");
var_dump($clean);

?>
```

上の例の出力は以下となります。

```
string(32) "          These are a few words :) ...  "
string(16) "     Example string
"
string(11) "Hello World"

string(30) "These are a few words :) ...  "
string(30) "These are a few words :) ...  "
string(7) "o World"
string(15) "Example string
"
```

参考

- trim()
- rtrim()

---

md5_file

(PHP 4 >= 4.2.0, PHP 5, PECL hash:1.1-1.3)

md5_file — 指定したファイルのMD5ハッシュ値を計算する

説明

string **md5_file** (string $filename [, bool $raw_output])

» RSA Data Security, Inc. MD5 メッセージダイジェストアルゴリズムを用いて filename パラメータで指定したファイルの MD5ハッシュを計算し、そのハッシュを返します。 ハッシュは、32 文字の 16 進数です。

パラメータ

filename

ファイル名

raw_output

TRUE の場合、長さ 16 の生のバイナリフォーマットで ダイジェストを返します。デフォルトは **FALSE** です。

返り値

成功時は文字列、そうでなければ **FALSE**

変更履歴

バージョン	説明
5.0.0	raw_output パラメータが追加されました
5.1.0	ストリーム API を使用した関数に変更されました。 これは md5_file('http://example.com/..') のようなラッパーで利用可能であることを意味します。

参考

- md5()
- sha1_file()
- crc32()

---

md5

(PHP 4, PHP 5, PECL hash:1.1-1.3)

md5 — 文字列のmd5ハッシュ値を計算する

説明

string **md5** (string $str [, bool $raw_output])

» RSA Data Security, Inc. の MD5メッセージダイジェストアルゴリズム を用いて str の MD5 ハッシュ値を計算し、 そのハッシュを返します。

パラメータ

str

文字列。

raw_output

オプションのraw_output に **TRUE** が指定された場合、 md5 ダイジェストが 16 バイト長のバイナリ形式で返されます。 デフォルトは **FALSE** です。

返り値

32 文字の 16 進数からなるハッシュを返します。

変更履歴

バージョン	説明
5.0.0	raw_output パラメータが追加されました。

例

Example#1 md5() の例

```
<?php
$str = 'apple';

if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f') {
    echo "Would you like a green or red apple?";
    exit;
}
?>
```

参考

- sha1_file()
- crc32()
- sha1()

---

metaphone

(PHP 4, PHP 5)

metaphone — 文字列の metaphone キーを計算する

説明

string **metaphone** (string $str [, int $phones])

str の metaphone キーを計算します。

soundex() と同様に metaphone は、 発音が似た単語について同じキーを作成します。metaphone は、 英語の発音の基本的ルールを知っているので、 soundex() よりも正確です。 metaphone が生成するキーは可変長です。

metaphone は、Lawrence Philips <lphilips at verity dot com> により 開発されました。["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995] で解説されています。

パラメータ

str

入力文字列。

phones

返り値

metaphone キーを文字列で返します。

---

money_format

(PHP 4 >= 4.3.0, PHP 5)

money_format — 数値を金額文字列にフォーマットする

説明

string **money_format** (string $format , float $number)

money_format() は、number をフォーマットして返します。この関数は C のライブラリ関数 **strfmon()** をラップしたものですが、一度に 変換できる数値がひとつだけであるという点が異なります。

パラメータ

format

フォーマット指定の書式は以下の順になります。

- % 文字
- フラグ(オプション)
- フィールドの幅(オプション)
- 左精度(オプション)
- 右精度(オプション)
- 変換文字(必須)

フラグ

以下のフラグのうちひとつあるいは複数が使用可能です。

=f

文字 = の後に続く(シングルバイトの)文字 f が、数値埋め文字として使用されます。 デフォルトはスペース文字です。

^

グループ化文字(現在のロケールで定義されている)を使用しないようにします。

+ あるいは (

正の数、負の数の書式を指定します。+ が使用された場合、 + および - に該当する そのロケールの符号マークが使用されます。(が使用された場合、負の数は括弧で囲まれます。何も指定しなかった場合、 デフォルトは + です。

!

出力文字列から通貨記号を除きます。

-

指定した場合、すべてのフィールドを左詰め(右側に数値埋め文字が追加される) にします。デフォルトはこれと反対で、すべてのフィールドを右詰め (左側に数値埋め文字が追加される)にします。

フィールドの幅

w

10 進の数値文字列で、フィールドの幅の最小値を指定します。フラグ - が使用されていない限り、フィールドは 右詰めとなります。デフォルト値は 0(ゼロ) です。

左精度

#n

10 進の基準文字(例: 小数点)より左側の最大の桁数 (n) を指定します。これは通常、 n より少ない桁数の数値に対して 数値埋め文字を使用

することで、出力の桁位置をそろえるために 使用されます。実際の桁数が n より 大きい場合、この設定は無視されます。

^ フラグでグループ化文字が抑止されていない場合、(もし存在するなら)数値埋め文字が追加される前にグループ化文字が 挿入されます。グループ化文字は数値埋め文字には適用されません。 たとえ数値埋め文字が数字であったとしても同様です。

位置あわせを確実にするため、出力中の数値の前後に表れる文字(たとえば 通貨記号や符号など)は、必要に応じて(正の数と負の数の長さをそろえるなど の理由で)スペース文字が付加されることがあります。

右精度

.p

ピリオドに続く数値(p) で、10 進の基準文字以降の桁数を指定します。 p の値が 0(ゼロ)であった場合、基準文字と それ以降の数値は省略されます。右精度が指定されていない場合、 使用中の現在のロケールからデフォルト値を検出します。 フォーマットされる数値は、フォーマット前にこの桁数に丸められます。

変換文字

i

ロケールの国際通貨フォーマット(例: USA ロケールでは USD 1,234.56)によってフォーマットします。

n

ロケールの国内通貨フォーマット(例: de_DE ロケールでは DM1.234,56)によってフォーマットします。

%

% 文字を返します。

number

フォーマットする数値。

返り値

フォーマットした文字列を返します。 フォーマット文字列の前後の文字は、そのまま返されます。

注意

注意: システムで strfmon が使用可能な場合のみ **money_format()** 関数が定義されます。例えば、Windows では strfmon は使用できません。 そのため **money_format()** は Windows では 定義されていません。

注意: ロケール設定のうち、**LC_MONETARY** カテゴリの内容が この関数の振る舞いに影響します。この関数を使用する前に、setlocale() で適切なデフォルトロケールを 設定してください。

例

Example#1 money_format() の例

この関数の使用法を詳しく説明するために、 さまざまなロケールおよびフォーマット指定を使用します。

```php
<?php

$number = 1234.56;

// en_US ロケールの国際フォーマットで表示します
setlocale(LC_MONETARY, 'en_US');
echo money_format('%i', $number) . "¥n";
//  USD 1,234.56

// イタリアの国内フォーマットで小数点以下 2 桁で表示します
setlocale(LC_MONETARY, 'it_IT');
echo money_format('%.2n', $number) . "¥n";
//  L. 1.234,56

// 負の数を使用します
$number = -1234.5672;

// US の国際フォーマットで、負の数には () を使用して
// 左精度を 10 桁にします
setlocale(LC_MONETARY, 'en_US');
echo money_format('%(#10n', $number) . "¥n";
// ($          1,234.57)

// 上と同じですが、それに加えて右精度を 2 桁
// 数値埋め文字として '*' を使用します
echo money_format('%=*(#10.2n', $number) . "¥n";
//  ($********1,234.57)

// 左詰め、幅 14 桁、左精度 8 桁、右精度 2 桁、グループ化文字なしで
// de_DE ロケールの国際フォーマットを使用します。
setlocale(LC_MONETARY, 'de_DE');
echo money_format('%=*^-14#8.2i', 1234.56) . "¥n";
//  DEM 1234,56****

// 変換指定の前後に宣伝文句を追加します
setlocale(LC_MONETARY, 'en_GB');
$fmt = 'The final value is %i (after a 10%% discount)';
echo money_format($fmt, 1234.56) . "¥n";
// The final value is  GBP 1,234.56 (after a 10% discount)

?>
```

参考

- setlocale()
- sscanf()
- sprintf()
- printf()
- number_format()

---

nl_langinfo

(PHP 4 >= 4.0.7, PHP 5)

nl_langinfo — 言語およびロケール情報を検索する

説明

string **nl_langinfo** (int $item)

nl_langinfo() はロケールカテゴリの独立した要素にアクセスするために使用されます。 localeconv() と異なり、全ての要素を返します。
nl_langinfo() はいかなる特定要素も取得可能です。

パラメータ

item

item は要素の整数値、もしくは要素の定数名です。 以下は、使用される item に対する定数名と説明の一覧です。 これらの定数のいくつかは特定のロケールに対して未定義、 もしくは値がありません。

nl_langinfo 定数

定数	説明
LC_TIME カテゴリの定数	
ABDAY_(1-7)	一週間中の n 番目の曜日名の略式表記
DAY_(1-7)	一週間中の n 番目の曜日名 (DAY_1 = 日曜日)
ABMON_(1-12)	n 番目の月の名前の略式表記
MON_(1-12)	n 番目の月の名前
AM_STR	午前を表す文字列
PM_STR	午後を表す文字列
D_T_FMT	strftime() で日時を表すためのフォーマット文字列として使用することが可能な文字列
D_FMT	strftime() で日付を表すためのフォーマット文字列として使用することが可能な文字列
T_FMT	strftime() で時刻を表すためのフォーマット文字列として使用することが可能な文字列
T_FMT_AMPM	strftime() でAM/PM付き 12 時間表記を表すためのフォーマット文字列として使用することが可能な文字列
ERA	ロケール固有の元号付きフォーマット
ERA_YEAR	ロケール固有の元号付きフォーマットでの年
ERA_D_T_FMT	ロケール固有の元号付きフォーマットでの日時 (strftime() で使用可能な文字列)
ERA_D_FMT	ロケール固有の元号付きフォーマットでの日付 (strftime() で使用可能な文字列)
ERA_T_FMT	ロケール固有の元号付きフォーマットでの時刻 (strftime() で使用可能な文字列)
LC_MONETARY カテゴリの定数	
INT_CURR_SYMBOL	国際通貨記号
CURRENCY_SYMBOL	地域通貨記号
CRNCYSTR	CURRENCY_SYMBOL と同じ値
MON_DECIMAL_POINT	小数点文字
MON_THOUSANDS_SEP	1000 単位桁区切り (3 桁ごとのグループ化) の区切り文字
MON_GROUPING	'grouping' と同じ
POSITIVE_SIGN	正値の表示に使用される記号
NEGATIVE_SIGN	負値の表示に使用される記号
INT_FRAC_DIGITS	国際的な方法で表現する際の小数点以下の桁数
FRAC_DIGITS	地域的な方法で表現する際の小数点以下の桁数
P_CS_PRECEDES	CURRENCY_SYMBOL の前に正値がある場合 1 を返す
P_SEP_BY_SPACE	CURRENCY_SYMBOL と正値がスペースで区切られる場合 1 を返す
N_CS_PRECEDES	CURRENCY_SYMBOL の前に負値がある場合 1 を返す
N_SEP_BY_SPACE	CURRENCY_SYMBOL と負値がスペースで区切られる場合 1 を返す
P_SIGN_POSN N_SIGN_POSN	• 量および通貨記号を括弧で括る場合、0 を返す • 量および通貨記号の前に符号文字列を置く場合、1 を返す • 量および通貨記号の後に符号文字列を置く場合、2 を返す • 通貨記号の直前に符号文字列を置く場合、3 を返す • 通貨記号の直後に符号文字列を置く場合、4 を返す
LC_NUMERIC カテゴリの定数	
DECIMAL_POINT	小数点文字
RADIXCHAR	DECIMAL_POINT と同じ値
THOUSANDS_SEP	1000 単位桁区切り (3 桁ごとのグループ化) の区切り文字
THOUSEP	THOUSANDS_SEP と同じ値
GROUPING	
LC_MESSAGES カテゴリの定数	
YESEXPR	'はい' の入力にマッチさせるための正規表現
NOEXPR	'いいえ' の入力にマッチさせるための正規表現
YESSTR	'はい' のための出力文字列
NOSTR	'いいえ' のための出力文字列
LC_CTYPE カテゴリの定数	
CODESET	文字エンコーディング名の文字列を返す

返り値

*要素を文字列で返します。item が有効でない場合は **FALSE** を返します。*

注意

> ***注意**: この関数は Windows 環境にはまだ実装されていません。*

参考

- *setlocale()*
- *localeconv()*

---

nl2br

(PHP 4, PHP 5)

nl2br — 改行文字の前に HTML の改行タグを挿入する

説明

string ***nl2br*** *(string $string)*

string に含まれるすべての改行文字の前に '<br />' を挿入して返します。

パラメータ

string

入力文字列。

返り値

変更後の文字列を返します。

変更履歴

バージョン	説明
4.0.5	nl2br() は XHTML 準拠となりました。 PHP 4.0.5 より前の全てのバージョンでは、 nl2br() は、*string* の全ての改行記号の前に ' ' の代わりに ' ' を挿入して返します。

例

Example#1 nl2br() の使用法

```
<?php
echo nl2br("foo isn't¥n bar");
?>
```

上の例の出力は以下となります。

```
foo isn't<br />
 bar
```

参考

- *htmlspecialchars()*
- *htmlentities()*
- *wordwrap()*
- *str_replace()*

---

number_format

(PHP 4, PHP 5)

number_format — 数字を千位毎にグループ化してフォーマットする

説明

string ***number_format*** *(float $number [, int $decimals [, string $dec_point]], string $thousands_sep)*

number_format() *は number をフォーマットして返します。 この関数は 1 つか 2 つもしくは 4 つのパラメータを受け取ります (3 つはありません) :*

パラメータが 1 つだけ渡された場合、 number は千位毎にカンマ (",") が追加され、 小数なしでフォーマットされます。

パラメータが 2 つ渡された場合、number は decimals 桁の小数の前にドット (".") 、 千位毎にカンマ (",") が追加されてフォーマットされます。

パラメータが 4 つ全て渡された場合、number はドット (".") の代わりに dec_point が decimals 桁の小数の前に、千位毎にカンマ (",") の代わりに thousands_sep が追加されてフォーマットされます。

パラメータ

number

フォーマットする数値。

decimals

小数点以下の桁数。

dec_point

小数点を表す区切り文字。

thousands_sep

千位毎の区切り文字。

thousands_sep は最初の文字だけが使用されます。 例えば、数字の 1000 に対する thousands_sep として bar を使用した場合、 ***number_format()*** *は 1b000 を返します。*

例

Example#1 number_format() の例

例えばフランスの表記法では、通常カンマ (',') を小数の区切りとした二桁の小数と、千位毎の区切りとしてスペース (' ') を用います。これを実現するには次のようにします。

```
<?php

$number = 1234.56;

// 英語での表記 (デフォルト)
$english_format_number = number_format($number);
// 1,235

// フランスの表記
$nombre_format_francais = number_format($number, 2, ',', ' ');
// 1 234,56

$number = 1234.5678;

// 千位毎の区切りがない英語での表記
$english_format_number = number_format($number, 2, '.', '');
// 1234.57

?>
```

参考

- money_format()
- sprintf()
- printf()
- sscanf()

---

ord

(PHP 4, PHP 5)

ord — 文字の ASCII 値を返す

説明

int **ord** (string $string)

string の先頭文字の ASCII 値を返します。

この関数は chr() と逆の動作をします。

パラメータ

string

文字。

返り値

ASCII 値を返します。

例

Example#1 ord() の例

```
<?php
$str = "¥n";
if (ord($str) == 10) {
    echo "\$str の先頭は改行文字です。\n";
}
?>
```

参考

- chr()
- » ASCII 値の表

---

parse_str

(PHP 4, PHP 5)

parse_str — 文字列を処理し、変数に代入する

説明

void **parse_str** (string $str [, array &$arr])

URL 経由で渡されるクエリ文字列と同様に str を処理し、現在のスコープに変数をセットします。

注意: 現在の QUERY_STRING を取得するには、変数 $_SERVER['QUERY_STRING'] を使用する事ができます。また、 PHP の外部から来る変数 のセクションも読んでください。

注意: magic_quotes_gpc の設定が、この関数の出力に影響を与えます。というのも **parse_str()** が使用している仕組みは PHP が $_GET や $_POST などの設定に使用しているものと同じだからです。

パラメータ

str

入力文字列。

arr

2 番目の引数 arr が指定された場合、 変数は、代わりに配列の要素としてこの変数に保存されます。

返り値

値を返しません。

変更履歴

バージョン	説明
4.0.3	arr パラメータが追加されました。

例

Example#1 parse_str() の使用法

```
<?php
$str = "first=value&arr[]=foo+bar&arr[]=baz";
parse_str($str);
echo $first;  // value
echo $arr[0]; // foo bar
echo $arr[1]; // baz

parse_str($str, $output);
echo $output['first'];  // value
echo $output['arr'][0]; // foo bar
echo $output['arr'][1]; // baz

?>
```

参考

- parse_url()
- pathinfo()
- http_build_query()
- get_magic_quotes_gpc()
- urldecode()

---

print

(PHP 4, PHP 5)

print — 文字列を出力する

説明

int **print** (string $arg)

arg を出力します。

print()は実際には関数ではありません（言語構造です）。このため、引数を括弧で括る必要はありません。

print() と echo() の違いに関するちょっとした議論については、 FAQTs Knowledge Base の次の記事を参照ください : » http://www.faqts.com/knowledge_base/view.phtml/aid/1/fid/40

パラメータ

arg

入力データ。

返り値

常に 1 を返します。

例

Example#1 print() の例

```
<?php
```

```
print("Hello World");

print "print() also works without parentheses.";

print "This spans
multiple lines. The newlines will be
output as well";

print "This spans¥nmultiple lines. The newlines will be¥noutput as well.";

print "escaping characters is done ¥"Like this¥".";

// print文の中で変数を使用することが可能です。
$foo = "foobar";
$bar = "barbaz";

print "foo is $foo"; // foo is foobar

// 配列も使用可能です。
$bar = array("value" => "foo");

print "this is {$bar['value']} !"; // this is foo !

// シングルクオートを使用すると値ではなく変数名が出力されます。
print 'foo is $foo'; // foo is $foo

// 他の文字を使用しない場合、変数だけを出力することが可能です。
print $foo;           // foobar

print <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;
?>
```

注意

注意: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

参考

- echo()
- printf()
- flush()

---

printf

(PHP 4, PHP 5)

printf — フォーマット済みの文字列を出力する

説明

```
int printf ( string $format [, mixed $args [, mixed $... ]] )
```

format にしたがって、出力を生成します。

パラメータ

format

format についての説明は sprintf() を参照ください。

args

...

返り値

出力した文字列の長さを返します。

参考

- print()
- sprintf()
- vprintf()
- sscanf()
- fscanf()
- flush()

---

quoted_printable_decode

(PHP 4, PHP 5)

quoted_printable_decode — quoted-printable 文字列を 8 ビット文字列に変換する

説明

string **quoted_printable_decode** (string $str)

この関数は、quoted printable 文字列をデコードし、 8 ビットバイナリ文字列を返します (» RFC2821 の section 4.5.2 ではなく » RFC2045 の section 6.7 によれば、付随するピリオドは行の開始から削除されません) 。

この関数は imap_qprint() に似ていますが、 動作に IMAP モジュールを必要としないという違いがあります。

パラメータ

str

入力文字列。

返り値

8 ビットバイナリ文字列を返します。

---

quotemeta

(PHP 4, PHP 5)

quotemeta — メタ文字をクォートする

説明

string **quotemeta** (string $str)

文字列 str について、

. ¥ + * ? [^] ($)

の前にバックスラッシュ文字 (¥) でクォートして返します。

パラメータ

str

入力文字列。

返り値

メタ文字をクォートした文字列を返します。

注意

注意: この関数はバイナリデータに対応しています。

参考

- addslashes()
- addcslashes()
- htmlentities()
- htmlspecialchars()
- nl2br()
- stripslashes()
- stripcslashes()
- ereg()

---

rtrim

(PHP 4, PHP 5)

rtrim — 文字列の最後から空白 (もしくは他の文字) を削除する

説明

string **rtrim** (string $str [, string $charlist])

この関数は文字列 str の最後から空白文字を取り除き、 取り除かれた文字列を返します。

2 番目のパラメータを指定しない場合、 **rtrim()** は以下の文字を削除します。

- " " (ASCII 32 (0x20)), 通常の空白。
- "¥t" (ASCII 9 (0x09)), タブ。

- "¥n" (ASCII 10 (0x0A)), 改行。
- "¥r" (ASCII 13 (0x0D)), 復帰。
- "¥0" (ASCII 0 (0x00)), NUL バイト。
- "¥x0B" (ASCII 11 (0x0B)), 垂直タブ。

パラメータ

str

入力文字列。

charlist

charlist パラメータにより、 削除する文字を指定することも可能です。 削除したい全ての文字をリストにしてください。.. を文字の範囲を指定する際に使用可能です。

返り値

変更後の文字列を返します。

変更履歴

バージョン	説明
4.1.0	charlist パラメータが追加されました。

例

Example#1 rtrim() の使用例

```
<?php

$text = "¥t¥tThese are a few words :) ...  ";
$binary = "¥x09Example string¥x0A";
$hello  = "Hello World";
var_dump($text, $binary, $hello);

print "¥n";

$trimmed = rtrim($text);
var_dump($trimmed);

$trimmed = rtrim($text, " ¥t.");
var_dump($trimmed);

$trimmed = rtrim($hello, "Hdle");
var_dump($trimmed);

// ASCII 制御文字 (0 から 31 まで) を
// $binary の末尾から取り除きます
$clean = rtrim($binary, "¥x00..¥x1F");
var_dump($clean);

?>
```

上の例の出力は以下となります。

```
string(32) "          These are a few words :) ...  "
string(16) "     Example string
"
string(11) "Hello World"

string(30) "          These are a few words :) ..."
string(26) "          These are a few words :)"
string(9) "Hello Wor"
string(15) "     Example string"
```

参考

- trim()
- ltrim()

---

setlocale

(PHP 4, PHP 5)

setlocale — ロケール情報を設定する

説明

```
string setlocale ( int $category , string $locale [, string $... ] )
string setlocale ( int $category , array $locale )
```

ロケール情報を設定します。

パラメータ

category

category は、名前付きの定数(または文字列)であり、 ロケール設定により影響を受ける関数のカテゴリを指定します。

- *LC_ALL 以下のものすべて*
- *LC_COLLATE 文字列の比較用。strcoll() 参照*
- *LC_CTYPE 文字の分類と変換。たとえば strtoupper()*
- *LC_MONETARY localeconv() 用*
- *LC_NUMERIC 数字の区切り文字用(localeconv() も参照ください)*
- *LC_TIME 日時。strftime()でフォーマットに使用*
- *LC_MESSAGES システムの応答用(PHP が libintl とともにコンパイルされている場合のみ使用可能)*

locale

locale が NULL もしくは空の文字列 "" の場合、ロケール名は上記のカテゴリと同じ名前の環境変数の値、 または環境変数 "LANG" からセットされます。

locale が "0" の場合、 ロケール設定は適用されず、単に現在の設定が返されます。

locale が配列もしくは追加のパラメータが続く場合、 それぞれの配列要素もしくはパラメータは成功するまで新規ロケールとしてセットされます。これは、ロケールが異なるシステムで異なる名前を持っている、 もしくはロケールが利用できない場合のフォールバックを提供するといった場合に有用です。

...

返り値

現在の新しいロケールを返します。ロケール機能が未実装、 指定されたロケールが存在しない、カテゴリ名が無効などの場合は FALSE を返します。

また、カテゴリ名が無効の場合は警告メッセージが発生します。 カテゴリやロケール名は、» RFC 1766 や » ISO 639 にあります。 ロケールの命名方式は、システムによって異なります。

> ***注意**: setlocale() の戻り値は、 PHP が実行されているシステムに依存します。 システムの setlocale 関数が返す値を返すためです。*

変更履歴

バージョン	説明
4.3.0	複数のロケールを渡せるようになりました。
4.2.0	*category* に文字列を指定することが非推奨となりました。 代わりに上記の定数を使用してください。これらを文字列として (クオートして) 渡すと、警告メッセージが表示されます。

例

Example#1 setlocale() の例

```
<?php
/* ロケールをオランダ語に設定 */
setlocale(LC_ALL, 'nl_NL');

/* 出力: vrijdag 22 december 1978 */
echo strftime("%A %e %B %Y", mktime(0, 0, 0, 12, 22, 1978));

/* PHP 4.3.0 以降、ドイツに対して利用可能な異なるロケール名を使用する */
$loc_de = setlocale(LC_ALL, 'de_DE@euro', 'de_DE', 'de', 'ge');
echo "Preferred locale for german on this system is '$loc_de'";
?>
```

Example#2 Windows での setlocale() の例

```
<?php
/* ロケールをオランダ語に設定 */
setlocale(LC_ALL, 'nld_nld');

/* 出力: vrijdag 22 december 1978 */
echo strftime("%A %d %B %Y", mktime(0, 0, 0, 12, 22, 1978));

/* PHP 4.3.0 以降、ドイツに対して利用可能な異なるロケール名を使用する */
$loc_de = setlocale(LC_ALL, 'de_DE@euro', 'de_DE', 'deu_deu');
echo "Preferred locale for german on this system is '$loc_de'";
?>
```

注意

警告

ロケール情報は、スレッド毎ではなくプロセス毎に維持されます。 もし PHP を IIS や Windows 用 Apache のようなマルチスレッドサーバ API 上で動作させている場合、 スクリプトを実行している間にロケールの設定が突然変わるのを 経験するかも知れませんが、スクリプト自身は決して setlocale() 自身をコールしていません。 これは同時に同一プロセスの異なるスレッドで実行されている他のスクリプトが setlocale() を使用してプロセスワイドなロケールを変更する事により発生します。

ヒント

Windows ユーザは Microsoft の MSDN の Web サイトに locale 文字列に関する有用な情報を見つけることができるでしょう。 サポートしている言語文字列は、» http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_crt_language_strings.asp 、サポートしている国/地域文字列は» http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vclib/html/_crt_country_strings.asp にあります。Windows システムは、ISO 3166-Alpha-3 によって定められた国/地域文字列の 3 文字コードをサポートしています。 これは» Unicode website にあります。

sha1_file

(PHP 4 >= 4.3.0, PHP 5, PECL hash:1.1-1.3)

sha1_file — ファイルの sha1 ハッシュを計算する

説明

string **sha1_file** (string $filename [, bool $raw_output])

» US Secure Hash Algorithm 1 を使用して filename の sha1 ハッシュを計算し、そのハッシュを 返します。ハッシュは 40 文字の 16 進数となります。

パラメータ

filename

ファイル名。

raw_output

TRUE を指定すると、長さ 20 のバイナリフォーマットで ダイジェストを返します。デフォルトは **FALSE** です。

返り値

成功した場合に文字列、それ以外の場合に **FALSE** を返します。

変更履歴

バージョン	説明
5.0.0	raw_output パラメータが追加されました。
5.1.0	ストリーム API を使用するように変更しました。つまり、 sha1_file('http://example.com/..') のように ラッパとともに使用できるということです。

参考

- sha1()
- md5_file()
- crc32()

---

sha1

(PHP 4 >= 4.3.0, PHP 5, PECL hash:1.1-1.3)

sha1 — 文字列の sha1 ハッシュを計算する

説明

string **sha1** (string $str [, bool $raw_output])

» US Secure Hash Algorithm 1 を使用して str の sha1 ハッシュを計算します。

パラメータ

str

入力文字列。

raw_output

オプションの raw_output に **TRUE** が指定された場合、sha1 ダイジェストは 20 バイト長のバイナリ形式で返されます。 それ以外の場合は、返り値は 40 文字の 16 進数となります。 デフォルトは **FALSE** です。

返り値

sha1 ハッシュを文字列で返します。

変更履歴

バージョン	説明
5.0.0	raw_output パラメータが追加されました。

例

Example#1 sha1() の例

```
<?php
```

```
$str = 'apple';

if (sha1($str) === 'd0be2dc421be4fcd0172e5afceea3970e2f3d940') {
     echo "Would you like a green or red apple?";
     exit;
}
?>
```

参考

- sha1_file()
- crc32()
- md5()

---

similar_text

(PHP 4, PHP 5)

similar_text — 二つの文字列の間の類似性を計算する

説明

int **similar_text** (string $first , string $second [, float &$percent])

この関数は、Oliver [1993] に記述されたように二つの文字列の間の類似性を計算します。 この実装は Oliver の擬似コードの様にスタックを使用せず、 プロセス全体の速度が改善されるかどうかにかかわらず再帰呼び出しを行うことに注意してください。 このアルゴリズムの複雑さは、O(N**3) であることにも注意してください。 ただし、N は最も長い文字列の長さです。

パラメータ

first

最初の文字列。

second

次の文字列。

percent

3 番目の引数としてリファレンスを渡すことにより、 **similar_text()** は類似性をパーセントで計算します。

返り値

両方の文字列でマッチした文字の数を返します。

参考

- levenshtein()
- soundex()

---

soundex

(PHP 4, PHP 5)

soundex — 文字列の soundex キーを計算する

説明

string **soundex** (string $str)

str の soundex キーを計算します。

soundex キーには、似たような発音の単語に関して同じ soundex キーが生成されるという特性があります。 このため、発音は知っているが、スペルがわからない場合に、 データベースを検索することを容易にすることができます。 soundex 関数は、ある文字から始まる 4 文字の文字列を返します。

この soundex 関数についての説明は、Donald Knuth の "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392 にあります。

パラメータ

str

入力文字列。

返り値

soundex キーを文字列で返します。

例

Example#1 Soundex の例

```
<?php
soundex("Euler")        == soundex("Ellery");     // E460
soundex("Gauss")        == soundex("Ghosh");      // G200
soundex("Hilbert")      == soundex("Heilbronn"); // H416
soundex("Knuth")        == soundex("Kant");       // K530
soundex("Lloyd")        == soundex("Ladd");       // L300
soundex("Lukasiewicz") == soundex("Lissajous"); // L222
?>
```

参考

- levenshtein()
- metaphone()
- similar_text()

---

sprintf

(PHP 4, PHP 5)

sprintf — フォーマットされた文字列を返す

説明

string **sprintf** (string $format [, mixed $args [, mixed $...]])

フォーマット文字列 format に基づき生成された文字列を返します。

パラメータ

format

フォーマット文字列は 0 個以上のディレクティブ（指示子） により構成されます。ディレクティブには、そのまま結果にコピーされる (% を除く) 通常の文字と変換指定子 (conversion specifications) があり、 取り出される際はどちらもそれ自身がパラメータとなります。このことは **sprintf()** の場合だけでなく printf() の場合も同様です。

各変換指定子は、パーセント記号 (%) の後に これらの要素が一つ以上続いたものになります。

1. オプションの符号指定子。これは、 数値で符号 (- あるいは +) を使用するよう指定します。 デフォルトでは、数値が負の場合の - 符号のみが使用されます。 この指定子により、正の数にも強制的に + 符号をつけることができます。 これは PHP 4.3.0 で追加されました。
2. オプションのパディング指定子。これは、 文字列が正しい長さになるまでどんな文字で埋めるかということを 指定します。これは空白かまたは 0 (文字 '0') のいずれかです。デフォルトでは空白で埋められます。 これ以外のパディング文字を指定するには、その文字の前に 単一引用符 (') を置きます。 後述の例を参照ください。
3. オプションのアラインメント指定子。これは、 結果を左寄せまたは右寄せにしたい場合に指定します。 デフォルトは右寄せです。ここで - 文字を指定すると左寄せとなります。
4. オプションの数字。これは表示幅指定子です。 結果を (最低) 何桁にするかを指定します。
5. オプションの精度指定子。これは、 浮動小数点数に対して数字を何桁まで表示するかを指定します。 文字列に対して使用した場合は、これは切り捨て位置として働きます。 この文字数を超える文字を切り捨てられます。

6. 型指定子。引数を何の型として扱うかを指定します。 指定できる型を以下に示します。

    - % - パーセント文字。引数は不要です。
    - b - 引数を整数として扱い、 バイナリの数値として表現します。
    - c - 引数を整数として扱い、その ASCII 値の文字として表現します。
    - d - 引数を整数として扱い、 10 進数として表現します。
    - e - 引数を科学記法として扱います (例 1.2e+2)。 精度の指定子は、PHP 5.2.1 以降では小数点以下の桁数を表します。 それより前のバージョンでは、有効数字の桁数 (ひとつ小さい値) を意味していました。
    - u - 引数を整数として扱い、符号無しの 10 進数として表現します。
    - f - 引数を double として扱い、 浮動小数点数として表現します。
    - F - 引数を float として扱い、 浮動小数点数として表現します (ロケールに依存しません)。 PHP 4.3.10 および PHP 5.0.3 以降で使用可能です。
    - o - 引数を整数として扱い、 8 進数として表現します。
    - s - 引数を文字列として扱い、表現します。
    - x - 引数を整数として扱い、16 進数として (小文字で)表現します。
    - X - 引数を整数として扱い、16 進数として (大文字で)表現します。

フォーマット文字列における引数の 番号付け/交換 をサポートしています。以下に例を示します。

Example#1 引数の交換

```
<?php
$format = 'There are %d monkeys in the %s';
printf($format, $num, $location);
?>
```

この出力は、"There are 5 monkeys in the tree" のようになります。 ここで、フォーマット文字列が別のファイルにある場合を考えてみましょう。 これは、出力を国際化したりする場合に行われる可能性があります。 たとえばフォーマット文字列が次のように書き換えられたとすると、

Example#2 引数の交換

```
<?php
$format = 'The %s contains %d monkeys';
printf($format, $num, $location);
?>
```

ここで、問題が発生します。フォーマット文字列における置換指示子の順番は、 コードにおける引数の順番と一致していません。 だからといってコードを変更するのではなく、 むしろ置換指示子が参照するフォーマット文字列のほうで指示を行う方が望ましいでしょう。 フォーマット文字列を次のように書き換えてみましょう。

Example#3 引数の交換

```
<?php
$format = 'The %2$s contains %1$d monkeys';
printf($format, $num, $location);
?>
```

こうすることによるもうひとつの利点は、 同じ置換指示子を複数回使用する際にコードに引数を追加せずにすむことです。 例えば、次のようになりま

す。

Example#4 引数の交換

```
<?php
$format = 'The %2$s contains %1$d monkeys.
           That¥'s a nice %2$s full of %1$d monkeys.';
printf($format, $num, $location);
?>
```

args

...

返り値

フォーマット文字列 format に基づき生成された文字列を返します。

変更履歴

バージョン	説明
4.0.6	引数の 番号付け/交換 をサポートするようになりました。

例

Example#5 printf() のさまざまな例

```
<?php
$n =  43951789;
$u = -43951789;
$c = 65; // ASCII コードの 65 は 'A' です

// %% に注目しましょう。これは、リテラル '%' を文字として出力します
printf("%%b = '%b'¥n", $n); // 2 進表現
printf("%%c = '%c'¥n", $c); // ascii 文字を表示します。chr() 関数と同じです
printf("%%d = '%d'¥n", $n); // 標準の整数表現
printf("%%e = '%e'¥n", $n); // 科学記法
printf("%%u = '%u'¥n", $n); // 正の整数の、符号なし整数表現
printf("%%u = '%u'¥n", $u); // 負の整数の、符号なし整数表現
printf("%%f = '%f'¥n", $n); // 浮動小数点表現
printf("%%o = '%o'¥n", $n); // 8 進表現
printf("%%s = '%s'¥n", $n); // 文字列表現
printf("%%x = '%x'¥n", $n); // 16 進表現 (小文字)
printf("%%X = '%X'¥n", $n); // 16 進表現 (大文字)

printf("%%+d = '%+d'¥n", $n); // 正の整数に符号指定子を使用
printf("%%+d = '%+d'¥n", $u); // 負の整数に符号指定子を使用
?>
```

上の例の出力は以下となります。

```
%b  =  '10100111101010011010101101'
%c  =  'A'
%d  =  '43951789'
%e  =  '4.39518e+7'
%u  =  '43951789'
%u  =  '4251015507'
%f  =  '43951789.000000'
%o  =  '247523255'
%s  =  '43951789'
%x  =  '29ea6ad'
%X  =  '29EA6AD'
%+d  =  '+43951789'
%+d  =  '-43951789'
```

Example#6 printf() の文字列指定子

```
<?php
$s = 'monkey';
$t = 'many  monkeys';

printf("[%s]¥n",      $s); // 標準の文字列出力
printf("[%10s]¥n",    $s); // 空白を使用して右詰め
printf("[%-10s]¥n",   $s); // 空白を使用して左詰め
printf("[%010s]¥n",   $s); // ゼロ埋めは文字列でも可能です
printf("[%'#10s]¥n",  $s); // ゼロの代わりに独自の文字 '#' で埋めます
printf("[%10.10s]¥n", $t); // 左詰めを行い、10 文字以上は切り捨てます
?>
```

上の例の出力は以下となります。

```
[monkey]
[     monkey]
[monkey     ]
[0000monkey]
[####monkey]
[many  monke]
```

Example#7 sprintf(): 整数のゼロ埋め

```php
<?php
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
?>
```

Example#8 sprintf(): 通貨をフォーマットする例

```php
<?php
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money は "123.1" を出力します。
$formatted = sprintf("%01.2f", $money);
// echo $formatted は "123.10"を出力します
?>
```

Example#9 sprintf(): 科学記法

```php
<?php
$number = 362525200;

echo sprintf("%.3e", $number); // 3.625e+8 を出力します
?>
```

参考

- printf()
- sscanf()
- fscanf()
- vsprintf()
- number_format()

---

sscanf

(PHP 4 >= 4.0.1, PHP 5)

sscanf — フォーマット文字列に基づき入力を処理する

説明

mixed **sscanf** (string $str , string $format [, mixed &$...])

関数 **sscanf()** は、printf() の入力版です。**sscanf()** は、文字列 str を読み込み、これを指定したフォーマット format に基づき解釈します。 このフォーマットは、sprintf()のマニュアルに記述されています。

フォーマット文字列の中のあらゆる空白文字は、入力文字列の中の 空白文字列にマッチします。つまり、フォーマット文字列の中にタブ文字 \t が含まれていても、それは入力中の半角スペースにマッチしてしまうということです。

パラメータ

str

入力文字列。

format

str を解釈するフォーマット。 sprintf() のドキュメントを参照ください。

...

オプションで指定する参照渡しの変数に、 パースされた値が格納されます。

返り値

この関数のパラメータが二つだけの場合、処理された値は配列として返されます。 それ以外の場合は、もしオプションのパラメータが渡されればこの関数は 割り当てられた値の数を返します。オプションのパラメータは 参照渡しにする必要があります。

例

Example#1 sscanf() の例

```php
<?php
// シリアル番号を得る
list($serial) = sscanf("SN/2350001", "SN/%d");
// 続いて製造日を得る
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate, "%s %d %d");
echo "Item $serial was manufactured on: $year-" . substr($month, 0, 3) . "-$day¥n";
?>
```

オプションのパラメータが指定された場合、この関数は、代入された値の数を返します。

Example#2 sscanf() - オプションパラメータの使用法

```php
<?php
// author 情報を取得し、DocBook エントリを生成
$auth = "24¥tLewis Carroll";
$n = sscanf($auth, "%d¥t%s %s", $id, $first, $last);
echo "<author id='$id'>
       <firstname>$first</firstname>
       <surname>$last</surname>
```

```
</author>¥n";
?>
```

参考

- fscanf()
- printf()
- sprintf()

---

str_getcsv

(No version information available, might be only in CVS)

str_getcsv — CSV 文字列をパースして配列に格納する

説明

array **str_getcsv** (string $input [, string $delimiter [, string $enclosure [, string $escape]]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

参考

- fgetcsv()

---

str_ireplace

(PHP 5)

str_ireplace — 大文字小文字を区別しない str_replace()

説明

mixed **str_ireplace** (mixed $search , mixed $replace , mixed $subject [, int &$count])

この関数は、subject の中に現れるすべての search (大文字小文字を区別しない)を replace に置き換えた文字列あるいは配列を 返します。一般に、凝った置換ルールが必要ないのであれば、 eregi_replace() あるいは preg_replace() で i 修正子を 使用するかわりにこの関数を使用すべきです。

パラメータ

search

> **注意**: すべての search 配列による置換は、 直前の置換の結果に対して作用します。

replace

subject

subject が配列の場合は、そのすべての要素に 対して検索と置換が行われ、返される結果も配列となります。

count

needles の中で、マッチして置換を行った数を count に返します。このパラメータは参照渡しとします。

search および replace が配列の場合は、 **str_ireplace()** はそれぞれの配列から取り出した 値を使用して subject の置換を行います。replace の要素数が search より少ない場合は、残りの要素は 空の文字列に置き換えられます。もし search が配列で replace が文字列だった場合は すべての search が同じ文字列に置き換えられます。

返り値

置換した文字列あるいは配列を返します。

変更履歴

バージョン	説明
5.0.0	count パラメータが追加されました。

例

Example#1 str_ireplace() の例

```
<?php
$bodytag = str_ireplace("%body%", "black", "<body text=%BODY%>");
?>
```

注意

> **注意**: この関数はバイナリセーフです。

参考

- str_replace()
- preg_replace()
- strtr()

---

str_pad

(PHP 4 >= 4.0.1, PHP 5)

str_pad — 文字列を固定長の他の文字列で埋める

説明

string **str_pad** (string $input , int $pad_length [, string $pad_string [, int $pad_type]])

この関数は文字列 input の左、右または両側を指定した長さで埋めます。オプションの引数 pad_string が指定されていない場合は、 input は空白で埋められ、それ以外の場合は、 pad_string からの文字で制限まで埋められます。

パラメータ

input

入力文字列。

pad_length

pad_length の値が負、 または入力文字列の長さよりも短い場合、埋める操作は行われません。

pad_string

注意: 必要とされる埋める文字数が pad_string の長さで均等に分割できない場合、pad_string は切り捨てられます。

pad_type

オプションの引数 pad_type には、 **STR_PAD_RIGHT**, **STR_PAD_LEFT**, **STR_PAD_BOTH** を指定可能です。 pad_type が指定されない場合、**STR_PAD_RIGHT** を仮定します。

返り値

埋めた後の文字列を返します。

例

Example#1 str_pad() の例

```
<?php
$input = "Alien";
echo str_pad($input, 10);                      // 結果は "Alien      "
echo str_pad($input, 10, "-=", STR_PAD_LEFT);  // 結果は "-=-=-Alien"
echo str_pad($input, 10, "_", STR_PAD_BOTH);   // 結果は "__Alien___"
echo str_pad($input, 6 , "___");               // 結果は "Alien_"
?>
```

---

str_repeat

(PHP 4, PHP 5)

str_repeat — 文字列を反復する

説明

string **str_repeat** (string $input , int $multiplier)

input を multiplier 回を繰り返した文字列を返します。

パラメータ

input

繰り返す文字列。

multiplier

input を繰り返す回数。

multiplier は 0 以上でなければなりません。 multiplier が 0 に設定された場合、この関数は空文字を返します。

返り値

繰り返した文字列を返します。

返り値

Example#1 str_repeat() の例

```
<?php
echo str_repeat("-=", 10);
?>
```

上の例の出力は以下となります。

```
-=-=-=-=-=-=-=-=-=-=
```

参考

- for
- str_pad()
- substr_count()

---

str_replace

(PHP 4, PHP 5)

str_replace — 検索文字列に一致したすべての文字列を置換する

説明

```
mixed str_replace ( mixed $search , mixed $replace , mixed $subject [, int &$count ] )
```

この関数は、subject の中の search を全て replace に置換します。

(正規表現のような) 技巧的な置換ルールを必要としない場合、 ereg_replace() または preg_replace() の代わりにこの関数を常用するべきです。

パラメータ

search と replace が配列の場合、**str_replace()** は各配列から値をひとつ取り出し、 subject 上で検索と置換を行うために使用します。 replace の値が search よりも少ない場合、 置換される値の残りの部分には空の文字列が使用されます。 search が配列で replace が文字列の場合、この置換文字列が search の各値について使用されます。しかし、 逆は意味がありません。

search あるいは replace が配列の場合は、配列の最初の要素から順に処理されます。

search

replace

subject

subject が配列の場合、 subject の各エントリについて検索と置換が行われ、 返り値は同様に配列となります。

count

注意: 指定した場合は、マッチして置換が行われた箇所の個数がここに格納されます。

返り値

この関数は、置換後の文字列あるいは配列を返します。

変更履歴

バージョン	説明
5.0.0	count パラメータが追加されました。
4.3.3	この関数の挙動がかわりました。以前のバージョンにはバグがあり、 search と replace の両方に配列を指定した場合に、 空の search 添字をスキップしても replace 配列上の内部ポインタが進みませんでした。 このバグは PHP 4.3.3 で修正されたので、このバグを前提としたスクリプトは、 この関数をコールする前に空の検索値を削除しておく必要があります。
4.0.5	ほとんどのパラメータに配列を渡せるようになりました。

例

Example#1 str_replace() の例

```
<?php
// <body text='black'> となります
$bodytag = str_replace("%body%", "black", "<body text='%body%'>");

// Hll Wrld f PHP となります
$vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
$onlyconsonants = str_replace($vowels, "", "Hello World of PHP");

// You should eat pizza, beer, and ice cream every day となります
$phrase  = "You should eat fruits, vegetables, and fiber every day.";
$healthy = array("fruits", "vegetables", "fiber");
$yummy   = array("pizza", "beer", "ice cream");
```

```
$newphrase  = str_replace($healthy, $yummy, $phrase);

// パラメータ count は PHP 5.0.0 で使用可能になりました
$str = str_replace("ll", "", "good golly miss molly!", $count);
echo $count; // 2

// 置換の順番を指定します
$str      = "Line 1¥nLine 2¥rLine 3¥r¥nLine 4¥n";
$order    = array("¥r¥n", "¥n", "¥r");
$replace  = '<br />';
// まず \r\n を最初に置換するので、二重に変換されることはありません
$newstr = str_replace($order, $replace, $str);

// 出力は apearpearle pear となります
$letters  = array('a', 'p');
$fruit    = array('apple', 'pear');
$text     = 'a p';
$output   = str_replace($letters, $fruit, $text);
echo $output;
?>
```

注意

注意: この関数はバイナリデータに対応しています。

注意: この関数は大文字小文字を区別します。区別せずに置換するには str_ireplace() を使用します。

参考

- str_ireplace()
- substr_replace()
- preg_replace()
- strtr()

---

str_rot13

(PHP 4 >= 4.2.0, PHP 5)

str_rot13 — 文字列に rot13 変換を行う

説明

string **str_rot13** (string $str)

Performs the ROT13 encoding on the str argument and returns the resulting string.

ROT13 は、各文字をアルファベット順に 13 文字シフトさせ、 アルファベット以外の文字はそのままとするエンコードを行います。 エンコードとデコードは同じ関数で行われます。 引数にエンコードされた文字列を指定した場合には、元の文字列が返されます。

パラメータ

str

入力文字列。

返り値

指定した文字列を ROT13 変換した結果を返します。

例

Example#1 str_rot13() の例

```
<?php

echo str_rot13('PHP 4.3.0'); // CUC 4.3.0

?>
```

変更履歴

バージョン	説明
4.3.0	この関数の挙動が修正されました。以前のバージョンでは、 str 自体も変更されてしまっていました。 ちょうど、参照渡しで渡したときと同じような挙動だったのです。

---

str_shuffle

(PHP 4 >= 4.3.0, PHP 5)

str_shuffle — 文字列をランダムにシャッフルする

説明

string **str_shuffle** (string $str)

str_shuffle() は文字列をシャッフルします。 考えられるすべての順列のうちのひとつを作成します。

パラメータ

str

入力文字列。

返り値

Returns the shuffled string.

例

Example#1 str_shuffle() の例

```
<?php
$str = 'abcdef';
$shuffled = str_shuffle($str);

// bfdaec のような文字列を返します
echo $shuffled;
?>
```

参考

- shuffle()
- rand()

---

str_split

(PHP 5)

str_split — 文字列を配列に変換する

説明

array **str_split** (string $string [, int $split_length])

文字列を配列に変換します。

パラメータ

string

入力文字列。

split_length

分割した部分の最大長。

返り値

オプションのパラメータ split_length が指定されている場合、 返される配列の各要素は、split_length の長さとなります。それ以外の場合、1 文字ずつ分割された配列となります。

split_length が 1 より小さい場合に **FALSE** を返します。split_length が string の長さより大きい場合、文字列全体が 最初の(そして唯一の)要素となる配列を返します。

例

Example#1 str_split() の使用例

```
<?php

$str = "Hello Friend";

$arr1 = str_split($str);
$arr2 = str_split($str, 3);

print_r($arr1);
print_r($arr2);

?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => H
    [1] => e
    [2] => l
    [3] => l
    [4] => o
    [5] =>
```

```
    [6] => F
    [7] => r
    [8] => i
    [9] => e
    [10] => n
    [11] => d
)

Array
(
    [0] => Hel
    [1] => lo
    [2] => Fri
    [3] => end
)
```

参考

- chunk_split()
- preg_split()
- explode()
- count_chars()
- str_word_count()
- for

---

str_word_count

(PHP 4 >= 4.3.0, PHP 5)

str_word_count — 文字列に使用されている単語についての情報を返す

説明

mixed **str_word_count** (string $string [, int $format [, string $charlist]])

string の単語数を数えます。 オプションの format が指定されていない場合、 見つかった単語の数を整数値で返します。 format が指定されている場合は結果が配列で返され、 配列の内容は format に依存します。 format に設定できる値と対応する出力については 以下で示します。

この関数を使用するうえで、'単語' は以下のように定義されます。すなわち、 「ロケールに依存しないアルファベットからなる文字列で、その先頭以外の部分に "'" および "-" が含まれていてもよい」。

パラメータ

string

文字列。

format

この関数の戻り値を設定します。現在サポートされている値は 以下のとおりです。

- 0 - 見つかった単語の数を返します。
- 1 - string の中に見つかった単語を含む 配列を返します。
- 2 - 連想配列を返します。string の中での 単語の開始位置がキー、単語自体を対応する値となります。

charlist

'単語' とみなされる文字に追加する文字のリスト。

返り値

選択した format に応じて、配列あるいは整数を返します。

変更履歴

バージョン	説明
5.1.0	charlist パラメータが追加されました。

例

Example#1 str_word_count() の例

```
<?php

$str = "Hello  fri3nd,  you're
        looking           good today!";

print_r(str_word_count($str, 1));
print_r(str_word_count($str, 2));
print_r(str_word_count($str, 1, 'àáãç3'));

echo str_word_count($str);

?>
```

上の例の出力は以下となります。

```
Array
(
     [0] => Hello
     [1] => fri
     [2] => nd
     [3] => you're
     [4] => looking
     [5] => good
     [6] => today
)

Array
(
     [0] => Hello
     [6] => fri
     [10] => nd
     [14] => you're
     [29] => looking
     [46] => good
     [51] => today
)

Array
(
     [0] => Hello
     [1] => fri3nd
     [2] => you're
     [3] => looking
     [4] => good
     [5] => today
)

7
```

参考

- explode()
- preg_split()
- split()
- count_chars()
- substr_count()

---

strcasecmp

(PHP 4, PHP 5)

strcasecmp — 大文字小文字を区別しないバイナリセーフな文字列比較を行う

説明

int **strcasecmp** (string $str1 , string $str2)

大文字小文字を区別しないバイナリセーフな文字列比較を行います。

パラメータ

str1

最初の文字列。

str2

次の文字列。

返り値

str1 が str2 より小さい場合は負、str1 が str2 より大きい場合は正、等しい場合は 0 を返します。

例

Example#1 strcasecmp() の例

```
<?php
$var1 = "Hello";
$var2 = "hello";
if (strcasecmp($var1, $var2) == 0) {
     echo '$var1 is equal to $var2 in a case-insensitive string comparison';
}
?>
```

参考

- preg_match()

- strcmp()
- substr()
- stristr()
- strncasecmp()
- strstr()

---

strchr

(PHP 4, PHP 5)

strchr — strstr() のエイリアス

説明

この関数は次の関数のエイリアスです。 strstr().

---

strcmp

(PHP 4, PHP 5)

strcmp — バイナリセーフな文字列比較

説明

int **strcmp** (string $str1 , string $str2)

この比較は大文字小文字を区別することに注意してください。

パラメータ

str1

最初の文字列。

str2

次の文字列。

返り値

str1 が str2 よりも小さければ < 0 を、str1 が str2 よりも大きければ > 0 を、 等しければ 0 を返します。

参考

- preg_match()
- strcasecmp()
- substr()
- stristr()
- strncasecmp()
- strncmp()
- strstr()

---

strcoll

(PHP 4 >= 4.0.5, PHP 5)

strcoll — ロケールに基づく文字列比較

説明

int **strcoll** (string $str1 , string $str2)

この比較は大文字小文字を区別すること、そして、 strcmp() とは異なり、 バイナリセーフではないことに注意してください。

strcoll() は比較を行う際に現在のロケールを使用します。 ロケールが C または POSIX の場合、この関数は strcmp() と等価です。

パラメータ

str1

最初の文字列。

str2

次の文字列。

返り値

str1 が str2 より小さい場合に 0 未満の値、str1 が str2 より大きい場合に 0 より大きな値、 両者が等しい場合に 0 を返します。

変更履歴

バージョン	説明
4.2.3	この関数が win32 でも動作するようになりました。

参考

- preg_match()
- strcmp()
- strcasecmp()
- substr()
- stristr()
- strncasecmp()
- strncmp()
- strstr()
- setlocale()

---

strcspn

(PHP 4, PHP 5)

strcspn — マスクにマッチしない最初のセグメントの長さを返す

説明

*int **strcspn** (string $str1 , string $str2 [, int $start [, int $length]])*

str1 において str2 の文字がどれも含まれていない最初のセグメントの長さを返します。

パラメータ

str1

str2

start

調べ始める位置。

length

調べる文字列の長さ。

返り値

セグメントの長さを返します。

変更履歴

バージョン	説明
4.3.0	*start* および *length* が追加されました。

注意

> ***注意**: この関数はバイナリデータに対応しています。*

参考

- strspn()

---

strip_tags

(PHP 4, PHP 5)

strip_tags — 文字列から HTML および PHP タグを取り除く

説明

*string **strip_tags** (string $str [, string $allowable_tags])*

この関数は、指定した文字列 (str) から全ての HTML および PHP タグを取り除きます。 この関数は、fgetss() 関数と同じタグ除去アルゴリズムを使用します。

パラメータ

str

入力文字列。

allowable_tags

オプションの2番目の引数により、取り除かないタグを指定できます。

> **注意**: HTML コメントや PHP タグも削除されるようになりました。この機能はハードコードされており、 allowable_tags で変更することはできません。

返り値

タグを除去した文字列を返します。

変更履歴

バージョン	説明
5.0.0	strip_tags() がバイナリセーフとなりました。
4.3.0	HTML のコメントも除去するようになりました。
4.0.0	allowable_tags パラメータが追加されました。

例

Example#1 strip_tags() の例

```
<?php
$text = '<p>Test paragraph.</p><!-- Comment --> <a href="#fragment">Other text</a>';
echo strip_tags($text);
echo "¥n";

// <p> と <a> は許可します
echo strip_tags($text, '<p><a>');
?>
```

上の例の出力は以下となります。

```
Test paragraph. Other text
<p>Test paragraph.</p> <a href="#fragment">Other text</a>
```

注意

警告

strip_tags() は HTML の検証を行わないため、 不完全または壊れたタグにより予想以上に多くのテキスト/データが削除される可能性があります。

警告

この関数は、allowable_tags で許可した全てのタグの属性を修正しません。 これには、style および onmouseover属性が含まれており、 悪意のあるユーザが他のユーザに見せるようなテキストを投稿する際に危険な行為を行う可能性があります。

参考

- htmlspecialchars()

---

stripcslashes

(PHP 4, PHP 5)

stripcslashes — addcslashes() でクォートされた文字列をアンクォートする

説明

string **stripcslashes** (string $str)

バックスラッシュを取り除いた文字列を返します。C言語と同様に ¥n, ¥r ..., 8進表現, 16進表現を認識します。

パラメータ

str

元に戻したい文字列。

返り値

元に戻した文字列を返します。

参考

- addcslashes()

stripos

(PHP 5)

stripos — 大文字小文字を区別せずに文字列が最初に現れる位置を探す

説明

int **stripos** (string $haystack , string $needle [, int $offset])

文字列 haystack の中で needle が最初に現れる位置を数字で返します。

strpos() と異なり、**stripos()** は大文字小文字を区別しません。

パラメータ

haystack

検索を行う文字列。

needle

needle は、 ひとつまたは複数の文字であることに注意しましょう。

needle が文字列でない場合は、 それを整数に変換し、その番号に対応する文字として扱います。

offset

オプションのパラメータ offset により、検索を開始する haystack の位置を指定することができます。 この場合でも返される位置は、 haystack の先頭からの位置のままとなります。

返り値

needle がみつからない場合、 strpos() は boolean **FALSE** を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

例

Example#1 stripos() の例

```
<?php
$findme     = 'a';
$mystring1  = 'xyz';
$mystring2  = 'ABC';

$pos1 = stripos($mystring1, $findme);
$pos2 = stripos($mystring2, $findme);

// いいえ、'a' は明らかに 'xyz' の中には存在しません
if ($pos1 === false) {
    echo "The string '$findme' was not found in the string '$mystring1'";
}

// === を使用していることに注意しましょう。単に == としても期待通りに動作
// しません。なぜなら 'a' は 0 番目(最初) の文字だからです。
if ($pos2 !== false) {
    echo "We found '$findme' in '$mystring2' at position $pos2";
}
?>
```

注意

注意: この関数はバイナリデータに対応しています。

参考

- strpos()
- strrpos()
- strrchr()
- substr()
- stristr()
- strstr()
- strripos()
- str_ireplace()

stripslashes

(PHP 4, PHP 5)

stripslashes — addslashes() でクォートされた文字列のクォート部分を取り除く

説明

string **stripslashes** (string $str)

クォートされた文字列を元に戻します。

> **注意**: magic_quotes_sybase が on の場合はバックスラッシュは取り除かれず、そのかわりに 2 つの アポストロフィが 1 つに置き換えられます。

stripslashes() の使用例は、PHP ディレクティブ magic_quotes_gpc が on（デフォルトでオン）かつ、 データをエスケープして（データベースのような）ある場所に挿入していない場合です。 例えば、単純に HTML フォームからのデータを直接出力するような場合です。

パラメータ

str

入力文字列。

返り値

バックスラッシュが取り除かれた文字列を返します(¥' が ' になるなど)。 2 つ並んだバックスラッシュ (¥¥) は 1 つのバックスラッシュ (¥) になります。

例

Example#1 stripslashes() の例

```
<?php
$str = "Is your name O¥'reilly?";

// 出力: Is your name O'reilly?
echo stripslashes($str);
?>
```

> **注意**: **stripslashes()** は再帰的な処理を行いません。 この関数を多次元配列に適用する場合は、 再帰的な関数を使用する必要があります。

Example#2 配列に対する stripslashes() の使用

```
<?php
function stripslashes_deep($value)
{
    $value = is_array($value) ?
                array_map('stripslashes_deep', $value) :
                stripslashes($value);

    return $value;
}

// 例
$array = array("f¥¥'oo", "b¥¥'ar", array("fo¥¥'o", "b¥¥'ar"));
$array = stripslashes_deep($array);

// 出力
print_r($array);
?>
```

上の例の出力は以下となります。

```
Array
(
    [0] => f'oo
    [1] => b'ar
    [2] => Array
        (
            [0] => fo'o
            [1] => b'ar
        )

)
```

参考

- addslashes()
- get_magic_quotes_gpc()

---

stristr

(PHP 4, PHP 5)

stristr — 大文字小文字を区別しない strstr()

説明

string **stristr** (string $haystack , string $needle , bool $before_needle)

haystack において needle が最初に見つかった位置から最後までを返します。

パラメータ

haystack

検索を行う文字列。

needle

needle が文字列でない場合は、 それを整数に変換し、その番号に対応する文字として扱います。

before_needle

TRUE にすると（デフォルトは FALSE です）、stristr() の返り値は、haystack の中で最初に needle があらわれる箇所より前の部分となります。

needle および haystack は大文字小文字を区別せずに評価されます。

返り値

マッチした部分文字列を返します。needle が見つからない場合は FALSE を返します。

変更履歴

バージョン	説明
6.0.0	before_needle パラメータが追加されました。
4.3.0	stristr() がバイナリセーフとなりました。

例

Example#1 stristr() の例

```
<?php
   $email = 'USER@EXAMPLE.com';
   echo stristr($email, 'e'); // 出力は ER@EXAMPLE.com となります
   echo stristr($email, 'e', true); // 出力は US となります
?>
```

Example#2 文字列が見つかるかどうかをテストする

```
<?php
   $string = 'Hello World!';
   if(stristr($string, 'earth') === FALSE) {
      echo '"earth" not found in string';
   }
// 出力は "earth" not found in string となります
?>
```

Example#3 文字列以外の needle の指定

```
<?php
   $string = 'APPLE';
   echo stristr($string, 97); // 97 = 小文字の a
// 出力は APPLE となります
?>
```

注意

注意: この関数はバイナリデータに対応しています。

参考

- strstr()
- strrchr()
- substr()
- preg_match()

---

strlen

(PHP 4, PHP 5)

strlen — 文字列の長さを得る

説明

int **strlen** (string $string)

与えられた string の長さを返します。

パラメータ

string

長さを調べる文字列。

返り値

成功した場合に string の長さ、 string が空の文字列だった場合に 0 を返します。

例

Example#1 strlen() の例

```
<?php
$str = 'abcdef';
echo strlen($str); // 6

$str = ' ab cd ';
echo strlen($str); // 7
?>
```

参考

- count()
- mb_strlen()

---

strnatcasecmp

(PHP 4, PHP 5)

strnatcasecmp — "自然順"アルゴリズムにより大文字小文字を区別しない文字列比較を行う

説明

int **strnatcasecmp** (string $str1 , string $str2)

この関数は、人間が行うような手法でアルファベットまたは数字の 文字列の順序を比較するアルゴリズムを実装します。この関数の動作は、strnatcmp() に似ていますが、 比較が大文字小文字を区別しない違いがあります。 詳細な情報については、Martin Pool の » 自然順文字列比較 のページを参照ください。

パラメータ

str1

最初の文字列。

str2

次の文字列。

返り値

他の文字列比較関数と同様に、この関数は、 str1 がstr2 より小さい場合に < 0、str1 が str2 より大きい場合に > 0 、等しい場合に 0 を返します。

参考

- preg_match()
- strcmp()
- strcasecmp()
- substr()
- stristr()
- strncasecmp()
- strncmp()
- strstr()
- setlocale()

---

strnatcmp

(PHP 4, PHP 5)

strnatcmp — "自然順"アルゴリズムにより文字列比較を行う

説明

int **strnatcmp** (string $str1 , string $str2)

この関数は、人間が行うような手法でアルファベットまたは数字の 文字列の順序を比較するアルゴリズムを実装します。この手法は、"自然順" と言われます。 この比較は、大文字小文字を区別することに注意してください。

パラメータ

str1

最初の文字列。

str2

次の文字列。

返り値

他の文字列比較関数と同様に、この関数は、 str1 が str2 より小さいに場合に < 0、str1 が str2 より大きい場合に > 0 、等しい場合に 0 を返します。

例

このアルゴリズムと （strcmp() を使用した）通常のコンピュータ文字列ソートの間の違いの例を次に示します。

```
<?php
$arr1 = $arr2 = array("img12.png", "img10.png", "img2.png", "img1.png");
echo "標準の文字列比較\n";
usort($arr1, "strcmp");
print_r($arr1);
echo "\n自然順での文字列比較\n";
usort($arr2, "strnatcmp");
print_r($arr2);
?>
```

上の例の出力は以下となります。

```
標準の文字列比較
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

自然順での文字列比較
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

詳細な情報については、Martin Pool の » 自然順文字列比較 のページを参照ください。

参考

- preg_match()
- strcasecmp()
- substr()
- stristr()
- strcmp()
- strncmp()
- strncasecmp()
- strnatcasecmp()
- strstr()
- natsort()
- natcasesort()

---

strncasecmp

(PHP 4 >= 4.0.2, PHP 5)

strncasecmp — バイナリセーフで大文字小文字を区別しない文字列比較を、最初の n 文字について行う

説明

*int **strncasecmp** (string $str1 , string $str2 , int $len)*

この関数は、 strcasecmp() に似ていますが、 各文字列から比較する文字数(の上限)(len) を指定できるという違いがあります。どちらかの文字列が len より短い場合、その文字列の長さが比較時に使用されます。

パラメータ

str1

最初の文字列。

str2

次の文字列。

len

比較する文字列の長さ。

返り値

str1 が str2 より短い場合に < 0 を返し、str1 が str2 より大きい場合に > 0、等しい場合に 0 を返します。

参考

- preg_match()
- strcmp()
- strcasecmp()
- substr()
- stristr()
- strstr()

---

strncmp

(PHP 4, PHP 5)

strncmp — 最初の n 文字についてバイナリセーフな文字列比較を行う

説明

int **strncmp** (string $str1 , string $str2 , int $len)

この関数は strcmp() に似ていますが、 各文字列から(最大)文字数(len) を比較に使用するところが異なります。

比較は大文字小文字を区別することに注意してください。

パラメータ

str1

最初の文字列。

str2

次の文字列。

len

比較する文字数。

返り値

str1 が str2 より小さい場合に < 0、str1 が str2 より大きい場合に > 0 、等しい場合に 0 を返します。

参考

- preg_match()
- strcmp()
- strcasecmp()
- substr()
- stristr()
- strncasecmp()
- strstr()

---

strpbrk

(PHP 5)

strpbrk — 文字列の中から任意の文字を探す

説明

string **strpbrk** (string $haystack , string $char_list)

strpbrk() は、文字列 haystack から char_list を探します。

パラメータ

haystack

char_list を探す文字列。

char_list

このパラメータは大文字小文字を区別します。

返り値

見つかった文字から始まる文字列、あるいは見つからなかった場合に **FALSE** を返します。

例

Example#1 strpbrk() の例

```
<?php

$text = 'This is a Simple text.';

// これは "is is a Simple text." を出力します。なぜなら 'i' が最初にマッチするからです。
echo strpbrk($text, 'mi');

// これは "Simple text." を出力します。なぜなら大文字小文字が区別されるからです。
echo strpbrk($text, 'S');
?>
```

---

strpos

(PHP 4, PHP 5)

strpos — 文字列が最初に現れる場所を見つける

説明

int **strpos** (string $haystack , mixed $needle [, int $offset])

文字列 haystack の中で、 needle が最初に現れた位置を数字で返します。 PHP 5 以前の strrpos() とは異なり、この関数は needle パラメータとして文字列全体をとり、 その文字列全体が検索対象となります。

パラメータ

haystack

検索を行う文字列。

needle

needle が文字列でない場合は、 それを整数に変換し、その番号に対応する文字として扱います。

offset

オプションのパラメータ offset により、 検索を開始する haystack の文字を指定することができます。 この場合でも、返される位置は haystack の先頭からの相対位置となります。

返り値

位置を表す整数値を返します。 needle が見つからない場合、 **strpos()** は boolean **FALSE** を返します。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

例

Example#1 strpos() の例

```
<?php
$mystring = 'abc';
$findme   = 'a';
$pos = strpos($mystring, $findme);

// === を使用していることに注目しましょう。単純に == を使ったのでは
// 期待通りに動作しません。なぜなら 'a' が 0 番目（最初）の文字だからです。
if ($pos === false) {
    echo "文字列 '$findme' は、文字列 '$mystring' の中で見つかりませんでした";
} else {
    echo "文字列 '$findme' が文字列 '$mystring' の中で見つかりました";
    echo " 見つかった位置は $pos です";
}

// オフセット以前の内容を無視して文字を探すこともできます。
$newstring = 'abcdef abcdef';
$pos = strpos($newstring, 'a', 1); // $pos は 0 ではなく 7 となります。
?>
```

注意

> **注意**: この関数はバイナリデータに対応しています。

参考

- strrpos()
- stripos()
- strripos()
- strrchr()

- substr()
- stristr()
- strstr()

---

strrchr

(PHP 4, PHP 5)

strrchr — 文字列中に文字が最後に現れる場所を取得する

説明

string **strrchr** (string $haystack , string $needle)

この関数は、文字列 haystack の中で needle が最後に現れた位置から、 haystack の終わりまでを返します。

パラメータ

haystack

検索を行う文字列。

needle

needle がひとつ以上の文字を含んでいる場合は、 最初のもののみが使われます。この動作は、 strchr() とは異なります。

needle が文字列でない場合は、 それを整数に変換し、その番号に対応する文字として扱います。

返り値

この関数は、部分文字列を返します。 needle が見つからない場合は **FALSE** を返します。

変更履歴

バージョン	説明
4.3.0	この関数はバイナリセーフとなりました。

例

Example#1 strrchr() の例

```php
<?php
// $PATH 中の最後のディレクトリを取得
$dir = substr(strrchr($PATH, ":"), 1);

// 最後の改行文字以降をすべて取得
$text = "Line 1¥nLine 2¥nLine 3";
$last = substr(strrchr($text, 10), 1 );
?>
```

注意

注意: この関数はバイナリデータに対応しています。

参考

- strstr()
- substr()
- stristr()

---

strrev

(PHP 4, PHP 5)

strrev — 文字列を逆順にする

説明

string **strrev** (string $string)

string を逆順にして返します。

パラメータ

string

逆順にしたい文字列。

返り値

逆順にした文字列を返します。

例

Example#1 strrev() で文字列を逆順にする

```
<?php
echo strrev("Hello world!"); // 出力は "!dlrow olleH" となります
?>
```

---

strripos

(PHP 5)

strripos — 文字列中で、特定の(大文字小文字を区別しない)文字列が最後に現れた位置を探す

説明

int **strripos** (string $haystack , string $needle [, int $offset])

文字列の中で、 大文字小文字を区別しないある文字列が最後に現れた位置を返します。 strrpos() と異なり、**strripos()** は大文字小文字を区別しません。

パラメータ

haystack

検索を行う文字列。

needle

needle は 1 文字あるいは複数の文字からなる 文字列であることに注意してください。

offset

パラメータ offset を指定すると、 文字列中の任意の位置から検索を始めることができます。

負の offset 値を指定すると、文字列の最初 から数えて offset 文字目から検索を始めます。

返り値

needle が最後に現れた位置を返します。 文字列の位置は 0 から始まるのであって、 1 からではないことに注意してください。

needle が見つからない場合、**FALSE** が返されます。

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

例

Example#1 単純な strripos() の例

```
<?php
$haystack  = 'ababcd';
$needle    = 'aB';

$pos       = strripos($haystack, $needle);

if ($pos === false) {
    echo "ごめんなさい、($needle) が ($haystack) の中に見つかりませんでした。";
} else {
    echo "おめでとう!\n";
    echo "($needle) が最後に ($haystack) に現れた位置は ($pos) です。";
}
?>
```

上の例の出力は以下となります。

```
おめでとう!
(aB) が最後に (ababcd) に現れた位置は (2) です。
```

参考

- strpos()
- stripos()
- strrchr()
- substr()
- stristr()
- strstr()

---

strrpos

(PHP 4, PHP 5)

strrpos — 文字列中に、ある文字が最後に現れる場所を探す

説明

int **strrpos** (string $haystack , string $needle [, int $offset])

文字列 haystack の中で、 needle が最後に現れた位置を数字で返します。 この場合、needle は単一文字でなければならないことに 注意してください。needle に文字列が指定された場合、その文字列の最初の文字だけが使われます。

needle が見つからない場合、**FALSE** を返します。

"位置 0 に文字が見つかった" と "文字が見つからなかった" 場合の返り値は混同しやすいです。この違いを見分ける方法を以下に示します。

```
<?php

// PHP 4.0.0 以降の場合:
$pos = strrpos($mystring, "b");
if ($pos === false) { // 注意: 等号が 3 つ
    // 見つからない...
}

// 4.0.0 より古いバージョンの場合:
$pos = strrpos($mystring, "b");
if (is_bool($pos) && !$pos) {
    // 見つからない...
}
?>
```

needle が文字列でない場合は数値に変換されて、 その結果が検索対象の文字として適用されます。

> **注意**: PHP 5.0.0 以降、 offset により文字列中の任意の文字位置から検索を開始することができます。 負の値を指定すると、文字の終端より前の任意の位置で検索を終了します。
>
> **注意**: PHP 5.0.0 以降、needle は 1 文字以上の文字列を指定可能です。

パラメータ

haystack

needle

offset

返り値

参考

- strpos()
- strripos()
- strrchr()
- substr()
- stristr()
- strstr()

---

strspn

(PHP 4, PHP 5)

strspn — マスクに一致する最初のセグメントの長さを返す

説明

int **strspn** (string $str1 , string $str2 [, int $start [, int $length]])

マスクに一致する最初のセグメントの長さを求めます。

たとえば、

```
<?php
$var = strspn("42 is the answer, what is the question ...", "1234567890");
?>
```

このコードは、$var に 2 を代入します。これは、 "1234567890" からの文字を含む最長の部分が "42" であるためです。

パラメータ

str1

最初の文字列。

str2

次の文字列。

start

調べ始める位置。 負の値を指定すると、文字列の末尾から数えた位置となります。

length

調べる文字列の長さ。 負の値を指定すると、文字列の末尾からの長さとなります。

返り値

str1 の中で、全て str2 の中の文字からなる最初のセグメントの長さを返します。

変更履歴

バージョン	説明
4.3.0	*start* および *length* が追加されました。

例

Example#1 strspn() の例

```
<?php
echo strspn("foo", "o", 1, 2); // 2
?>
```

注意

> ***注意**: この関数はバイナリデータに対応しています。*

参考

- *strcspn()*

---

strstr

(PHP 4, PHP 5)

strstr — 文字列が最初に現れる位置を見つける

説明

*string **strstr** (string $haystack , string $needle , bool $before_needle)*

haystack の中で needle が最初に現れる場所から文字列の終わりまでを返します。

> ***注意**: この関数は大文字小文字を区別することに注意してください。 大文字小文字を区別しない検索を行う場合は、stristr() を使用してください。*
>
> ***注意**: もし特定の haystack に needle があるかどうかを調べるだけの場合、 より高速でメモリ消費も少ない strpos() を代わりに使用してください。*

パラメータ

haystack

入力文字列。

needle

needle が文字列でない場合は、 それを整数に変換し、その番号に対応する文字として扱います。

before_needle

***TRUE** にすると (デフォルトは **FALSE** です)、**strstr()** の返り値は、haystack の中で最初に needle があらわれる箇所より前の部分となります。*

返り値

*部分文字列を返します。 needle が見つからない場合は **FALSE** を返します。*

変更履歴

バージョン	説明
6.0.0	*before_needle* パラメータが追加されました。
4.3.0	strstr() がバイナリセーフとなりました。

例

Example#1 strstr() の例

```
<?php
$email  = 'name@example.com';
$domain = strstr($email, '@');
echo $domain; // @example.com と表示します
```

```
$user = strstr($email, '@', true);
echo $user; // name と表示します
?>
```

参考

- preg_match()
- stristr()
- strpos()
- strrchr()
- substr()

---

strtok

(PHP 4, PHP 5)

strtok — 文字列をトークンに分割する

説明

string **strtok** (string $str , string $token)

strtok() は文字列 (str) を 何らかの文字 token によって区切られている小さな文字列 (トークン) に分割します。 "This is an example string" のような文字列がある場合、 空白文字をトークンとするとこの文字列を個々の単語に分割することができます。

strtok は最初のコールの時のみ string 引数を使用することに注意してください。 strtok は、文字列のどこにいるのかの情報を保持しているため、 2回目以降のコールではトークンのみを必要とします。 最初からやりなおす場合や新しい文字列をトークンに分割する場合、 初期化するために再度string引数を指定してstrtokをコールします。 文字列は、引数の文字のどれかが見つかったはトークンに分割されます。

パラメータ

str

より小さい文字列 (トークン) に分割する文字列。

token

str を分割する際に使用する区切り文字。

返り値

文字列トークンを返します。

例

Example#1 strtok() の例

```
<?php
$string = "This is¥tan example¥nstring";
/* タブと改行をトークンの区切りとして使用します */
$tok = strtok($string, " ¥n¥t");

while ($tok !== false) {
    echo "Word=$tok<br />";
    $tok = strtok(" ¥n¥t");
}
?>
```

空の部分が見つかった場合の動作は PHP 4.1.0 で変更されました。 以前は空の文字列を返していました、新しい、正しい動作は、 文字列のその部分を単に読み飛ばします。

Example#2 古い strtok() の動作

```
<?php
$first_token  = strtok('/something', '/');
$second_token = strtok('/');
var_dump($first_token, $second_token);
?>
```

上の例の出力は以下となります。

```
string(0) ""
string(9) "something"
```

Example#3 新しい strtok() の動作

```
<?php
$first_token  = strtok('/something', '/');
$second_token = strtok('/');
var_dump($first_token, $second_token);
?>
```

上の例の出力は以下となります。

```
string(9) "something"
bool(false)
```

注意

警告

この関数は論理値 **FALSE** を返す可能性がありますが、**FALSE** として評価される 0 や "" といった値を返す可能性もあります。詳細については 論理値のセクションを参照してください。この関数の返り値を調べるには ===演算子 を使用してください。

参考

- split()
- explode()

---

strtolower

(PHP 4, PHP 5)

strtolower — 文字列を小文字にする

説明

string **strtolower** (string $str)

string のアルファベット部分をすべて小文字にして返します。

「アルファベット部分」は現在のロケールにより決定されます。 つまり、たとえばデフォルトの「C」ロケールである場合は、 A ウムラウト (Ä) のような文字は変換されません。

パラメータ

str

入力文字列。

返り値

小文字に変換した文字列を返します。

例

Example#1 strtolower() の例

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtolower($str);
echo $str; // mary had a little lamb and she loved it so を返します
?>
```

注意

注意: この関数はバイナリデータに対応しています。

参考

- strtoupper()
- ucfirst()
- ucwords()
- mb_strtolower()

---

strtoupper

(PHP 4, PHP 5)

strtoupper — 文字列を大文字にする

説明

string **strtoupper** (string $string)

string のアルファベット部分をすべて大文字にして返します。

「アルファベット部分」は現在のロケールにより決定されます。 つまり、たとえばデフォルトの「C」ロケールである場合は、 a ウムラウト (ä) のような文字は変換されません。

パラメータ

string

入力文字列。

返り値

大文字にした文字列を返します。

例

Example#1 strtoupper() の例

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtoupper($str);
echo $str; // 「MARY HAD A LITTLE LAMB AND SHE LOVED IT SO」を出力します。
?>
```

注意

> **注意**: この関数はバイナリデータに対応しています。

参考

- strtolower()
- ucfirst()
- ucwords()
- mb_strtoupper()

---

strtr

(PHP 4, PHP 5)

strtr — 特定の文字を変換する

説明

```
string strtr ( string $str , string $from , string $to )
string strtr ( string $str , array $replace_pairs )
```

この関数は str を走査し、 from に含まれる文字が見つかると、そのすべてを to の中で対応する文字に置き換え、 その結果を返します。

from と to の長さが異なる場合、長い方の余分な文字は無視されます。

パラメータ

str

変換する文字列。

from

to に変換される文字列。

to

from を置換する文字列。

replace_pairs

replace_pairs パラメータを to や from のかわりに使用することができます。この場合は array('from' => 'to', ...) 形式の配列となります。

返り値

この関数は str を走査し、 from に含まれる文字が見つかると、そのすべてを to の中で対応する文字に置き換え、 その結果を返します。

変更履歴

バージョン	説明
4.0.0	オプションのパラメータ to および from が追加されました。

例

Example#1 strtr() の例

```
<?php
$addr = strtr($addr, "äåö", "aao");
?>
```

strtr() は、引数を 2 つ指定してコールすることが可能です。 引数を 2 つ指定してコールした場合、異なった動作となります。 文字列from では、ソース文字列で変換を行う 「文字列 -> 文字列」の組を指定する必要があります。 strtr() は、常にまず最も長くマッチする文字列を探し、既に置換されている部分文字列については再度置換を行いません。

Example#2 2 つの引数を伴う strtr() の例

```
<?php
```

```
$trans = array("hello" => "hi", "hi" => "hello");
echo strtr("hi all, I said hello", $trans);
?>
```

上の例の出力は以下となります。

```
hello all, I said hi
```

参考

- ereg_replace()

---

substr_compare

(PHP 5)

substr_compare — 指定した位置から指定した長さの 2 つの文字列について、バイナリ対応で比較する

説明

int **substr_compare** (string $main_str , string $str , int $offset [, int $length [, bool $case_insensitivity]])

substr_compare() は、main_str の offset 文字目以降の最大 length 文字を、str と比較します。

パラメータ

main_str

str

offset

比較を開始する位置。 負の値を指定した場合は、文字列の最後から数えます。

length

比較する長さ。

case_insensitivity

case_insensitivity が **TRUE** の場合、 大文字小文字を区別せずに比較します。

返り値

main_str の offset 以降が str より小さい場合に負の数、 str より大きい場合に正の数、 等しい場合に 0 を返します。offset が main_str の長さ以上であり、かつ length が設定されている場合、**substr_compare()** は警告を表示して **FALSE** を返します。

変更履歴

バージョン	説明
5.1.0	負の offset を使用できるようになりました。

例

Example#1 substr_compare() の例

```
<?php
echo substr_compare("abcde", "bc", 1, 2); // 0
echo substr_compare("abcde", "de", -2, 2); // 0
echo substr_compare("abcde", "bcg", 1, 2); // 0
echo substr_compare("abcde", "BC", 1, 2, true); // 0
echo substr_compare("abcde", "bc", 1, 3); // 1
echo substr_compare("abcde", "cd", 1, 2); // -1
echo substr_compare("abcde", "abc", 5, 1); // 警告
?>
```

---

substr_count

(PHP 4, PHP 5)

substr_count — 副文字列の出現回数を数える

説明

int **substr_count** (string $haystack , string $needle [, int $offset [, int $length]])

substr_count() は、文字列 haystack の中での副文字列 needle の出現回数を返します。 needle は英大小文字を区別することに注意してください。

> **注意**: この関数は重なり合う副文字列をカウントしません。以下の例を見てください！

パラメータ

haystack

検索対象の文字列

needle

検索する副文字列

offset

開始位置のオフセット

length

指定したオフセット以降に副文字列で検索する最大長。 オフセットと長さの総和が *haystack* の長さよりも長い場合、警告が発生します。

返り値

この関数は 整数 を返します。

変更履歴

バージョン	説明
5.1.0	*offset* と *length* パラメータが追加されました。

例

Example#1 substr_count() の例

```
<?php
$text = 'This is a test';
echo strlen($text); // 14

echo substr_count($text, 'is'); // 2

// 文字列は 's is a test' になっているので, 1 が表示される
echo substr_count($text, 'is', 3);

// テキストは 's i' になっているので, 0 が表示される
echo substr_count($text, 'is', 3, 3);

// 5+10 > 14 なので、警告が発生する
echo substr_count($text, 'is', 5, 10);


// 重なっている副文字列はカウントされないので、1 が表示される
$text2 = 'gcdgcdgcd';
echo substr_count($text2, 'gcdgcd');
?>
```

参考

- *count_chars()*
- *strpos()*
- *substr()*
- *strstr()*

---

substr_replace

(PHP 4, PHP 5)

substr_replace — 文字列の一部を置換する

説明

mixed **substr_replace** (*mixed $string* , *string $replacement* , *int $start* [, *int $length*])

substr_replace()は、文字列 *string* の *start* および（オプションの）*length* パラメータで区切られた部分を *replacement* で指定した文字列に置換します。

パラメータ

string

入力文字列。

replacement

置換する文字列。

start

start が正の場合、置換は *string* で *start* 番目の文字から始まります。

start が負の場合、置換は *string* の終端から *start* 番目の文字から始まります。

length

正の値を指定した場合、 string の置換される部分の長さを表します。 負の場合、置換を停止する位置が string の終端から何文字目であるかを表します。このパラメータが省略された場合、 デフォルト値は strlen(string)、すなわち、 string の終端まで置換することになります。 当然、もし length がゼロだったら、 この関数は string の最初から start の位置に replacement を挿入するということになります。

返り値

結果の文字列を返します。もし、string が配列の場合、配列が返されます。

例

Example#1 substr_replace() の例

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr />¥n";

/* 以下の2つの例は、全ての $var で 'bob' で置換します。 */
echo substr_replace($var, 'bob', 0) . "<br />¥n";
echo substr_replace($var, 'bob', 0, strlen($var)) . "<br />¥n";

/* $var の先頭に 'bob' を挿入します */
echo substr_replace($var, 'bob', 0, 0) . "<br />¥n";

/* 次の2つの例は、$var の 'MNRPQR' を 'bob'で置換します */
echo substr_replace($var, 'bob', 10, -1) . "<br />¥n";
echo substr_replace($var, 'bob', -7, -1) . "<br />¥n";

/* $var から 'MNRPQR' を削除します */
echo substr_replace($var, '', 10, -1) . "<br />¥n";
?>
```

注意

> **注意**: この関数はバイナリデータに対応しています。

参考

- str_replace()
- substr()

---

substr

(PHP 4, PHP 5)

substr — 文字列の一部分を返す

説明

string ***substr*** *(string $string , int $start [, int $length])*

文字列 string の、start で指定された位置から length バイト分の文字列を返します。

パラメータ

string

入力文字列。

start

start が正の場合、返される文字列は、 string の 0 から数えて start 番目から始まる文字列となります。 例えば、文字列'abcdef'において位置 0にある文字は、'a'であり、 位置2には'c'があります。

start が負の場合、返される文字列は、 string の後ろから数えて start 番目から始まる文字列となります。

Example#1 負の start の使用

```
<?php
$rest = substr("abcdef", -1);    // "f" を返す
$rest = substr("abcdef", -2);    // "ef" を返す
$rest = substr("abcdef", -3, 1); // "d" を返す
?>
```

length

length が指定され、かつ正である場合、 返される文字列は start (string の長さに依存します) から数えてlength 文字数分となります。 もし string が start の文字列長より小さいもしくは等しい場合、FALSE が返されます。

length が指定され、かつ負である場合、 string の終端から多くの文字が省略されます (start が負の場合、 開始位置は文字列の終端を過ぎているので)。 もし start が切り出し位置を超える場合、 空文字が返されます。

Example#2 負の length の使用

```
<?php
$rest = substr("abcdef", 0, -1);  // "abcde" を返す
$rest = substr("abcdef", 2, -1);  // "cde" を返す
$rest = substr("abcdef", 4, -4);  // "" を返す
$rest = substr("abcdef", -3, -1); // "de" を返す
?>
```

返り値

文字列の一部を返します。

例

Example#3 基本的な substr() の使用法

```
<?php
echo substr('abcdef', 1);     // bcdef
echo substr('abcdef', 1, 3);  // bcd
echo substr('abcdef', 0, 4);  // abcd
echo substr('abcdef', 0, 8);  // abcdef
echo substr('abcdef', -1, 1); // f

// 文字列中の 1 文字にアクセスすることも
// "中括弧" を使用することで可能
$string = 'abcdef';
echo $string{0};                 // a
echo $string{3};                 // d
echo $string{strlen($string)-1}; // f

?>
```

参考

- strrchr()
- substr_replace()
- preg_match()
- trim()
- mb_substr()
- wordwrap()

---

trim

(PHP 4, PHP 5)

trim — 文字列の先頭および末尾にあるホワイトスペースを取り除く

説明

string **trim** (string $str [, string $charlist])

この関数は str の最初および最後から空白文字を取り除き、 取り除かれた文字列を返します。2番目のパラメータを指定しない場合、 **trim()**は以下の文字を削除します。

- " " (ASCII 32 (0x20)), 通常の空白。
- "¥t" (ASCII 9 (0x09)), タブ。
- "¥n" (ASCII 10 (0x0A)), リターン。
- "¥r" (ASCII 13 (0x0D)), 改行。
- "¥0" (ASCII 0 (0x00)), NULバイト
- "¥x0B" (ASCII 11 (0x0B)), 垂直タブ

パラメータ

str

ホワイトスペースを取り除く string

charlist

charlist パラメータにより、削除する 文字を指定することも可能です。削除したい全ての文字をリストに してください。..を文字の範囲を指定する際に 使用可能です。

返り値

ホワイトスペースを取り除いた文字列

変更履歴

バージョン	説明
4.1.0	charlist オプションパラメータが追加されました

例

Example#1 trim()の使用例

```
<?php

$text   = "¥t¥tThese are a few words :) ...  ";
$binary = "¥x09Example string¥x0A";
$hello  = "Hello World";
```

```
var_dump($text, $binary, $hello);

print "¥n";

$trimmed = trim($text);
var_dump($trimmed);

$trimmed = trim($text, " ¥t.");
var_dump($trimmed);

$trimmed = trim($hello, "Hdle");
var_dump($trimmed);

// ASCII 制御文字 (0 から 31 まで) を
// $binary の先頭および末尾から取り除きます
$clean = trim($binary, "¥x00..¥x1F");
var_dump($clean);

?>
```

上の例の出力は以下となります。

```
string(32) "          These are a few words :) ...   "
string(16) "     Example string
"
string(11) "Hello World"

string(28) "These are a few words :) ..."
string(24) "These are a few words :)"
string(5) "o Wor"
string(14) "Example string"
```

Example#2 trim() を用いて配列の値をトリミングする

```
<?php
function trim_value(&$value)
{
    $value = trim($value);
}

$fruit = array('apple','banana ', ' cranberry ');
var_dump($fruit);

array_walk($fruit, 'trim_value');
var_dump($fruit);

?>
```

上の例の出力は以下となります。

```
array(3) {
  [0]=>
  string(5) "apple"
  [1]=>
  string(7) "banana "
  [2]=>
  string(11) " cranberry "
}
array(3) {
  [0]=>
  string(5) "apple"
  [1]=>
  string(6) "banana"
  [2]=>
  string(9) "cranberry"
}
```

参考

- ltrim()
- rtrim()

---

ucfirst

(PHP 4, PHP 5)

ucfirst — 文字列の最初の文字を大文字にする

説明

string **ucfirst** (string $str)

str の最初の文字がアルファベットであれば、 それを大文字にします。

「アルファベット」かどうかというのは現在のロケールにより決定されます。 たとえば、デフォルトの "C" ロケールでは、a ウムラウト (ä) は変換されません。

パラメータ

str

入力文字列。

返り値

変換後の文字列を返します。

例

Example#1 ucfirst() の例

```
<?php
$foo = 'hello world!';
$foo = ucfirst($foo);             // Hello world!

$bar = 'HELLO WORLD!';
$bar = ucfirst($bar);             // HELLO WORLD!
$bar = ucfirst(strtolower($bar)); // Hello world!
?>
```

参考

- strtolower()
- strtoupper()
- ucwords()

---

ucwords

(PHP 4, PHP 5)

ucwords — 文字列の各単語の最初の文字を大文字にする

説明

string **ucwords** (string $str)

文字がアルファベットの場合、str の各単語の最初の文字を大文字にしたものを返します。

単語の定義は、空白文字（スペース、フォームフィード、改行、キャリッジリターン、 水平タブ、垂直タブ）の直後にあるあらゆる文字からなる文字列です。

パラメータ

str

入力文字列。

返り値

変更後の文字列を返します。

例

Example#1 ucwords() の例

```
<?php
$foo = 'hello world!';
$foo = ucwords($foo);             // Hello World!

$bar = 'HELLO WORLD!';
$bar = ucwords($bar);             // HELLO WORLD!
$bar = ucwords(strtolower($bar)); // Hello World!
?>
```

注意

注意: この関数はバイナリデータに対応しています。

参考

- strtoupper()
- strtolower()
- ucfirst()

---

vfprintf

(PHP 5)

vfprintf — フォーマットされた文字列をストリームに書き込む

説明

int **vfprintf** (resource $handle , string $format , array $args)

format によって作成された文字列を handle で指定したストリームに書き込みます。

fprintf() と同様の動作をしますが、 可変引数ではなく引数の配列を受け取って処理します。

パラメータ

handle

format

format については、 sprintf() のドキュメントで説明されています。

args

返り値

出力された文字列の長さを返します。

例

Example#1 vfprintf(): 数値のゼロ埋め

```
<?php
if (!($fp = fopen('date.txt', 'w')))
    return;

vfprintf($fp, "%04d-%02d-%02d", array($year, $month, $day));
// ISO 形式にフォーマットした日付を date.txt に書き込みます
?>
```

参考

- printf()
- sprintf()
- sscanf()
- fscanf()
- vsprintf()
- number_format()

---

vprintf

(PHP 4 >= 4.0.7, PHP 5)

vprintf — フォーマットされた文字列を出力する

説明

int **vprintf** (string $format , array $args)

format に基づき文字列フォーマットされた文字列を出力します（フォーマットは sprintf() のドキュメントに既述されています)。

printf() と動作は同じですが、可変長の引数ではなく、 配列を引数として受け取ります。

パラメータ

format

args

返り値

出力された文字列の長さを返します。

参考

- printf()
- sprintf()
- vsprintf()

---

vsprintf

(PHP 4 >= 4.0.7, PHP 5)

vsprintf — フォーマットされた文字列を返す

説明

string ***vsprintf*** *(string $format , array $args)*

sprintf()と動作は同じですが、 可変長の引数ではなく配列を引数とします。

パラメータ

format

args

返り値

(sprintf()のドキュメントに記述された) format に基づきフォーマットされた文字列として配列値を返します。

参考

- *sprintf()*
- *vprintf()*

---

wordwrap

(PHP 4 >= 4.0.2, PHP 5)

wordwrap — 文字列分割文字を使用して指定した文字数数に文字列を分割する

説明

string ***wordwrap*** *(string $str [, int $width [, string $break [, bool $cut]]])*

指定した文字数で、指定した文字を用いて文字列を分割します。

パラメータ

str

入力文字列。

width

カラムの幅。デフォルトは 75。

break

オプションのパラメータ break を用いて行を分割します。 デフォルトは '¥n' です。

cut

*cut を **TRUE** に設定すると、 文字列は常に指定した幅でラップされます。このため、 指定した幅よりも長い単語がある場合には、分割されます (2番目の例を参照ください)。*

返り値

指定した文字列を指定したカラム数で分割したものを返します。

変更履歴

バージョン	説明
4.0.3	オプションのパラメータ *cut* が追加されました。

例

Example#1 wordwrap() の例

```
<?php
$text = "The quick brown fox jumped over the lazy dog.";
$newtext = wordwrap($text, 20, "<br />¥n");

echo $newtext;
?>
```

上の例の出力は以下となります。

```
The quick brown fox<br />
jumped over the lazy<br />
dog.
```

Example#2 wordwrap() の例

```
<?php
$text = "A very long woooooooooooord.";
$newtext = wordwrap($text, 8, "¥n", true);

echo "$newtext¥n";
?>
```

上の例の出力は以下となります。

```
A very
long
woooooooo
ooooord.
```

参考

- nl2br()
- chunk_split()

目次

- str_repeat — 文字列を反復する
- str_replace — 検索文字列に一致したすべての文字列を置換する
- str_rot13 — 文字列に rot13 変換を行う
- str_shuffle — 文字列をランダムにシャッフルする
- str_split — 文字列を配列に変換する
- str_word_count — 文字列に使用されている単語についての情報を返す
- strcasecmp — 大文字小文字を区別しないバイナリセーフな文字列比較を行う
- strchr — strstr のエイリアス
- strcmp — バイナリセーフな文字列比較
- strcoll — ロケールに基づく文字列比較
- strcspn — マスクにマッチしない最初のセグメントの長さを返す
- strip_tags — 文字列から HTML および PHP タグを取り除く
- stripcslashes — addcslashes でクォートされた文字列をアンクォートする
- stripos — 大文字小文字を区別せずに文字列が最初に現れる位置を探す
- stripslashes — addslashes でクォートされた文字列のクォート部分を取り除く
- stristr — 大文字小文字を区別しない strstr
- strlen — 文字列の長さを得る
- strnatcasecmp — "自然順"アルゴリズムにより大文字小文字を区別しない文字列比較を行う
- strnatcmp — "自然順"アルゴリズムにより文字列比較を行う
- strncasecmp — バイナリセーフで大文字小文字を区別しない文字列比較を、最初の n 文字について行う
- strncmp — 最初の n 文字についてバイナリセーフな文字列比較を行う
- strpbrk — 文字列の中から任意の文字を探す
- strpos — 文字列が最初に現れる場所を見つける
- strrchr — 文字列中に文字が最後に現れる場所を取得する
- strrev — 文字列を逆順にする
- strripos — 文字列中で、特定の(大文字小文字を区別しない)文字列が最後に現れた位置を探す
- strrpos — 文字列中に、ある文字が最後に現れる場所を探す
- strspn — マスクに一致する最初のセグメントの長さを返す
- strstr — 文字列が最初に現れる位置を見つける
- strtok — 文字列をトークンに分割する
- strtolower — 文字列を小文字にする
- strtoupper — 文字列を大文字にする
- strtr — 特定の文字を変換する
- substr_compare — 指定した位置から指定した長さの 2 つの文字列について、バイナリ対応で比較する
- substr_count — 副文字列の出現回数を数える
- substr_replace — 文字列の一部を置換する
- substr — 文字列の一部分を返す
- trim — 文字列の先頭および末尾にあるホワイトスペースを取り除く
- ucfirst — 文字列の最初の文字を大文字にする
- ucwords — 文字列の各単語の最初の文字を大文字にする
- vfprintf — フォーマットされた文字列をストリームに書き込む
- vprintf — フォーマットされた文字列を出力する
- vsprintf — フォーマットされた文字列を返す
- wordwrap — 文字列分割文字を使用して指定した文字数数に文字列を分割する

Subversion 関数

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

この拡張モジュールは、バージョン管理システム » Subversion (SVN) を PHP から操作するためのものです。 コマンドラインから svn をコールせずに PHP スクリプトから SVN リポジトリや作業コピーを操作することができます。

要件

この拡張モジュールを使うのに、Subversion のバイナリは不要です。しかし、この拡張モジュールをコンパイルする際には libsvn (Subversion のヘッダ) が必須となります。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 »
http://pecl.php.net/package/svn

./configure が SVN 関連のファイルを見つけられない場合（Subversion が標準とは異なる場所にインストールされている場合など）は、**./config --with-svn=$USR_PATH** として include/subversion-1/ フォルダの場所を指定します

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

警告

この拡張モジュールを libsvn 1.3 でコンパイルした場合は、 Subversion 1.4 で作成した作業コピーを操作しようとすると失敗します。

リソース型

Subversion の低レベル関数 svn_fs_* は、 ローカルのファイルシステム上のリポジトリを指すリソースを私用します。 これは、svn_repos_fs() および svn_fs_revision_root() で作成します（要確認）。

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

SVN_REVISON_HEAD (integer)
HEAD リビジョンを指すマジックナンバー（-1）。

svn_auth_set_parameter() で使用する定数
SVN_AUTH_PARAM_DEFAULT_USERNAME (string)
ベーシック認証で使用するデフォルトのユーザ名を表すプロパティ。
SVN_AUTH_PARAM_DEFAULT_PASSWORD (string)
ベーシック認証で使用するデフォルトのパスワードを表すプロパティ。
SVN_AUTH_PARAM_NON_INTERACTIVE (string)
SVN_AUTH_PARAM_DONT_STORE_PASSWORDS (string)
SVN_AUTH_PARAM_NO_AUTH_CACHE (string)
SVN_AUTH_PARAM_SSL_SERVER_FAILURES (string)
SVN_AUTH_PARAM_SSL_SERVER_CERT_INFO (string)
SVN_AUTH_PARAM_CONFIG (string)
SVN_AUTH_PARAM_SERVER_GROUP (string)
SVN_AUTH_PARAM_CONFIG_DIR (string)
PHP_SVN_AUTH_PARAM_IGNORE_SSL_VERIFY_ERRORS (string)
SSL 証明書の検証エラーを無視するカスタムプロパティ。

ファイルシステムの定数
SVN_FS_CONFIG_FS_TYPE (string)
ファイルシステムの形式を決める設定キー。
SVN_FS_TYPE_BDB (string)
ファイルシステムは Berkeley-DB です。
SVN_FS_TYPE_FSFS (string)
ファイルシステムはネイティブのファイルシステムです。

予約されているプロパティの定数
SVN_PROP_REVISION_DATE (string)
svn:date
SVN_PROP_REVISION_ORIG_DATE (string)
svn:original-date
SVN_PROP_REVISION_AUTHOR (string)
svn:author
SVN_PROP_REVISION_LOG (string)
svn:log

作業コピーの状態に関する定数
svn_wc_status_none (int)
状態が存在しません。
svn_wc_status_unversioned (int)
この作業コピー内でバージョン管理されていません。
svn_wc_status_normal (int)
アイテムは存在しますが、特に何もない状態です。
svn_wc_status_added (int)
追加予約されています。
svn_wc_status_missing (int)
バージョン管理されているアイテムですが、作業コピー内に存在しません。
svn_wc_status_deleted (int)
削除予約されています。
svn_wc_status_replaced (int)
いったん削除されたあとに再度追加されました。
svn_wc_status_modified (int)
アイテム（の内容あるいはプロパティ）が変更されました。
svn_wc_status_merged (int)
ローカルでの変更内容がリポジトリの変更内容とマージされました。
svn_wc_status_conflicted (int)
ローカルでの変更内容がリポジトリでの変更内容と衝突（conflict）しています。
svn_wc_status_ignored (int)
バージョン管理されていないアイテムですが、無視するよう設定されています。
svn_wc_status_obstructed (int)
バージョン管理されていないアイテムが、 バージョン管理下のリソースを邪魔しています。
svn_wc_status_external (int)
バージョン管理されていないパスで、 svn:externals を使用しています。
svn_wc_status_incomplete (int)
ディレクトリのエントリー覧がありません。

ノードの種類を表す定数
svn_node_none (int)
存在しない。
svn_node_file (int)
ファイル。
svn_node_dir (int)
ディレクトリ。
svn_node_unknown (int)
Subversion が判別できない何か。

svn_add

(PECL svn:0.1-0.2)

svn_add — 作業ディレクトリ内のアイテムの追加を予約する

説明

bool **svn_add** (string $path [, bool $recursive [, bool $force]])

path が指すファイル、 ディレクトリあるいはシンボリックリンクを作業コピーに追加します。 その作業コピー上で次に svn_commit() をコールしたときに、アイテムがリポジトリに追加されます。

パラメータ

path

追加したいアイテムのパス。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

recursive

アイテムがディレクトリである場合、その中身を再帰的に追加するかどうかを指定します。 デフォルトは **TRUE** です。

force

true の場合は、すでにバージョン管理下におかれているディレクトリの中身も調べ、 もしバージョン管理されていないファイルがあればそれも追加します。 デフォルトは **FALSE** です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 svn_add() の例

svn status が次のような結果を返す状態で

```
$ svn status
?       foobar.txt
```

次のコードを実行します。

```
<?php
svn_add('foobar.txt');
?>
```

すると、foobar.txt がリポジトリへ追加予約されます。

参考

- » SVN ドキュメントの svn add

---

svn_auth_get_parameter

(PECL svn:0.1-0.2)

svn_auth_get_parameter — 認証パラメータを取得する

説明

string **svn_auth_get_parameter** (string $key)

認証パラメータ key を取得します。 使用できるキーとその意味については 認証定数の一覧 を参照ください。

パラメータ

key

キーの名前。認証に関する定数 のいずれかを指定します。

返り値

key の値を文字列で返します。 パラメータが存在しない場合は **NULL** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- svn_auth_set_parameter()
- 認証定数

---

svn_auth_set_parameter

(PECL svn:0.1-0.2)

svn_auth_set_parameter — 認証パラメータを設定する

説明

void **svn_auth_set_parameter** (string $key , string $value)

認証パラメータ key の値を value に設定します。 使用できるキーとその意味については 認証定数の一覧 を参照ください。

パラメータ

key

キーの名前。認証に関する定数 のいずれかを指定します。

value

そのパラメータに指定する値。 値の書式は、パラメータによって異なります。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 デフォルトの認証の例

この例は、SVN のデフォルトのユーザ名を 'Bob'、デフォルトのパスワードを 'abc123' に設定します。

```
<?php
svn_auth_set_parameter(SVN_AUTH_PARAM_DEFAULT_USERNAME, 'Bob');
svn_auth_set_parameter(SVN_AUTH_PARAM_DEFAULT_PASSWORD, 'abc123');
?>
```

参考

- svn_auth_get_parameter()
- 認証定数

---

svn_cat

(PECL svn:0.1-0.2)

svn_cat — リポジトリ内のファイルの内容を返す

説明

string **svn_cat** (string $repos_url [, int $revision_no])

リポジトリ内のファイルを指す URL repos_url の中身を返します。オプションでリビジョン番号 revision_no を指定します。

パラメータ

repos_url

リポジトリ内のアイテムへのパスを表す URL 文字列。

revision_no

取得したいリビジョン番号を表す整数値。デフォルトは HEAD リビジョン。

返り値

成功した場合にアイテムの内容を表す文字列、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

この例は、あるファイルのリビジョン 20 の内容を取得します。

```
<?php
$contents = svn_cat('http://www.example.com/svnroot/calc/gui.c', 28)
?>
```

参考

- svn_list()
- » SVN ドキュメントの svn cat

---

svn_checkout

(PECL svn:0.1-0.2)

svn_checkout — リポジトリから作業コピーをチェックアウトする

説明

bool **svn_checkout** (string $repos , string $targetpath [, int $revision])

repos にあるリポジトリの リビジョン revision を、 targetpath の作業コピーにチェックアウトします。

パラメータ

repos

チェックアウトしたいリポジトリ内のディレクトリを表す URL 文字列。

targetpath

チェックアウト先ディレクトリのローカルパス。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

revision

チェックアウトしたいリビジョン番号を表す整数値。デフォルトは HEAD リビジョンで、これは最新のリビジョンを表します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

この例は、リポジトリ内のディレクトリを calc という名前のローカルディレクトリにチェックアウトします。

```
<?php
svn_checkout('http://www.example.com/svnroot/calc/trunk', dirname(__FILE__) . '/calc');
?>
```

dirname(__FILE__) を使用して、 相対パス calc を絶対パスに変換する必要があります。 もし calc ディレクトリが存在するのなら、realpath() を使って絶対パスを取得することもできます。

参考

- svn_add()
- svn_commit()
- svn_status()
- svn_update()
- » SVN ドキュメントの svn checkout

---

svn_cleanup

(PECL svn:0.1-0.2)

svn_cleanup — 作業コピーディレクトリを再帰的にクリーンアップし、中途半端な操作を解消してロックを削除する

説明

bool **svn_cleanup** (string $workingdir)

作業コピーディレクトリ workingdir を再帰的にクリーンアップし、中途半端になっている操作を完了してロックを削除します。 作業コピーが中途半端な状態になってしまったときに、 再度使用可能にするために用います。

パラメータ

workingdir

クリーンアップしたい作業ディレクトリへのパス。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

この例は、help-me という作業ディレクトリのクリーンアップを行います。

```
<?php
svn_cleanup(realpath('help-me'));
?>
```

SVN における相対パスの扱いは少々変わっているので、 realpath() のコールは必須です。

参考

- **update()**
- » SVN ドキュメントの svn cleanup

---

svn_client_version

(PECL svn:0.1-0.2)

svn_client_version — SVN クライアントライブラリのバージョンを返す

説明

string **svn_client_version** (void)

SVN クライアントライブラリのバージョンを返します。

返り値

バージョン番号を表す文字列を返します。通常は x.y.z 形式となります。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

```
<?php
echo svn_client_version();
?>
```

上の例の出力は、たとえば以下のようになります。

```
1.3.1
```

svn_commit

(PECL svn:0.1-0.2)

svn_commit — 変更内容を、ローカルの作業コピーからリポジトリに送信する

説明

array **svn_commit** (string $log , array $targets [, bool $dontrecurse])

配列 targets で指定した、 ローカルの作業コピーで変更されたファイルの内容をリポジトリにコミットします。 ログメッセージは log で指定します。 targets にディレクトリを指定すると、 dontrecurse が true でない限りは再帰的にコミットを行います。

> **注意**: この関数では、認証関連のパラメータを指定することができません。 ユーザ名やパスワードは svn_auth_set_parameter() で指定する必要があります。

パラメータ

log

コミット時に使用するログメッセージ。

targets

コミットするローカルパスやファイルの配列。

警告

このパラメータは配列である必要があります。 対象がひとつだけであっても、文字列で指定することはできません。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

dontrecurse

配列 targets にディレクトリが指定された場合の再帰コミットを無効にするフラグ。 デフォルトは **FALSE** です。

返り値

以下のような形式の配列を返します。

```
array(
    0 => そのコミットのリビジョン番号
    1 => コミット日時を表す ISO 8601 形式の文字列
    2 => コミッターの名前
)
```

失敗した場合は **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

この例は、calculator ディレクトリの内容をリポジトリにコミットします。 ユーザ名は Bob、パスワードは abc123（ほんとはもうちょっとマシなパスワードにしたほうがいいでしょうね）です。

```
<?php
svn_auth_set_parameter(SVN_AUTH_PARAM_DEFAULT_USERNAME, 'Bob');
svn_auth_set_parameter(SVN_AUTH_PARAM_DEFAULT_PASSWORD, 'abc123');
var_dump(svn_commit('Log message of Bob¥'s commit', array(realpath('calculator'))));
?>
```

上の例の出力は以下となります。

```
array(
   0 => 1415,
    1 => '2007-05-26T01:44:28.453125Z',
   2 => 'Bob'
)
```

参考

- svn_auth_set_parameter()
- » SVN ドキュメントの svn commit

svn_diff

(PECL svn:0.1-0.2)

svn_diff — ふたつのパスの差分を再帰的に取得する

説明

array **svn_diff** (string $path1 , int $rev1 , string $path2 , int $rev2)

ふたつのパス path1 および path2 の差分を再帰的に取得します。

> **注意**: 汎用的な diff ツールとは異なり、 バージョン管理の対象になっているローカルファイルしか差分取得の対象となりません。 その他のファイルの差分は取得できません。

パラメータ

path1

最初のパス。SVN リポジトリのファイル/ディレクトリを指す URL、 あるいはローカルのファイル/ディレクトリのパスのいずれかとなります。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

警告

ローカルファイルのパスを指定する際に、 バックスラッシュのみを使用して通常のスラッシュが存在しない場合は、 パスの検出に失敗してしまいます。この関数を使う際には、 バックスラッシュをすべてスラッシュに置き換えるようにしましょう。

rev1

最初のパスのリビジョン番号。最新のリビジョンを指定したい場合は **SVN_REVISON_HEAD** を使用します。

path2

もうひとつのパス。詳細は path1 の説明を参照ください。

rev2

もうひとつのパスのリビジョン番号。詳細は rev2 の説明を参照ください。

返り値

ふたつのストリームの内容を含む配列のリストを返します。 最初のストリームが diff の出力で、もうひとつはエラー出力です。 このストリームを読み込むには fread() を使用します。エラーが発生した場合は **FALSE** あるいは **NULL** を返します。

デフォルトでは、diff の出力は Subversion 版の unified diff 形式となります。しかし、Subversion の設定によっては » 外部の diff エンジン を使うこともできます。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

ここでは、この関数の基本的な使用法と結果の取得法を示します。

```
<?php
list($diff, $errors) = svn_diff(
    'http://www.example.com/svnroot/trunk/foo', SVN_REVISION_HEAD,
    'http://www.example.com/svnroot/branches/dev/foo', SVN_REVISION_HEAD
);
if (!$diff) exit;
$contents = '';
while (!feof($diff)) {
  $contents .= fread($diff, 8192);
}
fclose($diff);
fclose($errors);
var_dump($contents);
?>
```

上の例の出力は以下となります。

```
Index: http://www.example.com/svnroot/trunk/foo
===================================================================
--- http://www.example.com/svnroot/trunk/foo           (.../foo) (revision 23)
+++ http://www.example.com/svnroot/branches/dev/foo (.../foo) (revision 27)
 // この後に diff の内容が続きます
```

Example#2 あるパスのふたつのリビジョンの差分の取得

この例では、外部リポジトリ上の同じパスのアイテムについて、 ふたつのリビジョン間の差分を簡単に取得できるようにしたラッパー関数を実装します（デフォルトの構文だと少々冗長なので）。

```
<?php
function svn_diff_same_item($path, $rev1, $rev2) {
```

```
    return svn_diff($path, $rev1, $path, $rev2);
}
?>
```

Example#3 ローカルファイルの差分の取得

この例では、ふたつのローカルファイルの差分を 簡単に取得できるようなラッパー関数を実装します。 realpath() による補正と、 バックスラッシュに関するバグの対応を含めています。

```
<?php
function svn_diff_local($path1, $rev1, $path2, $rev2) {
    $path1 = str_replace('¥¥', '/', realpath($path1));
    $path2 = str_replace('¥¥', '/', realpath($path2));
    return svn_diff($path1, $rev1, $path2, $rev2);
}
?>
```

参考

- » SVN ドキュメントの svn diff

---

svn_fs_abort_txn

(PECL svn:0.2)

svn_fs_abort_txn — トランザクションを中断し、成功したら true、失敗したら false を返す

説明

bool **svn_fs_abort_txn** (resource $txn)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

トランザクションを中断し、成功したら true、失敗したら false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_apply_text

(PECL svn:0.2)

svn_fs_apply_text — 置換に使用するストリームを作成して返す

説明

resource **svn_fs_apply_text** (resource $root , string $path)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

置換に使用するストリームを作成して返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_begin_txn2

(PECL svn:0.2)

svn_fs_begin_txn2 — 新しいトランザクションを作成する

説明

resource **svn_fs_begin_txn2** (resource $repos , int $rev)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

新しいトランザクションを作成します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_change_node_prop

(PECL svn:0.2)

svn_fs_change_node_prop — 成功したら true、失敗したら false を返す

説明

bool **svn_fs_change_node_prop** (resource $root , string $path , string $name , string $value)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

成功したら true、失敗したら false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_check_path

(PECL svn:0.1-0.2)

svn_fs_check_path — 指定したリポジトリの fsroot パスにどんなアイテムが存在するかを調べる

説明

int **svn_fs_check_path** (resource $fsroot , string $path)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

指定したリポジトリの fsroot パスにどんなアイテムが存在するかを調べます。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_contents_changed

(PECL svn:0.2)

svn_fs_contents_changed — コンテンツが変更されている場合に true、されていない場合に false を返す

説明

bool **svn_fs_contents_changed** (resource $root1 , string $path1 , resource $root2 , string $path2)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

コンテンツが変更されている場合に true、されていない場合に false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_copy

(PECL svn:0.2)

svn_fs_copy — ファイルやディレクトリをコピーし、成功したら true、失敗したら false を返す

説明

bool **svn_fs_copy** (resource $from_root , string $from_path , resource $to_root , string $to_path)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ファイルやディレクトリをコピーし、成功したら true、失敗したら false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_delete

(PECL svn:0.2)

svn_fs_delete — ファイルやディレクトリを削除し、成功したら true、失敗したら false を返す

説明

bool ***svn_fs_delete*** *(resource $root , string $path)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ファイルやディレクトリを削除し、成功したら true、失敗したら false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_dir_entries

(PECL svn:0.1-0.2)

svn_fs_dir_entries — 指定したパスのディレクトリを列挙し、ディレクトリ名とファイルタイプのハッシュを返す

説明

array ***svn_fs_dir_entries*** *(resource $fsroot , string $path)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

指定したパスのディレクトリを列挙し、ディレクトリ名とファイルタイプのハッシュを返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_file_contents

(PECL svn:0.1-0.2)

svn_fs_file_contents — 指定したバージョンの fs から、ファイルの中身を読み込むためのストリームを返す

説明

resource ***svn_fs_file_contents*** *(resource $fsroot , string $path)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

指定したバージョンの fs から、ファイルの中身を読み込むためのストリームを返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_file_length

(PECL svn:0.1-0.2)

svn_fs_file_length — *指定したバージョンの fs から、ファイルの長さを返す*

説明

int ***svn_fs_file_length*** *(resource $fsroot , string $path)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

指定したバージョンの fs から、ファイルの長さを返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_is_dir

(PECL svn:0.2)

svn_fs_is_dir — *ディレクトリを指すパスである場合に true、それ以外の場合に false を返す*

説明

bool ***svn_fs_is_dir*** *(resource $root , string $path)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ディレクトリを指すパスである場合に true、それ以外の場合に false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_is_file

(PECL svn:0.2)

svn_fs_is_file — *ファイルを指すパスである場合に true、それ以外の場合に false を返す*

説明

bool ***svn_fs_is_file*** *(resource $root , string $path)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ファイルを指すパスである場合に true、それ以外の場合に false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_make_dir

(PECL svn:0.2)

svn_fs_make_dir — *空のディレクトリを作成し、成功したら true、失敗したら false を返す*

説明

bool ***svn_fs_make_dir*** *(resource $root , string $path)*
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

空のディレクトリを作成し、成功したら true、失敗したら false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

svn_fs_make_file

(PECL svn:0.2)

svn_fs_make_file — 空のファイルを作成し、成功したら true、失敗したら false を返す

説明

bool **svn_fs_make_file** (resource $root , string $path)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

空のファイルを作成し、成功したら true、失敗したら false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_node_created_rev

(PECL svn:0.1-0.2)

svn_fs_node_created_rev — fsroot 配下のパスが作成されたリビジョンを返す

説明

int **svn_fs_node_created_rev** (resource $fsroot , string $path)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

fsroot 配下のパスが作成されたリビジョンを返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_node_prop

(PECL svn:0.1-0.2)

svn_fs_node_prop — ノードのプロパティの値を返す

説明

string **svn_fs_node_prop** (resource $fsroot , string $path , string $propname)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ノードのプロパティの値を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_props_changed

(PECL svn:0.2)

svn_fs_props_changed — プロパティが変更されている場合に true、それ以外の場合に false を返す

説明

bool **svn_fs_props_changed** (resource $root1 , string $path1 , resource $root2 , string $path2)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

プロパティが変更されている場合に true、それ以外の場合に false を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_revision_prop

(PECL svn:0.1-0.2)

svn_fs_revision_prop — 指定したプロパティの値を取得する

説明

string **svn_fs_revision_prop** (resource $fs , int $revnum , string $propname)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

指定したプロパティの値を取得します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_revision_root

(PECL svn:0.1-0.2)

svn_fs_revision_root — リポジトリのルートの指定したバージョンのハンドルを取得する

説明

resource **svn_fs_revision_root** (resource $fs , int $revnum)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

リポジトリのルートの指定したバージョンのハンドルを取得します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_txn_root

(PECL svn:0.2)

svn_fs_txn_root — トランザクションのルートを作成して返す

説明

resource **svn_fs_txn_root** (resource $txn)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

トランザクションのルートを作成して返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_fs_youngest_rev

(PECL svn:0.1-0.2)

svn_fs_youngest_rev — ファイルシステム内で一番若いリビジョン番号を返す

説明

int **svn_fs_youngest_rev** (*resource $fs*)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

ファイルシステム内で一番若いリビジョン番号を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_import

(PECL svn:0.2)

svn_import — バージョン管理されていないパスをリポジトリにインポートする

説明

bool **svn_import** (*string $path , string $url , bool $nonrecursive*)

バージョン管理されていないパス path を、 url で指定したリポジトリにコミットします。 path がディレクトリで nonrecursive が **FALSE** の場合は、 再帰的にインポートします。

パラメータ

path

インポートするファイルあるいはディレクトリのパス。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

url

インポート先のリポジトリの URL。

nonrecursive

ディレクトリを再帰処理するかどうか。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

この例は、関数の基本的な使用法を示すものです。 new-files というディレクトリを http://www.example.com/svnroot/incoming/abc にインポートするには、次のようにします。

```
<?php
svn_import(realpath('new-files'), 'http://www.example.com/svnroot/incoming/abc', false);
?>
```

参考

- svn_add()
- » SVN ドキュメントの svn import

---

svn_log

(PECL svn:0.1-0.2)

svn_log — 指定したリポジトリ URL のコミットログメッセージを返す

説明

array **svn_log** (*string $repos_url [, int $revision_no]*)

svn_log() は、指定したリビジョン URL repos_url の中身の完全な履歴を返します。 revision_no を指定した場合は、そのリビジョンの履歴を返します。 この関数は、**svn log --verbose -r $revision_no $repos_url** と同等です。

警告

長い期間使用しているリポジトリでは、出力が巨大なものになる可能性があります（すべてのリビジョンに対して配列の項目ができあがります）。 この関数は **--limit NUM** フラグをサポートしていません。 また、リビジョンの範囲指定にも対応していません（**revision_no** は整数で指定しなければなりません）。

パラメータ

repos_url

履歴を取得したいアイテムのリポジトリ URL。

revision_no

ログを取得したいリビジョン番号。最新の情報を取得するには **SVN_REVISON_HEAD** を使用します。

返り値

成功した場合は、この関数は次のような構造の配列を返します。

```
[0] => Array (最新のリポジトリから順に並びます)
(
    [rev] => リビジョン番号
    [author] => コミットした人の名前
    [msg] => ログメッセージ
    [date] => ISO 8601 形式、つまり date('c') と同じ形式の日付文字列
    [paths] => Array (変更したファイルについての説明)
        (
            [0] => Array
                (
                    [action] => 変更の種類を表す文字
                    [path] =>  変更されたファイルの絶対パス
                )
            [1] => ...
        )
)
[1] => ...
```

注意: 出力は、常に数値添字の二次元配列となります。 ログメッセージがなかったり、ひとつだけだったりする場合でも同じです。

action の値は、 » status の出力の最初の列の内容 のサブセットで、以下のいずれかとなります。

アクション

文字	説明
M	アイテム/プロパティが変更されました
A	アイテムが追加されました
D	アイテムが削除されました
R	アイテムが置き換えられました

何も変更されていない場合は、空の配列を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 svn_log() の例

```php
<?php
print_r( svn_log('http://www.example.com/', 23) );
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
    [0] => Array
    (
        [rev] => 23
        [author] => 'joe'
        [msg] => 'チーズとサラミをサンドイッチに追加した。'
          [date] => '2007-04-06T16:00:27-04:00'
        [paths] => Array
            (
                [0] => Array
                    (
                        [action] => 'M'
                         [path] =>  '/sandwich.txt'
                    )
            )
    )
)
```

Example#2 svn と svn_log() による --limit のシミュレート

このサンプル関数は、SVN 実行ファイルを使用することで **--limit** スイッチの機能をシミュレートしたものです。 この結果をもとにして **svn_log()** を使用しています。

> **注意**: この関数は、合計で limit + 1 回のリクエストを行います。最初の一回は必要なリビジョンを取得するためのもので、 必要な範囲について個別にログを取得します。

```
<?php
/**
 * 直近の $limit 件のログエントリを取得する
 * @param $repos_url ログを取得したいアイテムのリポジトリ URL
 * @param $limit 取得する最大件数
 */
function svn_log_limit($repos_url, $limit) {
    $limit = (int) $limit;
    if ($limit <= 0) return array();
   // -q フラグを使用して、サーバがログメッセージを送信しないようにします
    $output = shell_exec("svn log -q --limit $limit $repos_url");
    preg_match_all('/^r(¥d+) /m', $output, $matches);
    $ret = array();
    foreach ($matches[1] as $rev) {
        $log = svn_log($repos_url, (int) $rev);
        $ret[] = $log[0]; // log is only one item long
    }
    return $ret;
}
?>
```

参考

- » SVN ドキュメントの svn log

---

svn_ls

(PECL svn:0.1-0.2)

svn_ls — 指定したリポジトリ URL のディレクトリ内の一覧を返す（オプションでリビジョン番号も指定可能）

説明

array **svn_ls** (string $repos_url [, int $revision_no])

この関数は、指定したリポジトリ URL のファイルやディレクトリの一覧を返します。 オプションで、特定のリビジョンを指定することもできます。これは、**svn list $repos_url[@$revision_no]** と同等です。

> **注意**: この関数は、作業コピーに対して使用することはできません。 repos_url には 必ず リポジトリの URL を指定しなければなりません。

パラメータ

url

リポジトリの URL。たとえば **http://www.example.com/svnroot** のようになります。ローカルの Subversion リポジトリに対してファイルシステム経由でアクセスしたい場合は、 file URI スキームを使用して **file:///home/user/svn-repos** のようにします。

revision

一覧を取得したいリビジョン番号。省略した場合は HEAD リビジョンを使用します。

返り値

成功した場合は、この関数は次のような構造の配列を返します。

```
[0] => Array
    (
        [created_rev] => 最後に変更されたリビジョン番号
        [last_author] => 最後に変更した人の名前
        [size] => ファイルのバイト数
        [time] => 最終更新日時を、その古さによって 'M d H:i'
                  あるいは 'M d Y' のいずれかの形式で表したもの
        [time_t] => 最終更新日時の unix タイムスタンプ
        [name] => ファイル/ディレクトリの名前
        [type] => 'file' あるいは 'dir' のいずれか
    )
[1] => ...
```

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 svn_ls() の例

```
<?php
print_r( svn_ls('http://www.example.com/svnroot/') );
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => Array
         (
              [created_rev] => 20
              [last_author] => Joe
             [size] => 0
              [time] => Apr 02 09:28
              [time_t] => 1175520529
             [name] => tags
             [type] => dir
         )
     [1] => Array
         (
              [created_rev] => 23
              [last_author] => Bob
             [size] => 0
              [time] => Apr 02 15:15
              [time_t] => 1175541322
             [name] => trunk
             [type] => dir
         )
)
```

参考

- » SVN ドキュメントの svn list

---

svn_repos_create

(PECL svn:0.1-0.2)

svn_repos_create — 新しい subversion リポジトリを指定したパスに作成する

説明

resource **svn_repos_create** (string $path [, array $config [, array $fsconfig]])

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

新しい subversion リポジトリを指定したパスに作成します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_repos_fs_begin_txn_for_commit

(PECL svn:0.2)

svn_repos_fs_begin_txn_for_commit — 新しいトランザクションを作成する

説明

resource **svn_repos_fs_begin_txn_for_commit** (resource $repos , int $rev , string $author , string $log_msg)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

新しいトランザクションを作成します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_repos_fs_commit_txn

(PECL svn:0.2)

svn_repos_fs_commit_txn — トランザクションをコミットし、新しいリビジョンを返す

説明

int ***svn_repos_fs_commit_txn*** *(resource $txn)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

トランザクションをコミットし、新しいリビジョンを返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_repos_fs

(PECL svn:0.1-0.2)

svn_repos_fs — リポジトリ用に、ファイルシステム上のハンドルを取得する

説明

resource ***svn_repos_fs*** *(resource $repos)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

リポジトリ用に、ファイルシステム上のハンドルを取得します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_repos_hotcopy

(PECL svn:0.1-0.2)

svn_repos_hotcopy — *repospath* にあるリポジトリのホットコピーを作成し、*destpath* にコピーする

説明

bool ***svn_repos_hotcopy*** *(string $repospath , string $destpath , bool $cleanlogs)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

repospath にあるリポジトリのホットコピーを作成し、*destpath* にコピーします。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_repos_open

(PECL svn:0.1-0.2)

svn_repos_open — リポジトリの共有ロックをオープンする

説明

resource ***svn_repos_open*** *(string $path)*

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

リポジトリの共有ロックをオープンします。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

svn_repos_recover

(PECL svn:0.1-0.2)

svn_repos_recover — 指定したパスのリポジトリのリカバリ処理を実行する

説明

bool **svn_repos_recover** (string $path)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

指定したパスのリポジトリのリカバリ処理を実行します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

svn_status

(PECL svn:0.1-0.2)

svn_status — 作業コピー内のファイルやディレクトリの状態を返す

説明

array **svn_status** (string $path [, bool $recursive [, bool $get_all [, bool $update [, bool $no_ignore]]]])

作業コピー内のファイルやディレクトリの状態を返します。 作業コピー内での変更、追加、削除などの内容を取得できます。

パラメータ

path

状態を取得したいファイルやディレクトリへのパス。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

recursive

再帰的にディレクトリを掘り下げていくかどうか。 デフォルトは **TRUE** です。

get_all

変更の有無にかかわらずすべてのアイテムの状態を返すかどうか。 デフォルトは **FALSE** です。

update

作業コピーが最新の状態かどうかにかかわらず、 常にサーバからの情報を返すようにするかどうか (通常の変更チェックのほかに、 サーバ上の最新を反映していないアイテムについての情報も追加します)。 デフォルトは **FALSE** です。

no_ignore

新しいファイルを探すときに svn:ignore プロパティを無視するかどうか。 デフォルトは **FALSE** です。

返り値

数値添字の配列を返します。配列の各要素は連想配列となり、 リポジトリ内のアイテムの状態の詳細を表します。

```
Array (
    [0] => Array (
        // アイテムの情報
    )
    [1] => ...
)
```

アイテムの情報を表す連想配列には、以下のようなキーが含まれます。

path
このエントリのファイル/ディレクトリの、ローカルファイルシステム上でのパスを表す文字列。
text_status
アイテムのテキストの状態。とりうる値については 状態に関する定数 を参照ください。
repos_text_status
リポジトリ内でのアイテムのテキストの状態。正確な値となるのは、 update が **TRUE** の場合のみです。 とりうる値については 状態に関する定数 を参照ください。
prop_status
アイテムのプロパティの状態。とりうる値については 状態に関する定数 を参照ください。
repos_prop_status
リポジトリ内でのアイテムのプロパティの状態。正確な値となるのは、 update が **TRUE** の場合のみです。 とりうる値については 状態に関する定数 を参照ください。

locked
そのアイテムがロックされているかどうか（設定されるのは ***TRUE*** *の場合のみです）。*
copied
アイテムがコピーされている（追加予約されている）かどうか。（設定されるのは ***TRUE*** *の場合のみです）。*
switched
switch コマンドによる変更がされているかどうか。（設定されるのは ***TRUE*** *の場合のみです）。*

これらのキーは、バージョン管理されているアイテムについてのみ設定されます。

name
リポジトリ内でのアイテムのベース名。
url
リポジトリ内でのアイテムの URL。
repos
リポジトリのベース URL。
revision
作業コピーのリビジョン番号。
kind
アイテムの形式。file あるいは directory のいずれか。とりうる値については 型に関する定数 を参照ください。
schedule
そのアイテムに予定されているアクション（追加や削除など）。 これらのマジックナンバーに対応する定数は定義されていません。 定数を定義するには、次のようにします。

```php
<?php
if (!defined('svn_wc_schedule_normal')) {
    define('svn_wc_schedule_normal',  0); //  何も予定なし
    define('svn_wc_schedule_add',     1); //  追加予定
    define('svn_wc_schedule_delete',  2); //  削除予定
    define('svn_wc_schedule_replace', 3); //  追加・削除予定
}
?>
```

deleted
アイテムが削除されたが、親リビジョンのログがあるかどうか（設定されるのは ***TRUE*** *の場合のみです）。*
absent
アイテムが行方不明、 つまり Subversion 上では存在するはずなのに実際には存在しない という状態になっているかどうか（設定されるのは ***TRUE*** *の場合のみです）。*
incomplete
Whether or not the entries file for a directory is incomplete.（設定されるのは ***TRUE*** *の場合のみです）。*
cmt_date
最終更新日時を表す Unix タイムスタンプ（update の影響を受けません）。
cmt_rev
最終更新時のリビジョン（update の影響を受けません）。
cmt_author
最終更新者（update の影響を受けません）。
prop_time
プロパティの最新日時を表す Unix タイムスタンプ。
text_time
テキストの最新日時を表す Unix タイムスタンプ。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な使用例

これは、この関数の基本的な使用法を示すものです。

```php
<?php
print_r(svn_status(realpath('wc')));
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array (
    [0] => Array (
            [path] => /home/bob/wc/sandwich.txt
          [text_status] => 8 // アイテムが修正されました
           [repos_text_status] => 1 // 情報が存在しません。update を使用します
          [prop_status] => 3 // 変更されていません
           [repos_prop_status] => 1 // 情報が存在しません。update を使用します
           [name] => sandwich.txt
             [url] => http://www.example.com/svnroot/deli/trunk/sandwich.txt
            [repos] => http://www.example.com/svnroot/
          [revision] => 123
          [kind] => 1 // ファイル
          [schedule] => 0 // 特別なアクションは予定されていません
           [cmt_date] => 1165543135
          [cmt_rev] => 120
           [cmt_author] => Alice
           [prop_time] => 1180201728
           [text_time] => 1180201729
    )
)
```

参考

- *svn_update()*
- *svn_log()*
- *» SVN ドキュメントの svn status*

---

svn_update

(PECL svn:0.1-0.2)

svn_update — 作業コピーをアップデートする

説明

int **svn_update** (string $path [, int $revno [, bool $recurse]])

path の作業コピーを revno で指定したリビジョンにアップデートします。 recurse が true の場合は、 再帰的にアップデートします。

パラメータ

path

ローカルの作業コピーへのパス。

> **注意**: 相対パスは、PHP バイナリが存在するディレクトリを基準として解決されます。呼び出しもとのスクリプトの作業ディレクトリを基準とするには、realpath() あるいは dirname(__FILE__) を使用してください。

revno

アップデートするリビジョン番号。デフォルトは **SVN_REVISION_HEAD** です。

recurse

ディレクトリを再帰処理するかどうか。デフォルトは **TRUE** です。

返り値

成功した場合に新しいリビジョン番号、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

例

Example#1 基本的な例

この関数の基本的な使用法は、次のようになります。

```
<?php
echo svn_update(realpath('working-copy'));
?>
```

上の例の出力は、たとえば以下のようになります。

```
234
```

参考

- svn_checkout()
- svn_commit()
- » SVN ドキュメントの svn update

目次

- svn_fs_copy — ファイルやディレクトリをコピーし、成功したら true、失敗したら false を返す
- svn_fs_delete — ファイルやディレクトリを削除し、成功したら true、失敗したら false を返す
- svn_fs_dir_entries — 指定したパスのディレクトリを列挙し、ディレクトリ名とファイルタイプのハッシュを返す
- svn_fs_file_contents — 指定したバージョンの fs から、ファイルの中身を読み込むためのストリームを返す
- svn_fs_file_length — 指定したバージョンの fs から、ファイルの長さを返す
- svn_fs_is_dir — ディレクトリを指すパスである場合に true、それ以外の場合に false を返す
- svn_fs_is_file — ファイルを指すパスである場合に true、それ以外の場合に false を返す
- svn_fs_make_dir — 空のディレクトリを作成し、成功したら true、失敗したら false を返す
- svn_fs_make_file — 空のファイルを作成し、成功したら true、失敗したら false を返す
- svn_fs_node_created_rev — fsroot 配下のパスが作成されたリビジョンを返す
- svn_fs_node_prop — ノードのプロパティの値を返す
- svn_fs_props_changed — プロパティが変更されている場合に true、それ以外の場合に false を返す
- svn_fs_revision_prop — 指定したプロパティの値を取得する
- svn_fs_revision_root — リポジトリのルートの指定したバージョンのハンドルを取得する
- svn_fs_txn_root — トランザクションのルートを作成して返す
- svn_fs_youngest_rev — ファイルシステム内で一番若いリビジョン番号を返す
- svn_import — バージョン管理されていないパスをリポジトリにインポートする
- svn_log — 指定したリポジトリ URL のコミットログメッセージを返す
- svn_ls — 指定したリポジトリ URL のディレクトリ内の一覧を返す（オプションでリビジョン番号も指定可能）
- svn_repos_create — 新しい subversion リポジトリを指定したパスに作成する
- svn_repos_fs_begin_txn_for_commit — 新しいトランザクションを作成する
- svn_repos_fs_commit_txn — トランザクションをコミットし、新しいリビジョンを返す
- svn_repos_fs — リポジトリ用に、ファイルシステム上のハンドルを取得する
- svn_repos_hotcopy — repospath にあるリポジトリのホットコピーを作成し、destpath にコピーする
- svn_repos_open — リポジトリの共有ロックをオープンする
- svn_repos_recover — 指定したパスのリポジトリのリカバリ処理を実行する
- svn_status — 作業コピー内のファイルやディレクトリの状態を返す
- svn_update — 作業コピーをアップデートする

Shockwave Flash 関数

導入

PHP は、Paul Haeberli が作成した libswf モジュールにより Shockwave Flash ファイルを作成する機能を提供します。

> **注意**: SWF サポートは、PHP 4 RC2 で追加されました。
> libswf は、Windows でサポートされていません。このライブラリの 開発は中止されており、他のシステムへ移植するためのソースは入手できません。
> 最新の SWF サポートについては、 MING 関数を参照してください。
>
> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 5.0.0.

要件

PHP でこの拡張モジュールを使用するようコンパイルするためには、 libswf ライブラリを必要とします。 libswf は » ftp://ftp.sgi.com/sgi/graphics/grafica/flash/ でダウンロードする ことができます。

インストール手順

libswf を入手した後にするべきことは、configure に --with-swf[=DIR] を指定すること だけです。ただし、DIR はディレクトリ include および lib を有する場所です。 ディレクトリ include にはファイル swf.h が、ディレクトリ lib には ファイル libswf.a がある必要があります。libswf の配布ファイルを展開した際には、 二つのファイルは一つのディレクトリにあります。結果的に、ファイルを 適当な場所に手動でコピーする必要があります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

MOD_COLOR (integer)
MOD_MATRIX (integer)
TYPE_PUSHBUTTON (integer)
TYPE_MENUBUTTON (integer)
BSHitTest (float)
BSDown (float)

BSOver (*float*)
BSUp (*float*)
OverDowntoIdle (*integer*)
IdletoOverDown (*integer*)
OutDowntoIdle (*integer*)
OutDowntoOverDown (*integer*)
OverDowntoOutDown (*integer*)
OverUptoOverDown (*integer*)
OverUptoIdle (*integer*)
IdletoOverUp (*integer*)
ButtonEnter (*integer*)
ButtonExit (*integer*)
MenuEnter (*integer*)
MenuExit (*integer*)

例

PHP を Shockwave Flash サポート付きでインストールした後は、 Shockwave ファイルを PHP から作成することができるようになります。 何ができるかが分かると驚かされることでしょう。次のコードを見てください。

Example#1 SWF の例

```
<?php
swf_openfile("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2(-100, 100, -100, 100);
swf_defineline(1, -70, 0, 70, 0, .2);
swf_definerect(4, 60, -10, 70, 0, 0);
swf_definerect(5, -60, 0, -70, 10, 0);
swf_addcolor(0, 0, 0, 0);

swf_definefont(10, "Mod");
swf_fontsize(5);
swf_fontslant(10);
swf_definetext(11, "This be Flash wit PHP!", 1);

swf_pushmatrix();
swf_translate(-50, 80, 0);
swf_placeobject(11, 60);
swf_popmatrix();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix();
    swf_scale(1-($p*.9), 1, 1);
    swf_rotate(60*$p, 'z');
    swf_translate(20+20*$p, $p/1.5, 0);
    swf_rotate(270*$p,  'z');
    swf_addcolor($p, 0, $p/1.2, -$p);
    swf_placeobject(1, 50);
    swf_placeobject(4, 50);
    swf_placeobject(5, 50);
    swf_popmatrix();
    swf_showframe();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject(50);
    if (($i%4) == 0) {
        swf_showframe();
    }
}

swf_startdoaction();
swf_actionstop();
swf_enddoaction();

swf_closefile();
?>
```

---

swf_actiongeturl

(PHP 4)

swf_actiongeturl — Shockwave Flash ムービーから URL を得る

説明

void ***swf_actiongeturl*** (string $url , string $target)

指定したターゲット target からパラメータ url で指定したURLを得ます。

パラメータ

url

URL を表す文字列。

target

対象を表す文字列。

返り値

値を返しません。

swf_actiongotoframe

(PHP 4)

swf_actiongotoframe — フレームを実行した後、停止する

説明

void **swf_actiongotoframe** (int $framenumber)

関数 **swf_actiongotoframe()** は、 framenumber で指定したフレームまで移動し、 実行した後で停止します。

パラメータ

framenumber

フレーム番号。

返り値

値を返しません。

swf_actiongotolabel

(PHP 4)

swf_actiongotolabel — 指定したラベルを有するフレームを表示する

説明

void **swf_actiongotolabel** (string $label)

関数 **swf_actiongotolabel()** は、 label で指定したラベルを有するフレームを 表示した後で停止します。

パラメータ

label

フレームのラベル。

返り値

値を返しません。

swf_actionnextframe

(PHP 4)

swf_actionnextframe — フレームを一つ進める

説明

void **swf_actionnextframe** (void)

フレームを一つ進めます。

返り値

値を返しません。

swf_actionplay

(PHP 4)

swf_actionplay — 現在のフレームから flash ムービーの実行を開始する

説明

void **swf_actionplay** (void)

現在のフレームから flash ムービーを開始します。

返り値

値を返しません。

swf_actionprevframe

(PHP 4)

swf_actionprevframe — フレームを一つ戻す

説明

void **swf_actionprevframe** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

---

swf_actionsettarget

(PHP 4)

swf_actionsettarget — アクションのコンテキストを設定する

説明

void **swf_actionsettarget** (string $target)

全てのアクションのコンテキストを設定します。 この関数を、現在実行中の他の flash ムービーを 制御するために使用することが可能です。

パラメータ

target

対象を表す文字列。

返り値

値を返しません。

---

swf_actionstop

(PHP 4)

swf_actionstop — 現在のフレームで flash ムービーの実行を終了する

説明

void **swf_actionstop** (void)

現在のフレームで flash ムービーの実行を終了します。

返り値

値を返しません。

---

swf_actiontogglequality

(PHP 4)

swf_actiontogglequality — 低品質/高品質を切り替える

説明

void **swf_actiontogglequality** (void)

flash ムービーのクオリティの低品質/高品質を切り替えます。

返り値

値を返しません。

---

swf_actionwaitforframe

(PHP 4)

swf_actionwaitforframe — フレームがロードされていない場合にアクションをスキップする

説明

void **swf_actionwaitforframe** (int $framenumber , int $skipcount)

関数 **swf_actionwaitforframe()** は、パラメータ framenumber で指定したフレームがロードされているか どうかを確認し、ロードされていない場合はパラメータ skipcount で指定したアクションの数だけスキップ します。この関数は "Loading..." 型のアニメーションの場合に有用です。

パラメータ

framenumber

フレーム番号。

skipcount

スキップするアクションの番号。

返り値

値を返しません。

---

swf_addbuttonrecord

(PHP 4)

swf_addbuttonrecord — 現在のボタンの位置、外観、アクティブエリアを制御する

説明

void **swf_addbuttonrecord** (int $states , int $shapeid , int $depth)

ボタンを使用する際の仕様を定義します。

パラメータ

states

ボタンが持つことのできる状態を定義します。以下の定数 **BSHitTest**、 **BSDown**、**BSOver** あるいは **BSUp** のいずれか、あるいは全てとなります。

shapeid

2 番目のパラメータ shapeid はボタンの形状であり、 通常はボタンの形状のオブジェクト ID です。

depth

このパラメータは、現在のフレームにおけるボタンの位置です。

返り値

値を返しません。

例

Example#1 swf_addbuttonrecord() の例

```
<?php
swf_startButton($objid, TYPE_MENUBUTTON);
swf_addButtonRecord(BSDown|BSOver, $buttonImageId, 340);
swf_onCondition(MenuEnter);
swf_actionGetUrl("http://www.example.com", "_level1");
swf_onCondition(MenuExit);
swf_actionGetUrl("", "_level1");
swf_endButton();
?>
```

---

swf_addcolor

(PHP 4)

swf_addcolor — グローバル加算色を、指定した rbga 値に設定する

説明

void **swf_addcolor** (float $r , float $g , float $b , float $a)

指定した rgba 色にグローバル加算色を設定します。 この色は、この後(暗黙の内に)関数 swf_placeobject()、 swf_modifyobject()、swf_addbuttonrecord() で使用されます。

オブジェクトがスクリーンに描かれる時に、オブジェクトの色に 指定した値が加算されます。

パラメータ

r

赤の値。

g

緑の値。

b

青の値。

a

アルファ値。

これらの値は正または負のどちらにもなります。

返り値

値を返しません。

---

swf_closefile

(PHP 4)

swf_closefile — 現在の Shockwave Flash ファイルを閉じる

説明

void ***swf_closefile*** *([int $return_file])*

swf_openfile() 関数によりオープンされたファイルを閉じます。

パラメータ

return_file

設定されている場合、SWF ファイルの内容が関数から返されます。

例

Example#1 ユーザの入力に基づき簡単な flash ファイルを生成し、出力します。 また、データベースにこれを保存します。

```
<?php

// 変数$textはユーザにより投稿されます。

// データベースアクセス用のグローバル変数
// (swf_savedata()関数で使用)
$DBHOST = "localhost";
$DBUSER = "sterling";
$DBPASS = "secret";

swf_openfile("php://stdout", 256, 256, 30, 1, 1, 1);

    swf_definefont(10, "Ligon-Bold");
        swf_fontsize(12);
        swf_fontslant(10);

    swf_definetext(11, $text, 1);

    swf_pushmatrix();
        swf_translate(-50, 80, 0);
        swf_placeobject(11, 60);
    swf_popmatrix();

    swf_showframe();

    swf_startdoaction();
        swf_actionstop();
    swf_enddoaction();

$data = swf_closefile(1);

$data ?
  swf_savedata($data) :
  die("Error could not save SWF file");

// void swf_savedata (string data)
// 後で取得できるように生成されたファイルをデータベースに保存します。
function swf_savedata($data)
{
    global $DBHOST,
           $DBUSER,
           $DBPASS;

    $dbh = @mysql_connect($DBHOST, $DBUSER, $DBPASS);

    if (!$dbh) {
        die (sprintf("Error [%d]: %s",
                     mysql_errno(), mysql_error()));
    }

    $stmt = "INSERT INTO swf_files (file) VALUES ('$data')";

    $sth = @mysql_query($stmt, $dbh);
```

```
    if (!$sth) {
        die (sprintf("Error [%d]: %s",
                     mysql_errno(), mysql_error()));
    }

    @mysql_free_result($sth);
    @mysql_close($dbh);
}
?>
```

---

swf_definebitmap

(PHP 4)

swf_definebitmap — ビットマップを定義する

説明

void **swf_definebitmap** (int $objid , string $image_name)

関数 **swf_definebitmap()** は、 指定したイメージのビットマップを定義します。

パラメータ

objid

SWF オブジェクト ID。

image_name

GIF、JPEG、RGB あるいは FI 形式のイメージ。 このイメージは Flash JPEG または Flash カラーマップフォーマットに変換されます。

返り値

値を返しません。

---

swf_definefont

(PHP 4)

swf_definefont — フォントを定義する

説明

void **swf_definefont** (int $fontid , string $fontname)

関数 **swf_definefont()** は、パラメータ fontname で指定したフォントを定義し、パラメータ fontid で指定した ID に設定します。この関数は、fontname で指定したフォントを 現在のフォントに設定します。

パラメータ

fontid

フォントに指定する ID。

fontname

現在のフォントとして設定するフォント。

返り値

値を返しません。

---

swf_defineline

(PHP 4)

swf_defineline — 線を定義する

説明

void **swf_defineline** (int $objid , float $x1 , float $y1 , float $x2 , float $y2 , float $width)

線を定義します。

パラメータ

objid

オブジェクト ID。

x1

開始位置の x 座標。

y1

開始位置の y 座標。

x2

終了位置の x 座標。

y2

終了位置の y 座標。

width

線の幅。

返り値

値を返しません。

---

swf_definepoly

(PHP 4)

swf_definepoly — 多角形を定義する

説明

void **swf_definepoly** (int $objid , array $coords , int $npoints , float $width)

x, y 座標の配列で指定した多角形を定義します

パラメータ

objid

オブジェクト ID。

coords

x, y 座標の配列。

npoints

coords で指定した配列に含まれる全ての点の数です。

width

多角形の輪郭の幅。0.0 を設定した場合、 多角形は塗りつぶされます。

返り値

値を返しません。

---

swf_definerect

(PHP 4)

swf_definerect — 長方形を定義する

説明

void **swf_definerect** (int $objid , float $x1 , float $y1 , float $x2 , float $y2 , float $width)

左上と右下の点の座標を指定し、長方形を定義します。

パラメータ

objid

オブジェクト ID。

x1

左上の点の x 座標。

y1

左上の点の y 座標。

x2

右下の点の x 座標。

y2

右下の点の y 座標。

width

長方形の輪郭の幅。0.0 の場合は長方形が塗りつぶされます。

返り値

値を返しません。

---

swf_definetext

(PHP 4)

swf_definetext — テキスト文字列を定義する

説明

void **swf_definetext** (int $objid , string $str , int $docenter)

現在のフォントおよびフォントサイズを使用して、テキスト文字列を定義します。

パラメータ

objid

オブジェクト ID。

str

テキストを表す文字列。

docenter

docenter は単語のセンタリングを定義します。 docenter が 1 の場合、単語は x 方向にセンタリングされます。

返り値

値を返しません。

---

swf_endbutton

(PHP 4)

swf_endbutton — 現在のボタンの定義を終了する

説明

void **swf_endbutton** (void)

関数 **swf_endbutton()** は、 現在のボタンの定義を終了します。

返り値

値を返しません。

---

swf_enddoaction

(PHP 4)

swf_enddoaction — 現在のアクションを終了する

説明

void **swf_enddoaction** (void)

関数 swf_startdoaction() で開始した 現在のアクションを終了します。

返り値

値を返しません。

---

swf_endshape

(PHP 4)

swf_endshape — 現在の形状の定義を完結する

説明

void ***swf_endshape*** *(void)*

swf_endshape() *は、 現在の形状の定義を完結します。*

返り値

値を返しません。

---

swf_endsymbol

(PHP 4)

swf_endsymbol — シンボルの定義を終了する

説明

void ***swf_endsymbol*** *(void)*

関数 swf_startsymbol() で開始したシンボルの定義を終了します。

返り値

値を返しません。

---

swf_fontsize

(PHP 4)

swf_fontsize — フォントの大きさを変更する

説明

void ***swf_fontsize*** *(float $size)*

フォントの大きさを、 パラメータ size で指定した値に変更します。

パラメータ

size

フォントの大きさを表す整数値。

返り値

値を返しません。

---

swf_fontslant

(PHP 4)

swf_fontslant — フォントの傾きを設定する

説明

void ***swf_fontslant*** *(float $slant)*

現在のフォントの傾きを、パラメータ slant で指定した角度に設定します。

パラメータ

slant

正の値は前向きの傾きを生成し、負の値は負の傾きを生成します。

返り値

値を返しません。

---

swf_fonttracking

(PHP 4)

swf_fonttracking — 現在のフォントのトラッキングを設定する

説明

void ***swf_fonttracking*** *(float $tracking)*

パラメータ tracking で指定した値にフォントのトラッキングを 設定します。この関数は、文字およびテキスト間の空白を増加させるために使用され、 正の値は文字間の空白を増加させ、負の値は空白を減少させます。

パラメータ

tracking

フォントのトラッキング。

返り値

値を返しません。

---

swf_getbitmapinfo

(PHP 4)

swf_getbitmapinfo — ビットマップに関する情報を得る

説明

array **swf_getbitmapinfo** (int $bitmapid)

ビットマップに関する情報を返します。

パラメータ

bitmapid

ビットマップ ID。

返り値

以下の要素からなる配列を返します。

- "size" - ビットマップのバイト単位の大きさ。
- "width" - ビットマップのピクセル単位の幅。
- "height" - ビットマップのピクセル単位の高さ。

---

swf_getfontinfo

(PHP 4)

swf_getfontinfo — フォントの情報を得る

説明

array **swf_getfontinfo** (void)

大文字の A と小文字の x の高さをピクセル単位で調べることによって、 フォントについての情報を取得します。

返り値

以下のパラメータを持つ連想配列を返します。

- Aheight - 大文字 A のピクセル単位の高さ。
- xheight - 小文字 x のピクセル単位の高さ。

---

swf_getframe

(PHP 4)

swf_getframe — 現在のフレームのフレーム番号を得る

説明

int **swf_getframe** (void)

関数 **swf_getframe()** は、現在のフレームの 番号を得ます。

返り値

現在のフレーム番号を整数値で返します。

---

swf_labelframe

(PHP 4)

swf_labelframe — 現在のフレームにラベルを付ける

説明

void **swf_labelframe** (string $name)

現在のフレームにパラメータ name で 指定した名前を付けます。

パラメータ

name

フレームのラベル。

返り値

値を返しません。

---

swf_lookat

(PHP 4)

swf_lookat — 視点変換を定義する

説明

void **swf_lookat** (float $view_x , float $view_y , float $view_z , float $reference_x , float $reference_y , float $reference_z , float $twist)

視点の位置およびシーンの基準点の座標を定義することにより、 視点変換を定義します。

パラメータ

view_x

視点位置の x 座標。

view_y

視点位置の y 座標。

view_z

視点位置の z 座標。

reference_x

基準点の x 座標。

reference_y

基準点の y 座標。

reference_z

基準点の z 座標。

twist

視点の z 軸に関する回転を制御します。

返り値

値を返しません。

---

swf_modifyobject

(PHP 4)

swf_modifyobject — オブジェクトを修正する

説明

void **swf_modifyobject** (int $depth , int $how)

オブジェクトの位置または/および色を指定した深さ depth に更新します。

パラメータ

depth

深さを表す整数値。

how

更新するものを定義します。how は、定数 **MOD_MATRIX** あるいは **MOD_COLOR** のいずれか、 またはその両方の組み合わせです。

MOD_COLOR は（関数 swf_mulcolor() で指定した）現在の乗算色および(関数 swf_addcolor() で 指定した)現在の加算色をオブジェクトを塗る際に使用します。**MOD_MATRIX** はオブジェクトを配置する際に現在の行列を使用します。

返り値

値を返しません。

---

swf_mulcolor

(PHP 4)

swf_mulcolor — グローバル乗算色を指定した rbga 値に設定する

説明

void **swf_mulcolor** (float $r , float $g , float $b , float $a)

指定した rgba 色にグローバル乗算色を設定します。 この色は、この後（暗黙の内に）関数 swf_placeobject()、 swf_modifyobject()、swf_addbuttonrecord() で使用されます。

オブジェクトがスクリーンに描かれる時にオブジェクトの色に 指定した値が掛けられます。

パラメータ

r

赤の値。

g

緑の値。

b

青の値。

a

アルファ値。

これらの値は正または負のどちらにもなります。

返り値

値を返しません。

---

swf_nextid

(PHP 4)

swf_nextid — 次の未使用のオブジェクト ID を返す

説明

int **swf_nextid** (void)

関数 **swf_nextid()** は、次に利用可能なオブジェクト ID を返します。

返り値

ID を表す整数値を返します。

---

swf_oncondition

(PHP 4)

swf_oncondition — アクションリストのトリガとして使用されるトランジションを定義する

説明

void **swf_oncondition** (int $transition)

関数 **swf_onCondition()** は、 アクションリストのトリガとなるトランジションを定義します。

パラメータ

transition

使用可能なトランジションの型は複数あり、 TYPE_MENUBUTTON として定義されたボタンについては次のものがあります。

- IdletoOverUp

- *OverUptoIdle*
- *OverUptoOverDown*
- *OverDowntoOverUp*
- *IdletoOverDown*
- *OutDowntoIdle*
- *MenuEnter (IdletoOverUp|IdletoOverDown)*
- *MenuExit (OverUptoIdle|OverDowntoIdle)*

TYPE_PUSHBUTTON 用としては、次のオプションがあります。

- *IdletoOverUp*
- *OverUptoIdle*
- *OverUptoOverDown*
- *OverDowntoOverUp*
- *OverDowntoOutDown*
- *OutDowntoOverDown*
- *OutDowntoIdle*
- *ButtonEnter (IdletoOverUp|OutDowntoOverDown)*
- *ButtonExit (OverUptoIdle|OverDowntoOutDown)*

返り値

値を返しません。

---

swf_openfile

(PHP 4)

swf_openfile — 新規に Shockwave Flash ファイルをオープンする

説明

*void **swf_openfile** (string $filename , float $width , float $height , float $framerate , float $r , float $g , float $b)*

新しいファイルをオープンします。 この関数は最初にコールしなければなりません。そうしないと、 スクリプトは動作しません。

パラメータ

filename

SWF ファイルへのパス。出力を画面に送りたい場合は、これを php://stdout に設定します。

width

ムービーの幅。

height

ムービーの高さ。

framerate

フレームレート。

r

背景の赤の値。

g

背景の緑の値。

b

背景の青の値。

返り値

値を返しません。

変更履歴

バージョン	説明
4.0.1	*php://stdout* のサポートが追加されました。

---

swf_ortho2

(PHP 4)

swf_ortho2 — ユーザ座標の 2D 直交マッピングを現在のビューポイントに定義する

説明

*void **swf_ortho2** (float $xmin , float $xmax , float $ymin , float $ymax)*

ユーザ座標の 2 次元直交 マッピングを現在のビューポートに定義します。この関数は、 デフォルトを Flash ムービーの 1 対 1 のマッピングとします。

遠近法による変換を望む場合、関数 swf_perspective() を使用することが可能です。

パラメータ

xmin

xmax

ymin

ymax

返り値

値を返しません。

---

swf_ortho

(PHP 4 >= 4.0.1)

swf_ortho — 現在のビューポートにおけるユーザー座標の直交マッピングを定義する

説明

void **swf_ortho** (float $xmin , float $xmax , float $ymin , float $ymax , float $zmin , float $zmax)

ユーザ座標の直交マッピングを 現在のビューポートに定義します。

パラメータ

xmin

xmax

ymin

ymax

zmin

zmax

返り値

値を返しません。

---

swf_perspective

(PHP 4)

swf_perspective — 遠近法による投影変換を定義する

説明

void **swf_perspective** (float $fovy , float $aspect , float $near , float $far)

遠近法による投影変換を定義します。

> **注意**: 遠近法による投影を行う際には、様々な歪み効果が現れる可能性があります。 これは、Flash プレイヤーが有しているのは二次元行列のみだからです。 きれいに投影されないケースもあります。

パラメータ

fovy

y 方向の視野角。

aspect

描画領域のビューポートのアスペクト比。

near

近いクリッピング面。

far

遠いクリッピング面。

返り値

値を返しません。

---

swf_placeobject

(PHP 4)

swf_placeobject — オブジェクトを画面に配置する

説明

void **swf_placeobject** (int $objid , int $depth)

オブジェクトを、深さ depth で現在のフレームに配置します。

この関数は、(swf_mulcolor() で指定した) 現在の乗算色および(swf_addcolor() で指定した) 現在の加算色をオブジェクトに色づけする際に使用します。また、 オブジェクトの位置を決める際に現在の行列を使用します。

パラメータ

objid

オブジェクト ID。

depth

1 から 65535 までの間の値。

返り値

値を返しません。

---

swf_polarview

(PHP 4)

swf_polarview — 極座標で視点を定義する

説明

void **swf_polarview** (float $dist , float $azimuth , float $incidence , float $twist)

swf_polarview() 関数は、極座標で視点を定義します。

パラメータ

dist

視点とワールド空間の原点との距離。

azimuth

x,y 座標平面におけるアジマス角。 y 軸からの距離で表します。

incidence

y,z 平面における傾斜角を定義します。 z 軸からの距離で表します。傾斜角とは、 z 軸を基準としたビューポートの角度です。

twist

右手則を用いて、視線方向に関して視点を回転させる量を指定します。

返り値

値を返しません。

---

swf_popmatrix

(PHP 4)

swf_popmatrix — 以前の変換行列を回復する

説明

void **swf_popmatrix** (void)

現在の変換行列をスタックから回復します。

返り値

値を返しません。

---

swf_posround

(PHP 4)

swf_posround — オブジェクトを配置または移動する際の変換時の丸め処理を有効または無効にする

説明

void **swf_posround** (int $round)

オブジェクトを配置または移動する際に行う変換時の丸め処理を有効または無効にします。 丸めが有効になると読みやすくなる場合が多いです。

パラメータ

round

丸める操作の有無を指定します。 この値を 1 にした場合に丸めが有効になり、0 にした場合に丸めが無効になります。

返り値

値を返しません。

---

swf_pushmatrix

(PHP 4)

swf_pushmatrix — 現在の変換行列をスタックに入れる

説明

void **swf_pushmatrix** (void)

現在の変換行列をスタックに入れます。

返り値

値を返しません。

---

swf_removeobject

(PHP 4)

swf_removeobject — オブジェクトを削除する

説明

void **swf_removeobject** (int $depth)

depth で指定した深さのオブジェクトを削除します。

パラメータ

depth

深さを表す整数値。

返り値

値を返しません。

---

swf_rotate

(PHP 4)

swf_rotate — 現在の座標を回転する

説明

void **swf_rotate** (float $angle , string $axis)

パラメータ angle で指定した角度だけ、 パラメータ axis で指定した座標軸の周りに 現在の座標軸を回転します。

パラメータ

angle

回転角度。

axis

軸。有効な値は x (x 軸)、 y (y 軸) あるいは z (z 軸) のいずれかです。

返り値

値を返しません。

---

swf_scale

(PHP 4)

swf_scale — 現在の変換をスケーリングする

説明

void **swf_scale** (float $x , float $y , float $z)

swf_scale() は、指定した値で曲線の座標をスケーリングします。

パラメータ

x

x 方向のスケール値。

y

y 方向のスケール値。

z

z 方向のスケール値。

返り値

値を返しません。

---

swf_setfont

(PHP 4)

swf_setfont — 現在のフォントを変更する

説明

void **swf_setfont** (int $fontid)

swf_setfont() は、現在のフォントをパラメータ fontid で指定した値に設定します。

パラメータ

fontid

フォント ID。

返り値

値を返しません。

---

swf_setframe

(PHP 4)

swf_setframe — 指定したフレームに切替える

説明

void **swf_setframe** (int $framenumber)

アクティブなフレームを framenumber で指定したフレームに変更します。

パラメータ

framenumber

設定するフレーム番号。

返り値

値を返しません。

---

swf_shapearc

(PHP 4)

swf_shapearc — 円弧を描画する

説明

void **swf_shapearc** (float $x , float $y , float $r , float $ang1 , float $ang2)

円弧を描画します。

パラメータ

x

中心の x 座標。

y

中心の y 座標。

r

円弧の半径。

ang1

開始角度。

ang2

終了角度。

返り値

値を返しません。

---

swf_shapecurveto3

(PHP 4)

swf_shapecurveto3 — 三次ベジエ曲線を描画する

説明

void **swf_shapecurveto3** (float $x1 , float $y1 , float $x2 , float $y2 , float $x3 , float $y3)

指定した座標を使用して、三次ベジエ曲線を描画します。

その後、現在の位置は x3 および y3 が指す場所となります。

パラメータ

x1

最初の曲線外制御点の x 座標。

y1

最初の曲線外制御点の y 座標。

x2

二番目の曲線外制御点の x 座標。

y2

二番目の曲線外制御点の y 座標。

x3

終点の x 座標。

y3

終点の y 座標。

返り値

値を返しません。

---

swf_shapecurveto

(PHP 4)

swf_shapecurveto — 二点間に二次ベジエ曲線を描画する

説明

void ***swf_shapecurveto*** *(float $x1 , float $y1 , float $x2 , float $y2)*

現在の位置から、指定したふたつの点を使用して二次ベジエ曲線を描画します。

その後、現在の位置は x2 および y2 が指す場所となります。

パラメータ

x1

最初の点の x 座標。

y1

最初の点の y 座標。

x2

二番目の点の x 座標。

y2

二番目の点の y 座標。

返り値

値を返しません。

---

swf_shapefillbitmapclip

(PHP 4)

swf_shapefillbitmapclip — 現在の塗りつぶしモードをクリップ付きビットマップに設定する

説明

void ***swf_shapefillbitmapclip*** *(int $bitmapid)*

塗りつぶしをクリップ付きのビットマップに設定します。 空いた部分は、ビットマップで埋められます。

パラメータ

bitmapid

ビットマップ ID。

返り値

値を返しません。

---

swf_shapefillbitmaptile

(PHP 4)

swf_shapefillbitmaptile — 現在の塗りつぶしモードをタイル状のビットマップに設定する

説明

void ***swf_shapefillbitmaptile*** *(int $bitmapid)*

タイル状タイルでの塗りつぶしを設定します。空いた部分は、 ビットマップで埋められます。

パラメータ

bitmapid

ビットマップ ID。

返り値

値を返しません。

---

swf_shapefilloff

(PHP 4)

swf_shapefilloff — 塗りつぶしをオフにする

説明

void ***swf_shapefilloff*** *(void)*

現在の形状において塗りつぶしをオフにします。

返り値

値を返しません。

---

swf_shapefillsolid

(PHP 4)

swf_shapefillsolid — 現在の塗りつぶし色を指定した色に設定する

説明

void **swf_shapefillsolid** (float $r , float $g , float $b , float $a)

現在の塗りつぶし形式をソリッド(塗りつぶし有)にし、 塗りつぶし色を指定した値にします。

パラメータ

r

赤の値。

g

緑の値。

b

青の値。

a

アルファ値。

返り値

値を返しません。

---

swf_shapelinesolid

(PHP 4)

swf_shapelinesolid — 現在の行のスタイルを設定する

説明

void **swf_shapelinesolid** (float $r , float $g , float $b , float $a , float $width)

現在の線のスタイルを、指定した色と幅に設定します。

パラメータ

r

赤の値。

g

緑の値。

b

青の値。

a

アルファ値。

width

線の幅。0.0 が設定された場合、線は描画されません。

返り値

値を返しません。

---

swf_shapelineto

(PHP 4)

swf_shapelineto — 線を描画する

説明

void ***swf_shapelineto*** *(float $x , float $y)*

パラメータ x 、 y で指定した x,y 座標まで線を描きます。 現在の位置はパラメータ x,y となります。

パラメータ

x

対象の x 座標。

y

対象の y 座標。

返り値

値を返しません。

---

swf_shapemoveto

(PHP 4)

swf_shapemoveto — 現在の位置を移動する

説明

void ***swf_shapemoveto*** *(float $x , float $y)*

現在の位置を、指定した場所に移動します。

パラメータ

x

対象の x 座標。

y

対象の y 座標。

返り値

値を返しません。

---

swf_showframe

(PHP 4)

swf_showframe — 現在のフレームを表示する

説明

void ***swf_showframe*** *(void)*

現在のフレームを出力します。

返り値

値を返しません。

---

swf_startbutton

(PHP 4)

swf_startbutton — ボタンの定義を開始する

説明

void ***swf_startbutton*** *(int $objid , int $type)*

ボタンの定義を開始します。

パラメータ

objid

オブジェクト ID。

type

TYPE_MENUBUTTON あるいは **TYPE_PUSHBUTTON** のいずれかです。 定数 **TYPE_MENUBUTTON** はマウスを押したときにボタンからフォーカスを移すことを可能とし、 定数 **TYPE_PUSHBUTTON** は可能としません。

返り値

値を返しません。

---

swf_startdoaction

(PHP 4)

swf_startdoaction — 現在のフレームのアクションリストの記述を開始する

説明

void **swf_startdoaction** (void)

現在のフレームのアクションリストの記述を開始します。この関数は、 現在のフレームのアクションが定義される前にコールする必要があります。

返り値

値を返しません。

---

swf_startshape

(PHP 4)

swf_startshape — 複雑な形状を開始する

説明

void **swf_startshape** (int $objid)

複雑な形状を開始します。

パラメータ

objid

オブジェクト ID。

返り値

値を返しません。

---

swf_startsymbol

(PHP 4)

swf_startsymbol — シンボルを定義する

説明

void **swf_startsymbol** (int $objid)

シンボルとしてオブジェクト ID を定義します。シンボルは、同時に実行可能な 小さな flash ムービーです。

パラメータ

objid

シンボルとして定義したいオブジェクトの ID。

返り値

値を返しません。

---

swf_textwidth

(PHP 4)

swf_textwidth — 文字列の幅を得る

説明

float **swf_textwidth** (string $str)

現在のフォントおよびフォントサイズを用いて、 文字列の幅をピクセル値で返します。

パラメータ

str

文字列。

返り値

文字列の幅を表す float 値を返します。

---

swf_translate

(PHP 4)

swf_translate — 現在の座標軸を変換する

説明

void **swf_translate** (float $x , float $y , float $z)

指定した値で現在の座標を変換します。

パラメータ

x

x の値。

y

y の値。

z

z の値。

返り値

値を返しません。

---

swf_viewport

(PHP 4)

swf_viewport — 描画を行う範囲を選択する

説明

void **swf_viewport** (float $xmin , float $xmax , float $ymin , float $ymax)

描画の範囲として xmin から xmax 、 ymin から ymax までを 選択します。この関数がコールされない場合、描画範囲はデフォルトとして 画面の大きさになります。

パラメータ

xmin

xmax

ymin

ymax

返り値

値を返しません。

目次

- swf_addbuttonrecord — 現在のボタンの位置、外観、アクティブエリアを制御する
- swf_addcolor — グローバル加算色を、指定した rbga 値に設定する
- swf_closefile — 現在の Shockwave Flash ファイルを閉じる
- swf_definebitmap — ビットマップを定義する
- swf_definefont — フォントを定義する
- swf_defineline — 線を定義する
- swf_definepoly — 多角形を定義する
- swf_definerect — 長方形を定義する
- swf_definetext — テキスト文字列を定義する
- swf_endbutton — 現在のボタンの定義を終了する
- swf_enddoaction — 現在のアクションを終了する
- swf_endshape — 現在の形状の定義を完結する
- swf_endsymbol — シンボルの定義を終了する
- swf_fontsize — フォントの大きさを変更する
- swf_fontslant — フォントの傾きを設定する
- swf_fonttracking — 現在のフォントのトラッキングを設定する
- swf_getbitmapinfo — ビットマップに関する情報を得る
- swf_getfontinfo — フォントの情報を得る
- swf_getframe — 現在のフレームのフレーム番号を得る
- swf_labelframe — 現在のフレームにラベルを付ける
- swf_lookat — 視点変換を定義する
- swf_modifyobject — オブジェクトを修正する
- swf_mulcolor — グローバル乗算色を指定した rbga 値に設定する
- swf_nextid — 次の未使用のオブジェクト ID を返す
- swf_oncondition — アクションリストのトリガとして使用されるトランジションを定義する
- swf_openfile — 新規に Shockwave Flash ファイルをオープンする
- swf_ortho2 — ユーザ座標の 2D 直交マッピングを現在のビューポイントに定義する
- swf_ortho — 現在のビューポートにおけるユーザー座標の直交マッピングを定義する
- swf_perspective — 遠近法による投影変換を定義する
- swf_placeobject — オブジェクトを画面に配置する
- swf_polarview — 極座標で視点を定義する
- swf_popmatrix — 以前の変換行列を回復する
- swf_posround — オブジェクトを配置または移動する際の変換時の丸め処理を有効または無効にする
- swf_pushmatrix — 現在の変換行列をスタックに入れる
- swf_removeobject — オブジェクトを削除する
- swf_rotate — 現在の座標を回転する
- swf_scale — 現在の変換をスケーリングする
- swf_setfont — 現在のフォントを変更する
- swf_setframe — 指定したフレームに切替える
- swf_shapearc — 円弧を描画する
- swf_shapecurveto3 — 三次ベジエ曲線を描画する
- swf_shapecurveto — 二点間に二次ベジエ曲線を描画する
- swf_shapefillbitmapclip — 現在の塗りつぶしモードをクリップ付きビットマップに設定する
- swf_shapefillbitmaptile — 現在の塗りつぶしモードをタイル状のビットマップに設定する
- swf_shapefilloff — 塗りつぶしをオフにする
- swf_shapefillsolid — 現在の塗りつぶし色を指定した色に設定する
- swf_shapelinesolid — 現在の行のスタイルを設定する
- swf_shapelineto — 線を描画する
- swf_shapemoveto — 現在の位置を移動する
- swf_showframe — 現在のフレームを表示する
- swf_startbutton — ボタンの定義を開始する
- swf_startdoaction — 現在のフレームのアクションリストの記述を開始する
- swf_startshape — 複雑な形状を開始する
- swf_startsymbol — シンボルを定義する
- swf_textwidth — 文字列の幅を得る
- swf_translate — 現在の座標軸を変換する
- swf_viewport — 描画を行う範囲を選択する

Swish 関数

導入

swish 拡張モジュールは、Swish-e API 用のバインディングを提供します。 Swish-e は "Simple Web Indexing System for Humans - Enhanced" を略したもので、オープンソースのインデックス作成/検索システムです。 Swish-e 自体のライセンスは GPL ですが、Swish-e ソース

コードの URL にあるように、アプリケーションからそのライブラリにリンクすることができます。 » http://swish-e.org を参照ください。

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

要件

PECL/swish は、PHP 5.1.3 以降で動作します。

インストール手順

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/swish.

最新の PECL/swish Win32 DLL のダウンロード先はこちらです。 » php_swish.dll

実行時設定

設定ディレクティブは定義されていません。

例

Example#1 基本的な検索クエリ

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $results = $swish->query("test OR text");

    echo "Found ", $results->hits, " results¥n";
    while ($result = $results->nextResult()) {
        var_dump($result);
        break; //break after the first result
    }

} catch (SwishException $e) {
    echo "Error: ", $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
Found 9 results
object(SwishResult)#3 (8) {
   ["swishreccount"]=>
  int(1)
   ["swishrank"]=>
   int(1000)
   ["swishfilenum"]=>
   int(10)
   ["swishdbfile"]=>
   string(13) "index.swish-e"
   ["swishdocpath"]=>
   string(23) "README.SUBMITTING_PATCH"
   ["swishtitle"]=>
  NULL
   ["swishdocsize"]=>
   int(4557)
   ["swishlastmodified"]=>
   int(1072136752)
}
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

Swish::META_TYPE_UNDEF (integer)
Swish::META_TYPE_STRING (integer)
Swish::META_TYPE_ULONG (integer)
Swish::META_TYPE_DATE (integer)
Swish::IN_FILE_BIT (integer)
Swish::IN_TITLE_BIT (integer)
Swish::IN_HEAD_BIT (integer)
Swish::IN_BODY_BIT (integer)
Swish::IN_COMMENTS_BIT (integer)
Swish::IN_HEADER_BIT (integer)
Swish::IN_EMPHASIZED_BIT (integer)
Swish::IN_META_BIT (integer)
Swish::IN_FILE (integer)
Swish::IN_TITLE (integer)
Swish::IN_HEAD (integer)
Swish::IN_BODY (integer)
Swish::IN_COMMENTS (integer)
Swish::IN_HEADER (integer)

Swish::IN_EMPHASIZED (integer)
Swish::IN_META (integer)
Swish::IN_ALL (integer)

定義済みクラス

定義済みクラス

Swish

プロパティ

- indexes - 使用するインデックスおよびそのプロパティの配列。 プロパティの一覧は、Swish-e のバージョンに依存します。

メソッド

- Swish::__construct - 新しい Swish オブジェクトを作成します。 エラー時に SwishException をスローします。
- Swish->prepare - SwishSearch オブジェクトを準備して返します。 エラー時に SwishException をスローします。
- Swish->query - クエリを実行し、SwishResults オブジェクトを返します。 エラー時に SwishException をスローします。
- Swish->getMetaList - 指定したインデックスファイルのメタエントリの配列を返します。
- Swish->getPropertyList - 指定したインデックスファイルのプロパティの配列を返します。

SwishSearch

メソッド

- SwishSearch->setStructure - 検索オブジェクトの構造フラグを設定します。このフラグを使用して、 HTML ドキュメントの一部のみに検索制限をかけます。
- SwishSearch->setPhraseDelimiter - フレーズの区切り文字を設定します。デフォルトはダブルクォートです。
- SwishSearch->setSort - 結果の並び順を設定します。
- SwishSearch->setLimit - 結果の限界を設定します。 エラー時に SwishException をスローします。
- SwishSearch->resetLimit - 制限をリセットします。
- SwishSearch->execute - クエリを実行し、SwishResults オブジェクトを返します。 エラー時に SwishException をスローします。

SwishResults

プロパティ

- hits - この SwishResults オブジェクトの結果の数。
- indexes - 検索で使用するインデックスの配列。

メソッド

- SwishResults->nextResult - 次の SwishResult オブジェクト、 あるいはもう結果がない場合に **FALSE** を返します。
- SwishResults->seekResult - SwishResults オブジェクト内の現在のシーク位置を設定します。 エラー時に SwishException をスローします。
- SwishResults->getParsedWords - クエリ内の単語の配列を、ストップワードを削除して返します。
- SwishResults->getRemovedStopwords - 削除したストップワードの配列を返します。

SwishResult

プロパティ

- プロパティは Swish-e のバージョンに依存します。サンプルを参照ください。

メソッド

- SwishResult->getMetaList - この結果で使用するインデックスのメタエントリの配列を返します。
- SwishResult->stem - 単語の語幹を取得し、結果を文字列の配列で返します。 エラー時に SwishException をスローします。

SwishException

SwishException は、組み込みの Exception クラスを継承したもので、 このクラスと同じプロパティやメソッドを保持しています。詳細は 例外 (exceptions) を参照ください。

Swish::__construct

(PECL swish:0.1-0.3.0)

Swish::__construct — Swish オブジェクトを作成する

説明

void **Swish::__construct** (string $index_names)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

index_names

空白で区切られた、インデックスファイルの一覧。

エラー / 例外

エラー時に SwishException をスローします。

例

Example#1 Swish::__construct() の例

```
<?php

try {
    $swish = new Swish("index1 index2");
} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

foreach ($swish->indexes as $index) {
    var_dump($index["name"]);
    var_dump($index["headers"]["Total Words"]);
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
string(6) "index1"
int(1888)
string(6) "index2"
int(2429)
```

Swish->getMetaList

(PECL swish:0.1-0.3.0)

Swish->getMetaList — このインデックスのメタエントリー覧を取得する

説明

array **Swish->getMetaList** (string $index_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

index_name

インデックスファイルの名前。

返り値

指定したインデックスのメタエントリの配列を返します。

例

Example#1 基本的な Swish->getMetaList() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    var_dump($swish->getMetaList("index.swish-e"));

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
array(1) {
  [0]=>
  array(3) {
    ["Name"]=>
     string(12) "swishdefault"
    ["Type"]=>
    int(0)
    ["ID"]=>
    int(1)
  }
}
```

Swish->getPropertyList

(PECL swish:0.1-0.3.0)

Swish->getPropertyList — このインデックスのプロパティー覧を取得する

説明

array **Swish->getPropertyList** (string $index_name)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

index_name

インデックスファイルの名前。

返り値

指定したインデックスのプロパティー覧を配列で返します。

例

Example#1 基本的な Swish->getPropertyList() の例

```php
<?php

try {

    $swish = new Swish("index.swish-e");
    $properties = $swish->getPropertyList("index.swish-e");
     foreach ($properties as $prop) {
         echo $prop["Name"],"¥n";
    }

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
swishreccount
swishrank
swishfilenum
swishdbfile
swishdocpath
swishtitle
swishdocsize
swishlastmodified
```

Swish->prepare

(PECL swish:0.1-0.3.0)

Swish->prepare — 検索クエリを準備する

説明

object **Swish->prepare** ([string $query])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

準備した検索オブジェクトを返します。 後でこれを使用して何度でもクエリを実行することができます。

パラメータ

query

オプションのクエリ文字列。クエリは、SwishSearch->execute() メソッドで設定することもできます。

返り値

SwishSearch オブジェクトを返します。

エラー / 例外

エラー時に SwishException をスローします。

例

Example#1 基本的な Swish->prepare() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare("search query");
    $results = $search->execute();
    echo "Found: ", $results->hits, " hits¥n";

    $results = $search->execute("new search");

    echo "Found: ", $results->hits, " hits¥n";
} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
Found: 2 hits
Found: 5 hits
```

Swish->query

(PECL swish:0.1-0.3.0)

Swish->query — クエリを実行し、結果オブジェクトを返す

説明

```
object Swish->query ( string $query )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

デフォルトのパラメータで検索を行うためのお手軽なメソッドです。

パラメータ

query

クエリ文字列。

返り値

SwishResults オブジェクトを返します。

エラー / 例外

エラー時に SwishException をスローします。

例

Example#1 基本的な Swish->query() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $results = $swish->query("test query");

    echo "Found: ", $results->hits, " hits¥n";

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}
```

```
?>
```

上の例の出力は、たとえば以下のようになります。

```
Found: 1 hits
```

SwishResult->getMetaList

(PECL swish:0.1-0.3.0)

SwishResult->getMetaList — メタエントリの一覧を取得する

説明

```
array SwishResult->getMetaList ( void )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

返り値

swish->getmetalist() と同じ配列を返しますが、結果ハンドルからのインデックスファイルを使用します。

SwishResult->stem

(PECL swish:0.1-0.3.0)

SwishResult->stem — 指定した単語の語幹を取得する

説明

```
array SwishResult->stem ( string $word )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

インデックス化の際に使用したファジーモードにもとづいて、 単語の語幹を取得します。各結果オブジェクトはインデックスにリンクされるので、 結果はこのインデックスに基づくものとなります。

パラメータ

word

取り除く単語。

返り値

単語の活用形を取り除いた配列を返します（通常はひとつだけです）。

エラー / 例外

エラー時に SwishException をスローします。

例

Example#1 基本的な の例

```
<?php

try {

    $swish = new Swish("ext/swish/tests/index.swish-e");
    $results = $swish->query("testing OR others");

    if ($result = $results->nextResult()) {
        var_dump($result->stem("testing")); // この結果は、インデックスで使用している stemmer に依存します
        var_dump($result->stem("others"));
    }

} catch (SwishException $e) {
    echo "Error: ", $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
array(1) {
  [0]=>
  string(4) "test"
}
array(1) {
```

```
   [0]=>
   string(5) "other"
}
```

SwishResults->getParsedWords

(PECL swish:0.1-0.3.0)

SwishResults->getParsedWords — パースされた単語の配列を取得する

説明

array **SwishResults->getParsedWords** (string $index_name)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

indexi_name

Swish オブジェクトを初期化する際に使用するインデックスの名前。

返り値

パースされた単語からストップワードを取り除いた配列を返します。 パースされた単語の一覧は、結果の中で検索語句を強調させるのに便利です。

例

Example#1 基本的な SwishResults->getParsedWords() の例

```php
<?php

try {

    $swish = new Swish("index.swish-e");
    $results = $swish->query("'some characters' and numbers");

    var_dump($results->getParsedWords("index.swish-e"));
    var_dump($results->indexes[0]['parsed_words']); // 同じ結果を別の方法で得ます

} catch (SwishException $e) {
    echo "Error: ", $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
array(4) {
   [0]=>
   string(4) "some"
   [1]=>
   string(10) "characters"
   [2]=>
   string(3) "and"
   [3]=>
   string(7) "numbers"
}
array(4) {
   [0]=>
   string(4) "some"
   [1]=>
   string(10) "characters"
   [2]=>
   string(3) "and"
   [3]=>
   string(7) "numbers"
}
```

SwishResults->getRemovedStopwords

(PECL swish:0.1-0.3.0)

SwishResults->getRemovedStopwords — クエリから削除したストップワードの配列を取得する

説明

array **SwishResults->getRemovedStopwords** (string $index_name)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて

変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

index_name

Swish オブジェクトの初期化に使用するインデックスの名前。

返り値

クエリから取り除かれたストップワードの配列を返します。

SwishResults->nextResult

(PECL swish:0.1-0.3.0)

SwishResults->nextResult — 次の検索結果を取得する

説明

object **SwishResults->nextResult** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

返り値

結果セットから、次の SwishResult オブジェクトを返します。 もう結果がない場合に **FALSE** を返します。

例

Example#1 基本的な SwishResults->nextResult() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute("lost");
     while($result = $results->nextResult()) {
        /* 結果オブジェクトで何かを行います */
    }

} catch (SwishException $e) {
     echo $e->getMessage(), "¥n";
}

?>
```

SwishResults->seekResult

(PECL swish:0.1-0.3.0)

SwishResults->seekResult — 現在のシークポインタを指定した位置に設定する

説明

int **SwishResults->seekResult** (int $position)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

position

ゼロから始まる位置番号。ゼロより小さくすることはできません。

返り値

成功した場合に新しい位置を返します。

エラー / 例外

エラー時に SwishException をスローします。

例

Example#1 基本的な SwishResults->seekResult() の例

```
<?php
```

```
try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute("lost");

    var_dump($results->seekResult(0)); //  これは成功します
    var_dump($results->seekResult(100)); //  これは失敗します

} catch (SwishException $e) {
    echo "Error: ", $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
int(0)
Error: No more results
```

SwishSearch->execute

(PECL swish:0.1-0.3.0)

SwishSearch->execute — 検索を実行し、結果を取得する

説明

```
object SwishSearch->execute ([ string $query ] )
```

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

検索オブジェクトに設定したパラメータにもとづいて、 インデックスファイルを検索します。

パラメータ

query

クエリ文字列はオプションパラメータです。 Swish->prepare() メソッドで設定することもできます。クエリ文字列は 実行した後もそのまま残ります。 つまり、一度設定した内容で複数回検索を行うことができます。

返り値

SwishResults オブジェクトを返します。

エラー / 例外

エラー時に SwishException をスローします。

例

Example#1 基本的な SwishSearch->execute() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute("query");
    echo "First query found: ", $results->hits, " hits¥n";

    $results = $search->execute("new OR query");
    echo "Second query found: ", $results->hits, " hits¥n";

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
First query found: 2 hits
Second query found: 12 hits
```

SwishSearch->resetLimit

(PECL swish:0.1-0.3.0)

SwishSearch->resetLimit — 検索の制限をリセットする

説明

void **SwishSearch->resetLimit** (void)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

SwishSearch->setLimit で設定した検索の制限をリセットします。

返り値

値を返しません。

例

Example#1 基本的な SwishSearch->resetLimit() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute("time");
    echo "First query found: ", $results->hits, " hits¥n";

    $search->setLimit("swishdocsize", "3000", "6000"); // ドキュメントの大きさを 3000 バイトから 6000 バイトまでに制限します
    $results = $search->execute("time");
    echo "Second query found: ", $results->hits, " hits¥n";

    $search->resetLimit();
    $results = $search->execute("time");
    echo "Third query found: ", $results->hits, " hits¥n";

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
First query found: 5 hits
Second query found: 2 hits
Third query found: 5 hits
```

SwishSearch->setLimit

(PECL swish:0.1-0.3.0)

SwishSearch->setLimit — 検索の限界を設定する

説明

void **SwishSearch->setLimit** (string $property , string $low , string $high)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

property

検索結果のプロパティ名。

low

プロパティの最小値。

high

プロパティの最大値。

返り値

値を返しません。

エラー / 例外

エラー時に SwishException をスローします。

例

Example#1 基本的な SwishSearch->setLimit() の例

```
<?php
try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute("time");
    echo "First query found: ", $results->hits, " hits¥n";

    $i = 0;
    while($result = $results->nextResult()) {
        echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes¥n";
    }

    $search->setLimit("swishdocsize", "3000", "6000"); // ドキュメントのサイズを 3000 バイトから 6000 バイトまでに制限します
    $results = $search->execute("time");
    echo "Second query found: ", $results->hits, " hits¥n";

    $i = 0;
    while($result = $results->nextResult()) {
        echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes¥n";
    }

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
First query found: 5 hits
Hit #1 - 4261 bytes
Hit #2 - 37937 bytes
Hit #3 - 7126 bytes
Hit #4 - 15427 bytes
Hit #5 - 4768 bytes
Second query found: 2 hits
Hit #1 - 4261 bytes
Hit #2 - 4768 bytes
```

SwishSearch->setPhraseDelimiter

(PECL swish:0.1-0.3.0)

SwishSearch->setPhraseDelimiter — フレーズの区切り文字を設定する

説明

void **SwishSearch->setPhraseDelimiter** (string $delimiter)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

delimiter

フレーズの区切り文字。デフォルトの区切り文字はダブルクォートです。

返り値

値を返しません。

例

Example#1 基本的な SwishSearch->setPhraseDelimiter() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute('"every time"'); // "every time" を検索します
    echo "First query found: ", $results->hits, " hits¥n";

    $search->setPhraseDelimiter("'");
    $results = $search->execute("'every time'"); // 同じクエリですが、別の区切り文字を使用します
    echo "Second query found: ", $results->hits, " hits¥n";

    $search->setPhraseDelimiter('"');
    $results = $search->execute("'every time'"); // "every" および "time" を検索します
    echo "Third query found: ", $results->hits, " hits¥n";

    //let's look at parsed words
    var_dump($results->getParsedWords("index.swish-e"));
```

```
} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
First query found: 1 hits
Second query found: 1 hits
Third query found: 2 hits
array(2) {
  [0]=>
  string(5) "every"
  [1]=>
  string(4) "time"
}
```

SwishSearch->setSort

(PECL swish:0.1-0.3.0)

SwishSearch->setSort — 並び順を設定する

説明

void **SwishSearch->setSort** (string $sort)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

sort

結果の並び順は、プロパティ名とソート方向 ("asc" あるいは "desc") を組み合わせた文字列で指定します。 たとえば "swishrank desc"、"swishdocpath asc"、"swishtitle asc"、 "swishdocsize desc"、"swishlastmodified desc" などです。

返り値

値を返しません。

例

Example#1 基本的な SwishSearch->setSort() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute("time");
    echo "First query found: ", $results->hits, " hits¥n";

    $i = 0;
    while($result = $results->nextResult()) {
        echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes¥n";
    }

    $search->setSort("swishdocsize desc"); //  ドキュメントのサイズで並べ替えます
    $results = $search->execute("time");
    echo "Second query found: ", $results->hits, " hits¥n";

    $i = 0;
    while($result = $results->nextResult()) {
        echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes¥n";
    }

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
First query found: 5 hits
Hit #1 - 4261 bytes
Hit #2 - 37937 bytes
Hit #3 - 7126 bytes
Hit #4 - 15427 bytes
Hit #5 - 4768 bytes
Second query found: 5 hits
Hit #1 - 37937 bytes
Hit #2 - 15427 bytes
Hit #3 - 7126 bytes
Hit #4 - 4768 bytes
```

Hit #5 - 4261 bytes

SwishSearch->setStructure

(PECL swish:0.1-0.3.0)

SwishSearch->setStructure — 検索オブジェクトの構造フラグを設定する

説明

void **SwishSearch->setStructure** (int $structure)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

structure

構造フラグはビットマスクで表され、HTML ドキュメントの特定の部分 (title、meta、body など) に検索対象を絞り込みます。 指定できる値の一覧を以下に示します。複数の値を組み合わせるには、 論理 OR 演算子を使用します。以下に例を示します。

- **Swish::IN_FILE**
- **Swish::IN_TITLE**
- **Swish::IN_HEAD**
- **Swish::IN_BODY**
- **Swish::IN_COMMENTS**
- **Swish::IN_HEADER**
- **Swish::IN_EMPHASIZED**
- **Swish::IN_META**

返り値

値を返しません。

例

Example#1 基本的な SwishSearch->setStructure() の例

```
<?php

try {

    $swish = new Swish("index.swish-e");
    $search = $swish->prepare();

    $results = $search->execute("time");
    echo "First query found: ", $results->hits, " hits¥n";

    $search->setStructure(Swish::IN_TITLE|Swish::IN_HEAD); // title および head を検索します
    $results = $search->execute("time");
    echo "Second query found: ", $results->hits, " hits¥n";

    $search->setStructure(Swish::IN_ALL); //  ドキュメント全体を検索します。デフォルト値です
    $results = $search->execute("time");
    echo "Third query found: ", $results->hits, " hits¥n";

} catch (SwishException $e) {
    echo $e->getMessage(), "¥n";
}

?>
```

上の例の出力は、たとえば以下のようになります。

```
First query found: 5 hits
Second query found: 0 hits
Third query found: 5 hits
```

目次

- *Swish->query — クエリを実行し、結果オブジェクトを返す*
- *SwishResult->getMetaList — メタエントリの一覧を取得する*
- *SwishResult->stem — 指定した単語の語幹を取得する*
- *SwishResults->getParsedWords — パースされた単語の配列を取得する*
- *SwishResults->getRemovedStopwords — クエリから削除したストップワードの配列を取得する*
- *SwishResults->nextResult — 次の検索結果を取得する*
- *SwishResults->seekResult — 現在のシークポインタを指定した位置に設定する*
- *SwishSearch->execute — 検索を実行し、結果を取得する*
- *SwishSearch->resetLimit — 検索の制限をリセットする*
- *SwishSearch->setLimit — 検索の限界を設定する*
- *SwishSearch->setPhraseDelimiter — フレーズの区切り文字を設定する*
- *SwishSearch->setSort — 並び順を設定する*
- *SwishSearch->setStructure — 検索オブジェクトの構造フラグを設定する*

Sybase 関数

導入

要件

インストール手順

Sybase-DB サポートを有効にするには、PHP の configure で --with-sybase[=DIR] を指定して ください。DIR は Sybase のホームディレクトリで、デフォルトは /home/sybase です。 Sybase-CT サポートを有効にするには、PHP の configure で --with-sybase-ct[=DIR] を指定して ください。DIR は Sybase のホームディレクトリで、デフォルトは /home/sybase です。

実行時設定

php.ini の設定により動作が変化します。

Sybase 設定オプション

名前	デフォルト	変更の可否	変更履歴
sybase.allow_persistent	"1"	PHP_INI_ALL	PHP <= 4.0.2 では PHP_INI_ALL、PHP <= 4.0.3 では PHP_INI_SYSTEM。
sybase.max_persistent	"-1"	PHP_INI_ALL	PHP <= 4.0.2 では PHP_INI_ALL、PHP <= 4.0.3 では PHP_INI_SYSTEM。
sybase.max_links	"-1"	PHP_INI_ALL	PHP <= 4.0.2 では PHP_INI_ALL、PHP <= 4.0.3 では PHP_INI_SYSTEM。
sybase.interface_file	"/usr/sybase/interfaces"	PHP_INI_SYSTEM	
sybase.min_error_severity	"10"	PHP_INI_ALL	
sybase.min_message_severity	"10"	PHP_INI_ALL	
sybase.compatability_mode	"0"	PHP_INI_ALL	
magic_quotes_sybase	"0"	PHP_INI_ALL	PHP 6.0.0 で削除されました。

以下に設定ディレクティブに関する簡単な説明を示します。

sybase.allow_persistent boolean

持続的な Sybase 接続を使用可能とするかどうか。

sybase.max_persistent integer

プロセス毎の持続的な Sybase 接続の最大数。-1 は制限無しを意味します。

sybase.max_links integer

プロセス毎の持続的接続を含む Sybase 接続の最大数。 -1 は制限無しを意味します。

sybase.min_error_severity integer

出力するエラーの重要度の下限。

sybase.min_message_severity integer

出力するエラーの重要度の下限。

sybase.compatability_mode boolean

PHP 3.0の古いバージョンとの互換モード。onの場合、結果を 文字列データとして処理する代わりにPHPが自動的にSybaseの型に基づ き結果の型を決めるようにします。この互換モードがサポートされ続け ることはおそらくないため、自分のコードに必要な変更を行い、このオ プションをoffにしてください。

magic_quotes_sybase boolean

magic_quotes_sybase がonの場合、 magic_quotes_gpc または magic_quotes_runtime が有効の時にシングルクオートはバックスラッシュではなくシングルク オートでエスケープされます。

注意: magic_quotes_sybase がONの時に、 magic_quotes_gpc の設定を完全に上書きする ことに注意してください。この場合、たとえ magic_quotes_gpc が有効の場合でも ダブルクオートもバックスラッシュもNULもエスケープされません。

Sybase-CT configuration options

名前	デフォルト	変更の可否	変更履歴
sybct.allow_persistent	"1"	PHP_INI_SYSTEM	PHP <= 4.0.2 では PHP_INI_ALL、PHP 4.0.2 以降で使用可能です。PHP 4.0.3 で削除されました。
sybct.max_persistent	"-1"	PHP_INI_SYSTEM	PHP <= 4.0.2 では PHP_INI_ALL、PHP 4.0.2 以降で使用可能です。PHP 4.0.3 で削除されました。
sybct.max_links	"-1"	PHP_INI_SYSTEM	PHP <= 4.0.2 では PHP_INI_ALL、PHP 4.0.2 以降で使用可能です。PHP 4.0.3 で削除されました。
sybct.min_server_severity	"10"	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 4.0.3 で削除されました。
sybct.min_client_severity	"10"	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 4.0.3 で削除されました。
sybct.hostname	NULL	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 4.0.3 で削除されました。
sybct.deadlock_retry_count	"0"	PHP_INI_ALL	PHP 4.3.0 以降で使用可能です。

以下に設定ディレクティブに関する簡単な説明を示します。

sybct.allow_persistent boolean

持続的なSybase-CT接続を使用可能にするかどうか。 デフォルトは、onです。

sybct.max_persistent integer

プロセス毎の持続的なSybase接続の最大数。-1は制限無しを意味します。

sybct.max_links integer

プロセス毎の持続的接続を含むSybase-CT接続の最大数。 -1は制限無しを意味します。

sybct.min_server_severity integer

sybct.min_server_severity以上の重要度を有するサーバーメッセージ は警告を出力します。この値は、スクリプト内で sybase_min_server_severity()をコールすること により設定することも可能です。デフォルトは10で、 重要度がこの値以上の情報が出力されます。

sybct.min_client_severity integer

sybct.min_client_severity以上の重要度を有する クライアントライブラリメッセージが警告として出力されます。 この値は、スクリプトで sybase_min_client_severity()をコールすること により設定することも可能です。デフォルトは10で、 出力を結果的に無効にしています。

sybct.login_timeout integer

接続の試みが失敗した場合に、成功するまで待ち続ける最大の秒数。 接続の試みがタイムアウトした際にもし max_exeution_time をこえていたら、 接続失敗の処理をする前にスクリプトが終了してしまうことに注意しましょう。 デフォルトは 1 分です。

sybct.timeout integer

select_db やクエリ操作が失敗した場合に、成功するまで待ち続ける秒数。 操作がタイムアウトした際にもし max_exeution_time をこえていたら、 接続失敗の処理をする前にスクリプトが終了してしまうことに注意しましょう。 デフォルトは無制限です。

sybct.hostname string

sp_whoで表示するための接続を行うホスト名。デフォルトはありません。

sybct.deadlock_retry_count int

デッドロックが発生した場合に何回目まで再試行するかを設定します。 デフォルトは -1 すなわち "永遠に" です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

定義済み定数

定数は定義されていません。

---

sybase_affected_rows

(PHP 4, PHP 5)

sybase_affected_rows — 直近のクエリで変更された行の数を得る

説明

int **sybase_affected_rows** ([resource $link_identifier])

sybase_affected_rows() は、指定したリンク ID が 指すサーバにおいて直近の INSERT、UPDATE、DELETE クエリで変更された レコードの数を

返します。

このコマンドは、SELECT 文には使用できません。レコードを変更する 文のみに使用できます。SELECT から返された行の数を得たい場合は sybase_num_rows() を使用してください。

パラメータ

link_identifier

リンク ID を省略した場合は、最後にオープンしたリンクを使用します。

返り値

変更された行数を整数値で返します。

例

Example#1 削除クエリ

```
<?php
/* データベースに接続します */
sybase_connect('SYBASE', '', '') or
die("接続できません");
sybase_select_db("db");

sybase_query("DELETE FROM sometable WHERE id < 10");
printf("削除したレコード数: %d\n", sybase_affected_rows());
?>
```

上の例の出力は以下となります。

```
削除したレコード数: 10
```

参考

- sybase_num_rows()

---

sybase_close

(PHP 4, PHP 5)

sybase_close — Sybase 接続を閉じる

説明

bool **sybase_close** ([resource $link_identifier])

sybase_close()は、指定されたリンク link_identifier が指す Sybase データベースへのリンクを閉じます。

持続的でないリンクは、スクリプトの実行終了時に自動的に閉じられるため、 この関数は、通常の場合コールする必要がないことに注意してください。

sybase_close() は、 sybase_pconnect() により生成された 持続的なリンクは閉じません。

パラメータ

link_identifier

リンク ID を省略した場合は、最後にオープンしたリンクを使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- sybase_connect()
- sybase_pconnect()

---

sybase_connect

(PHP 4, PHP 5)

sybase_connect — Sybase サーバ接続をオープンする

説明

resource **sybase_connect** ([string $servername [, string $username [, string $password [, string $charset [, string $appname]]]]])

sybase_connect() は、Sybase サーバへの接続を確立します。

同じ引数で **sybase_connect()** を 2 度コールした場合、 新たなリンクは確立されず、代わりに既にオープンされたリンクのリンク ID が返されます。

sybase_close() をコールすることにより明示的に閉じた場合を除き、 サーバへのリンクはスクリプトの実行終了時に閉じられます。

パラメータ

servername

引数 servername は、'interfaces' ファイル中で定義された 有効なサーバ名である必要があります。

username

Sybase のユーザ名。

password

username のパスワード。

charset

接続に使用する文字セットを指定します。

appname

返り値

成功した場合に正の Sybase リンク ID、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.0.2	charset パラメータが追加されました。

例

Example#1 sybase_connect() の例

```
<?php
$link = sybase_connect('SYBASE', '', '')
        or die("接続できません !");
echo "接続に成功しました";
sybase_close($link);
?>
```

参考

- sybase_pconnect()
- sybase_close()

---

sybase_data_seek

(PHP 4, PHP 5)

sybase_data_seek — 内部行ポインタを移動する

説明

bool **sybase_data_seek** (resource $result_identifier , int $row_number)

sybase_data_seek()は、指定された結果 ID が指す Sybase 結果の内部行ポインタを指定した行番号に移動します。 sybase_fetch_row() を次にコールした場合、 その行が返されます。

パラメータ

result_identifier

row_number

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- sybase_fetch_row()

---

sybase_deadlock_retry_count

(PHP 4 >= 4.3.0, PHP 5)

sybase_deadlock_retry_count — デッドロックの再試行回数を設定する

説明

```
void sybase_deadlock_retry_count ( int $retry_count )
```

sybase_deadlock_retry_count() を使用して、デッドロックの 際の再試行回数を定義することが可能です。デフォルトでは、デッドロックが 発生した際は Sybase によってプロセスが停止されたり実行中のスクリプトが 停止したり（例えば set_time_limit()）、あるいは クエリが成功するまでずっと再試行を繰り返します。

パラメータ

retry_count

retry_count の値	
-1	永遠に再試行します（デフォルト）。
0	再試行しません。
n	n 回再試行します。

返り値

値を返しません。

注意

> **注意**: この関数は、Sybase の CT ライブラリインターフェイスでのみ使用可能で、DB ライブラリでは使用できません。

---

sybase_fetch_array

(PHP 4, PHP 5)

sybase_fetch_array — 行を配列として取り込む

説明

```
array sybase_fetch_array ( resource $result )
```

sybase_fetch_array() は sybase_fetch_row() の拡張版です。データを結果配列の 数値インデックスに保存するのに加えて、フィールド名をキーとした 連想インデックスにもデータを保存します。

sybase_fetch_array() を使用した場合でも、 かなりの機能が付加されるにもかかわらず、 sybase_fetch_row() を使用した場合に比べて著しく遅くなるということはないということを重要なこととして記しておきます。

パラメータ

result

返り値

取り込んだ行を表す配列を返します。もう行がない場合には **FALSE** を返します。

> **注意**: 同じ名前のフィールドを選択する場合（たとえば join を行った場合など）、 連想インデックスは順に番号を付加します。 詳細は例を参照ください。

例

Example#1 同一フィールド名

```php
<?php

$dbh = sybase_connect('SYBASE', '', '');
$q = sybase_query('SELECT * FROM p, a WHERE p.person_id= a.person_id');
var_dump(sybase_fetch_array($q));
sybase_close($dbh);
?>
```

上の例の出力は以下のようになります（2 つのテーブルには、ともに "person_id" という名前のカラムがあると仮定します）。

```
array(4) {
   [0]=>
   int(1)
    ["person_id"]=>
   int(1)
   [1]=>
   int(1)
    ["person_id1"]=>
   int(1)
}
```

参考

- sybase_fetch_row()
- sybase_fetch_assoc()
- sybase_fetch_object()

---

sybase_fetch_assoc

(PHP 4 >= 4.3.0, PHP 5)

sybase_fetch_assoc — 結果の行を連想配列として取得する

説明

array **sybase_fetch_assoc** (resource $result)

sybase_fetch_assoc() は sybase_fetch_row() で数値インデックスのかわりに カラム名を使用するバージョンです。複数のテーブルから同じ名前の カラムが取得された場合、それらの名前は name, name1, name2, ..., nameN のような形式で返されます。

sybase_fetch_assoc() を使用した場合でも、 かなりの機能が付加されるにもかかわらず、 sybase_fetch_row() を使用した場合に比べて著しく遅くなるということはないということを重要なこととして記しておきます。

パラメータ

result

返り値

取得した行に対応する配列を返します。行がもうない場合には **FALSE** を返します。

注意

> **注意**: この関数は、Sybase の CT ライブラリインターフェイスでのみ使用可能で、DB ライブラリでは使用できません。

参考

- sybase_fetch_row()
- sybase_fetch_array()
- sybase_fetch_object()

---

sybase_fetch_field

(PHP 4, PHP 5)

sybase_fetch_field — 結果からフィールド情報を取得する

説明

object **sybase_fetch_field** (resource $result [, int $field_offset])

sybase_fetch_field() は、あるクエリーの結果において、 フィールドに関する情報を得るために使用します。

パラメータ

result

field_offset

フィールドオフセットが指定されない場合、 **sybase_fetch_field()** によりまだ取り込まれていない次のフィールドが取り込まれます。

返り値

フィールド情報を含むオブジェクトを返します。

オブジェクトのプロパティを以下に示します。

- name - カラム名。そのカラムがある関数の結果である場合、 このプロパティは、computed#N にセットされます。ただし、#N はシリアルナンバーです。
- column_source - そのカラムが取り出されたテーブル
- max_length - カラムの最大長
- numeric - そのカラムが数値である場合に 1
- type - カラムのデータ型

参考

- sybase_field_seek()

---

sybase_fetch_object

(PHP 4, PHP 5)

sybase_fetch_object — 行をオブジェクトとして取り込む

説明

object ***sybase_fetch_object*** *(resource $result [, mixed $object])*

sybase_fetch_object() *は sybase_fetch_array() に似ていますが、 違いが一つあります。それは、配列の代わりにオブジェクトを返すことです。*

速度面では、この関数は sybase_fetch_array() と同等であり、 sybase_fetch_row() とほとんど同じです （違いはわずかです）。

パラメータ

result

object

返されれるオブジェクトの型を指定するには、2 番目のパラメータ object を使用します。このパラメータを指定しなかった場合は、 オブジェクトは stdClass 型となります。

返り値

取り込まれた行に対するプロパティを有するオブジェクトを返します。 また、行がもうない場合に ***FALSE*** *を返します。*

変更履歴

バージョン	説明
	この関数は数値のオブジェクトメンバを返さなくなりました。 以前はこのような挙動でした。 `object(stdclass)(3) { [0]=> string(3) "foo" ["foo"]=> string(3) "foo" [1]=> string(3) "bar" ["bar"]=> string(3) "bar" }`
4.3.0	しかし、新しい挙動は次のようになります。 `object(stdclass)(3) { ["foo"]=> string(3) "foo" ["bar"]=> string(3) "bar" }`

例

Example#1 sybase_fetch_object() の結果を Foo で返す

```php
<?php
    class Foo {
        var $foo, $bar, $baz;
    }

    // [...]
    $qrh= sybase_query('SELECT foo, bar, baz FROM example');
    $foo= sybase_fetch_object($qrh, 'Foo');
    $bar= sybase_fetch_object($qrh, new Foo());
    // [...]
?>
```

参考

- sybase_fetch_array()
- sybase_fetch_row()

---

sybase_fetch_row

(PHP 4, PHP 5)

sybase_fetch_row — 行を配列として取得する

説明

array ***sybase_fetch_row*** *(resource $result)*

sybase_fetch_row() *は、指定された結果 ID が指す結果から 1 行分のデータを取り込みます。*

連続的に ***sybase_fetch_rows()*** *をコールした場合、 結果セットにおける次の行が返されます。 また、もう行がない場合は* ***FALSE*** *が返されます。*

パラメータ

result

返り値

取り込まれた行に対応する配列を返します。もう行がない場合は **FALSE** を返します。 各結果カラムは *0* から始まる配列オフセットに保持されます。

データ型

PHP	Sybase
string	VARCHAR, TEXT, CHAR, IMAGE, BINARY, VARBINARY, DATETIME
int	NUMERIC（小数点以下なし）, DECIMAL（小数点以下なし）, INT, BIT, TINYINT, SMALLINT
float	NUMERIC（小数点以下あり）, DECIMAL（小数点以下あり）, REAL, FLOAT, MONEY
NULL	NULL

参考

- *sybase_fetch_array()*
- *sybase_fetch_assoc()*
- *sybase_fetch_object()*
- *sybase_data_seek()*
- *sybase_result()*

---

sybase_field_seek

(PHP 4, PHP 5)

sybase_field_seek — フィールドオフセットを設定する

説明

bool ***sybase_field_seek*** *(resource $result , int $field_offset)*

指定したフィールドオフセットに移動します。次にフィールドオフセットを指定しないで *sybase_fetch_field()* をコールした場合、 このフィールドが返されます。

パラメータ

result

field_offset

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- *sybase_fetch_field()*

---

sybase_free_result

(PHP 4, PHP 5)

sybase_free_result — 結果メモリを開放する

説明

bool ***sybase_free_result*** *(resource $result)*

sybase_free_result() は、スクリプト実行時に大量の メモリを使用することが懸念される場合にのみコールする必要があります。 すべての結果メモリは、スクリプト終了時に自動的に開放されます。 結果 *ID* を引数として ***sybase_freeresult()*** を コールすることが可能で、関連する結果メモリは開放されます。

パラメータ

result

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

sybase_get_last_message

(PHP 4, PHP 5)

sybase_get_last_message — サーバから直近のメッセージを返す

説明

string **sybase_get_last_message** (void)

sybase_get_last_message() はサーバからの直近のメッセージを返します。

返り値

メッセージを文字列で返します。

参考

- sybase_min_message_severity()

---

sybase_min_client_severity

(PHP 4, PHP 5)

sybase_min_client_severity — クライアントの severity の最小値を設定する

説明

void **sybase_min_client_severity** (int $severity)

sybase_min_client_severity() はクライアントの severity レベルの最小値を設定します。

パラメータ

severity

返り値

値を返しません。

注意

注意: この関数は、Sybase の CT ライブラリインターフェイスでのみ使用可能で、DB ライブラリでは使用できません。

参考

- sybase_min_server_severity()

---

sybase_min_error_severity

(PHP 4, PHP 5)

sybase_min_error_severity — エラーの severity の最小値を設定する

説明

void **sybase_min_error_severity** (int $severity)

sybase_min_error_severity() はエラー severity レベルの最小値を設定します。

パラメータ

severity

返り値

値を返しません。

注意

注意: この関数は、Sybase の CT ライブラリインターフェイスでのみ使用可能で、DB ライブラリでは使用できません。

参考

- sybase_min_message_severity()

---

sybase_min_message_severity

(PHP 4, PHP 5)

sybase_min_message_severity — *メッセージの severity の最小値を設定する*

説明

void ***sybase_min_message_severity*** *(int $severity)*

sybase_min_message_severity() *はメッセージの severity レベルの最小値を設定します。*

パラメータ

severity

返り値

値を返しません。

注意

> ***注意***: *この関数は、Sybase の DB ライブラリインターフェイスを使用する場合のみ利用可能で、 CT ライブラリでは利用できません。*

参考

- *sybase_min_error_severity()*

---

sybase_min_server_severity

(PHP 4, PHP 5)

sybase_min_server_severity — *サーバの severity の最小値を設定する*

説明

void ***sybase_min_server_severity*** *(int $severity)*

sybase_min_server_severity() *はサーバの severity レベルの最小値を設定します。*

パラメータ

severity

返り値

値を返しません。

注意

> ***注意***: *この関数は、Sybase の CT ライブラリインターフェイスでのみ使用可能で、DB ライブラリでは使用できません。*

参考

- *sybase_min_client_severity()*

---

sybase_num_fields

(PHP 4, PHP 5)

sybase_num_fields — *結果におけるフィールドの数を取得する*

説明

int ***sybase_num_fields*** *(resource $result)*

sybase_num_fields() *は、 結果セットにおけるフィールド数を返します。*

パラメータ

result

返り値

フィールドの数を返します。

参考

- *sybase_query()*
- *sybase_fetch_field()*
- *sybase_num_rows()*

---

sybase_num_rows

(PHP 4, PHP 5)

sybase_num_rows — 結果における行の数を取得する

説明

int **sybase_num_rows** (resource $result)

sybase_num_rows() は、結果セットの行数を返します。

パラメータ

result

返り値

結果セットの行の数を返します。

参考

- sybase_num_fields()
- sybase_query()
- sybase_fetch_row()

---

sybase_pconnect

(PHP 4, PHP 5)

sybase_pconnect — Sybase の持続的な接続をオープンする

説明

resource **sybase_pconnect** ([string $servername [, string $username [, string $password [, string $charset [, string $appname]]]]])

sybase_pconnect() は、2 つの違いを除いて sybase_connect() と全く同様に動作します。

まず、接続時にこの関数は最初同じホスト、ユーザ名、パスワードで オープンされた（持続的）リンクを見つけようとします。 見つかった場合、新しい接続をオープンする代わりにこれに関する ID が返されます。

第二に、SQL サーバーへの接続はスクリプト実行終了時にクローズされません。 代わりに、このリンクは将来的に使用するためにオープンされたままに なります (sybase_close() は、 **sybase_pconnect()** により確立されたリンクを 閉じません）。

このため、この型のリンクは'持続的'と呼ばれます。

パラメータ

servername

引数 servername は、'interfaces' ファイル中で定義された 有効なサーバ名である必要があります。

username

Sybase のユーザ名。

password

username のパスワード。

charset

接続に使用する文字セットを指定します。

appname

返り値

成功時に正の Sybase 持続リンク ID を、エラー時に **FALSE** を返します。

変更履歴

バージョン	説明
4.0.2	charset パラメータが追加されました。

参考

- sybase_connect()

---

sybase_query

(PHP 4, PHP 5)

sybase_query — Sybase クエリを送信する

説明

mixed **sybase_query** (string $query [, resource $link_identifier])

sybase_query()は、指定されたリンク ID が指すサーバ上で現在アクティブなデータベースにクエリを送信します。

パラメータ

query

link_identifier

リンク ID が指定されない場合、最後にオープンされたリンクが指定されたと 仮定されます。リンクがオープンされていない場合、この関数は、sybase_connect() がコールされた時と同様にリンクを 確立しようと試み、これを使用します。

返り値

成功時に正の Sybase 結果 ID を、エラー時に **FALSE** を返します。 クエリは成功したが、結果としてカラムを何も返さなかった場合は **TRUE** を返します。

参考

- sybase_select_db()
- sybase_connect()

---

sybase_result

(PHP 4, PHP 5)

sybase_result — 結果データを取得する

説明

string **sybase_result** (resource $result , int $row , mixed $field)

指定された Sybase 結果セットにおいて、指定した行および オフセットにおけるセルの内容を返します。

大きな結果セットを処理する際には、(以下に示す) 行全体を取り込む関数の一つを使用することを考慮すべきです。 これらの関数は、一回の関数コールで複数のセルの内容を返すので、 **sybase_result()**よりも著しく高速です。 また、field 引数において数値オフセットで指定するほうが、フィールド名やテーブル名.フィールド名で指定するよりも 著しく高速であるということにも注意してください。

推奨される高性能な代替案は次のようなものです。 sybase_fetch_row()、 sybase_fetch_array() および sybase_fetch_object()。

パラメータ

result

row

field

引数 field は、フィールドのオフセット、フィールド名、 またはテーブル名.フィールド名とすることができます。カラム名の エイリアスが定義されている場合('select foo as bar from...')、カラム名の 代わりにエイリアスを使用してください。

返り値

sybase_result() は、Sybase 結果セットからセルの内容を返します。

---

sybase_select_db

(PHP 4, PHP 5)

sybase_select_db — Sybase データベースを選択する

説明

bool **sybase_select_db** (string $database_name [, resource $link_identifier])

sybase_select_db() は、指定されたリンク ID が指す データベースをサーバ上の現在アクティブなデータベースに設定します。

その後の sybase_query() のコールは、 このアクティブなデータベースにおいて行われます。

パラメータ

database_name

link_identifier

リンク ID が指定されない場合、最後にオープンされたリンクが指定されたものとみなします。 リンクがオープンされていない場合、この関数は、sybase_connect() がコールされた時と同様に リンクを確立しようと試み、これを使用します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- sybase_connect()
- sybase_pconnect()
- sybase_query()

---

sybase_set_message_handler

(PHP 4 >= 4.3.0, PHP 5)

sybase_set_message_handler — サーバでメッセージが発生した際にコールされるハンドラを指定する

説明

bool **sybase_set_message_handler** (callback $handler [, resource $connection])

sybase_set_message_handler() は、 サーバからのメッセージを処理するユーザ関数を設定します。グローバル関数の名前、 あるいはオブジェクトへの参照とメソッド名を保持する配列を指定します。

パラメータ

handler

ハンドラは、以下の 5 つの引数をこの順に受け取ります。 number, severity, state, line number そして description。 最初の 4 つは整数値で、最後は文字列です。関数が **FALSE** を返した場合、PHP は通常のエラーメッセージを表示します。

connection

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
4.3.5	connection パラメータが追加されました。

例

Example#1 sybase_set_message_handler() コールバック関数

```
<?php
    function msg_handler($msgnumber, $severity, $state, $line, $text)
    {
        var_dump($msgnumber, $severity, $state, $line, $text);
    }

    sybase_set_message_handler('msg_handler');
?>
```

Example#2 クラスへの sybase_set_message_handler() コールバック

```
<?php
    class Sybase {
         function handler($msgnumber, $severity, $state, $line, $text)
        {
            var_dump($msgnumber, $severity, $state, $line, $text);
        }
    }

    $sybase= new Sybase();
    sybase_set_message_handler(array($sybase, 'handler'));
?>
```

Example#3 sybase_set_message_handler() が処理しないメッセージ

```
<?php
    // この関数から FALSE を返すことは、それをこの関数では処理しないことを
    // 意味します。エラーは警告として表示され、ハンドラが何も設定されていない
    // 場合と同様に処理できます。
    function msg_handler($msgnumber, $severity, $state, $line, $text)
    {
        if (257 == $msgnumber) {
            return false;
        }
        var_dump($msgnumber, $severity, $state, $line, $text);
    }

    sybase_set_message_handler('msg_handler');
?>
```

注意

注意: この関数は、Sybase の CT ライブラリインターフェイスでのみ使用可能で、DB ライブラリでは使用できません。

---

sybase_unbuffered_query

(PHP 4 >= 4.3.0, PHP 5)

sybase_unbuffered_query — Sybase クエリを送信し、ブロックしない

説明

resource **sybase_unbuffered_query** (string $query , resource $link_identifier [, bool $store_result])

sybase_unbuffered_query() は、指定したリンク ID に 関連付けられたサーバ上の現在アクティブなデータベースにクエリを送信します。 リンク ID が指定されていない場合は、直近にオープンされたリンクが 指定されたものと仮定します。リンクがオープンされていない場合は、sybase_connect() がコールされた場合と同様の手順で リンクのオープンを試み、それを使用します。

sybase_query() とは異なり、 **sybase_unbuffered_query()** は結果セットの最初の 行のみを読み込みます。それ以降の行は、必要に応じて sybase_fetch_array() またはそれに類する関数で 読み込みます。sybase_data_seek() は指定した 行を読み込みます。この挙動により、大きな結果セットを扱う際に よりよいパフォーマンスを確保できます。

sybase_num_rows() が正確な行数を返すのは、結果セット 全体が読み込まれた場合のみです。Sybase は行数を知ることができず、 クライアントの実装によって行数を計算しています。

> **注意**: 結果セットをすべて読み込む前に次のクエリを実行しようとした場合、 PHP は警告を発生し、未処理の結果をすべてキャンセルします。これを 避けるには、sybase_free_result() を使用します。 この関数は、バッファリングされていないクエリで未処理の結果をすべてキャンセルします。

パラメータ

query

link_identifier

store_result

結果セットをメモリ中に読み込む必要がないことを指示するため、 オプションの store_result に **FALSE** を 指定することが可能です。これによってメモリの使用量を抑えることができ、 大きな結果セットを扱う場合に有用です。

返り値

成功時に正の Sybase 結果 ID 、エラー時に **FALSE** を返します。

例

Example#1 sybase_unbuffered_query() の例

```
<?php

$dbh = sybase_connect('SYBASE', '', '');
$q = sybase_unbuffered_query('select firstname, lastname from huge_table', $dbh, false);
sybase_data_seek($q, 10000);
$i = 0;

while ($row = sybase_fetch_row($q)) {
    echo $row[0], ' ', $row[1], '<br />';
    if ($i++ > 40000) {
        break;
    }
}

sybase_free_result($q);
sybase_close($dbh);

?>
```

注意

> **注意**: この関数は、Sybase の CT ライブラリインターフェイスでのみ使用可能で、DB ライブラリでは使用できません。

参考

- sybase_query()

目次

- sybase_fetch_row — 行を配列として取得する
- sybase_field_seek — フィールドオフセットを設定する
- sybase_free_result — 結果メモリを開放する
- sybase_get_last_message — サーバから直近のメッセージを返す
- sybase_min_client_severity — クライアントの severity の最小値を設定する
- sybase_min_error_severity — エラーの severity の最小値を設定する
- sybase_min_message_severity — メッセージの severity の最小値を設定する
- sybase_min_server_severity — サーバの severity の最小値を設定する
- sybase_num_fields — 結果におけるフィールドの数を取得する
- sybase_num_rows — 結果における行の数を取得する
- sybase_pconnect — Sybase の持続的な接続をオープンする
- sybase_query — Sybase クエリを送信する
- sybase_result — 結果データを取得する
- sybase_select_db — Sybase データベースを選択する
- sybase_set_message_handler — サーバでメッセージが発生した際にコールされるハンドラを指定する
- sybase_unbuffered_query — Sybase クエリを送信し、ブロックしない

TCP ラッパ関数（TCP Wrappers）

導入

TCP ラッパは、古典的な Unix のメカニズムを提供します。これは リモートクライアントが指定した IP アドレスから接続できるかどうかを 調べるように設計されています。

インストール手順

Tcpwrap は、現在は PECL » http://pecl.php.net/package/tcpwrap で取得できます。

*nix 系のシステム上で » PEAR が有効な 場合は、pear インストーラを用いて tcpwrap 拡張モジュールをインストールする ことが可能です。その場合は以下のようなコマンドを入力します。 **pear -v install tcpwrap**

tar.gz パッケージをダウンロードし、tcpwrap を手動でインストールすることも 可能です。

Example#1 tcpwrap の手動インストール

```
gunzip  tcpwrap-xxx.tgz
tar  -xvf  tcpwrap-xxx.tar
cd  tcpwrap-xxx
phpize
./configure  &&  make  &&  make  install
```

---

tcpwrap_check

(PECL tcpwrap:0.1-1.0)

tcpwrap_check — tcpwrap のチェックを実行する

説明

bool **tcpwrap_check** (string $daemon , string $address [, string $user [, bool $nodns]])

この関数は /etc/hosts.allow および /etc/hosts.deny の内容をもとに、あるクライアントが daemon サービスへのアクセスを許可されているか 否かを調べます。

パラメータ

daemon

サービスの名前。

address

クライアントのリモートアドレス。IP アドレスあるいはドメイン名の どちらでも指定可能。

user

ユーザ名（オプション）。

nodns

address がドメイン名の場合、それを IP アドレスに名前解決するために DNS が使用されます。 nodns を **TRUE** に指定すると、この処理を 禁止します。

返り値

アクセスが許可されている場合に **TRUE** 、それ以外の場合に **FALSE** を返します。

例

Example#1 ローカルホストからの全接続を拒否する

/etc/hosts.deny ファイルに以下の行が含まれており、

```
php: 127.0.0.1
```

以下のようなコードを実行する

```
<?php
if (!tcpwrap_check('php', $_SERVER['REMOTE_ADDR'])) {
  die('You are not welcome here');
}
?>
```

参考

詳細な情報は、hosts_access(3) の man ページを参照ください。

Tidy 関数

導入

Tidyは、「Tidy HTML clean and repair utility」用のバインディングで、 HTML文書の誤りを直すだけでなく、操作することやドキュメントツリーを操作する ことも可能となります。

要件

Tidyを使用するには、libtidyがインストールされている必要があります。 libtidyは、tidyのホームページ » http://tidy.sourceforge.net/ で入手可能です。

インストール手順

Tidy は、現在、PHP 4.3.xおよびPHP 5用として、PECL拡張モジュールとして » http://pecl.php.net/package/tidy. から入手可能です。

注意: Tidy 1.0 はPHP 4.3.x専用で、Tidy 2.0はPHP 5は専用です。

» PEAR は、 *nix互換のシステムで利用可能で、tidy拡張モジュールをインストールする際に 以下のようにpearインストーラを使用することができます。 : **pecl install tidy**

次のようにマニュアルでtar.gzパッケージをダウンロードし、tidyをインストール することも可能です。

Example#1 PHP 4.3.x において tidy をマニュアルでインストールする

```
gunzip tidy-xxx.tgz
tar -xvf tidy-xxx.tar
cd tidy-xxx
phpize
./configure && make && make install
```

Windows ユーザは、拡張モジュールの DLL を » http://pecl4win.php.net/ext.php/php_tidy.dll からダウンロードすることが可能です。

PHP5 においては、--with-tidy オプションを使ってコンパイルするだけです。

実行時設定

php.ini の設定により動作が変化します。

Tidy設定オプション

名前	デフォルト	変更の可否	変更履歴
tidy.default_config	""	PHP_INI_SYSTEM	PHP 5.0.0 以降で利用可能です。
tidy.clean_output	"0"	PHP_INI_USER	PHP 5 では PHP_INI_PERDIR。PHP 5.0.0 以降で利用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

tidy.default_config string

tidy 設定ファイルへのデフォルトパス

tidy.clean_output boolean

Tidy による出力の修正のオン/オフを切り替える

警告

もし動的な画像のような HTML ではないコンテンツを生成する場合、 tidy.clean_output をオンにしないでください。

リソース型

リソース型は定義されていません。

定義済みクラス

tidyNode

メソッド

- tidyNode::getParent - カレントノードの親を返します。
- tidyNode->hasChildren - カレントノードが子を持つ場合、TRUE を返します。
- tidyNode->hasSiblings - カレントノードが兄弟を持つ場合、TRUE を返します。
- tidyNode->isAsp - カレントノードが ASP コードの場合、TRUE を返します。
- tidyNode->isComment - カレントノードが コメントの場合、TRUE を返します。
- tidyNode->isHtml - カレントノードが HTML の場合、TRUE を返します。
- tidyNode->isJste - カレントノードが JSTE の場合、TRUE を返します。
- tidyNode->isPhp - カレントノードが PHP の場合、TRUE を返します。
- tidyNode->isText - カレントノードがテキスト（マークアップでない）の場合、 TRUE を返します。

プロパティ

- value - ノードの値（HTML テキストなど）
- name - タグの名前（html, a, など）
- type - ノードの種類（上記の定数の 1 つ。例えば TIDY_NODETYPE_PHP）
- line* - ノードが開始する行
- column* - ノードが開始するカラム
- proprietary* - もしノードがプロプライエタリなタグの場合、 TRUE
- id - タグの ID（上記の定数の 1 つ。例えば TIDY_TAG_FRAME）
- attribute - カレントノードの属性の配列。存在しない場合 NULL
- child - 子の tidyNode の配列。存在しない場合 NULL

注意: * 付きのプロパティは PHP 5.1.0 からのみ利用可能です。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

それぞれの TIDY_TAG_XXX は HTML タグに相当します。 例えば、 TIDY_TAG_A は link タグに相当します。 それぞれの TIDY_ATTR_XXX は、HTML 属性に相当します。 例えば、 TIDY_ATTR_HREF は前の例の href 属性に相当します。

以下の定数が定義されています。

tidy タグ定数

定数
TIDY_TAG_UNKNOWN
TIDY_TAG_A
TIDY_TAG_ABBR
TIDY_TAG_ACRONYM
TIDY_TAG_ALIGN
TIDY_TAG_APPLET
TIDY_TAG_AREA
TIDY_TAG_B
TIDY_TAG_BASE
TIDY_TAG_BASEFONT
TIDY_TAG_BDO
TIDY_TAG_BGSOUND
TIDY_TAG_BIG
TIDY_TAG_BLINK
TIDY_TAG_BLOCKQUOTE
TIDY_TAG_BODY
TIDY_TAG_BR
TIDY_TAG_BUTTON

定数
TIDY_TAG_CAPTION
TIDY_TAG_CENTER
TIDY_TAG_CITE
TIDY_TAG_CODE
TIDY_TAG_COL
TIDY_TAG_COLGROUP
TIDY_TAG_COMMENT
TIDY_TAG_DD
TIDY_TAG_DEL
TIDY_TAG_DFN
TIDY_TAG_DIR
TIDY_TAG_DIV
TIDY_TAG_DL
TIDY_TAG_DT
TIDY_TAG_EM
TIDY_TAG_EMBED
TIDY_TAG_FIELDSET
TIDY_TAG_FONT
TIDY_TAG_FORM
TIDY_TAG_FRAME
TIDY_TAG_FRAMESET
TIDY_TAG_H1
TIDY_TAG_H2
TIDY_TAG_H3
TIDY_TAG_H4
TIDY_TAG_H5
TIDY_TAG_H6
TIDY_TAG_HEAD
TIDY_TAG_HR
TIDY_TAG_HTML
TIDY_TAG_I
TIDY_TAG_IFRAME
TIDY_TAG_ILAYER
TIDY_TAG_IMG
TIDY_TAG_INPUT
TIDY_TAG_INS
TIDY_TAG_ISINDEX
TIDY_TAG_KBD
TIDY_TAG_KEYGEN
TIDY_TAG_LABEL
TIDY_TAG_LAYER
TIDY_TAG_LEGEND
TIDY_TAG_LI
TIDY_TAG_LINK
TIDY_TAG_LISTING
TIDY_TAG_MAP
TIDY_TAG_MARQUEE
TIDY_TAG_MENU
TIDY_TAG_META

定数
TIDY_TAG_MULTICOL
TIDY_TAG_NOBR
TIDY_TAG_NOEMBED
TIDY_TAG_NOFRAMES
TIDY_TAG_NOLAYER
TIDY_TAG_NOSAVE
TIDY_TAG_NOSCRIPT
TIDY_TAG_OBJECT
TIDY_TAG_OL
TIDY_TAG_OPTGROUP
TIDY_TAG_OPTION
TIDY_TAG_P
TIDY_TAG_PARAM
TIDY_TAG_PLAINTEXT
TIDY_TAG_PRE
TIDY_TAG_Q
TIDY_TAG_RP
TIDY_TAG_RT
TIDY_TAG_RTC
TIDY_TAG_RUBY
TIDY_TAG_S
TIDY_TAG_SAMP
TIDY_TAG_SCRIPT
TIDY_TAG_SELECT
TIDY_TAG_SERVER
TIDY_TAG_SERVLET
TIDY_TAG_SMALL
TIDY_TAG_SPACER
TIDY_TAG_SPAN
TIDY_TAG_STRIKE
TIDY_TAG_STRONG
TIDY_TAG_STYLE
TIDY_TAG_SUB
TIDY_TAG_TABLE
TIDY_TAG_TBODY
TIDY_TAG_TD
TIDY_TAG_TEXTAREA
TIDY_TAG_TFOOT
TIDY_TAG_TH
TIDY_TAG_THEAD
TIDY_TAG_TITLE
TIDY_TAG_TR
TIDY_TAG_TR
TIDY_TAG_TT
TIDY_TAG_U
TIDY_TAG_UL
TIDY_TAG_VAR
TIDY_TAG_WBR
TIDY_TAG_XMP

tidy 属性定数

定数
TIDY_ATTR_UNKNOWN
TIDY_ATTR_ABBR
TIDY_ATTR_ACCEPT
TIDY_ATTR_ACCEPT_CHARSET
TIDY_ATTR_ACCESSKEY
TIDY_ATTR_ACTION
TIDY_ATTR_ADD_DATE
TIDY_ATTR_ALIGN
TIDY_ATTR_ALINK
TIDY_ATTR_ALT
TIDY_ATTR_ARCHIVE
TIDY_ATTR_AXIS
TIDY_ATTR_BACKGROUND
TIDY_ATTR_BGCOLOR
TIDY_ATTR_BGPROPERTIES
TIDY_ATTR_BORDER
TIDY_ATTR_BORDERCOLOR
TIDY_ATTR_BOTTOMMARGIN
TIDY_ATTR_CELLPADDING
TIDY_ATTR_CELLSPACING
TIDY_ATTR_CHAR
TIDY_ATTR_CHAROFF
TIDY_ATTR_CHARSET
TIDY_ATTR_CHECKED
TIDY_ATTR_CITE
TIDY_ATTR_CLASS
TIDY_ATTR_CLASSID
TIDY_ATTR_CLEAR
TIDY_ATTR_CODE
TIDY_ATTR_CODEBASE
TIDY_ATTR_CODETYPE
TIDY_ATTR_COLOR
TIDY_ATTR_COLS
TIDY_ATTR_COLSPAN
TIDY_ATTR_COMPACT
TIDY_ATTR_CONTENT
TIDY_ATTR_COORDS
TIDY_ATTR_DATA
TIDY_ATTR_DATAFLD
TIDY_ATTR_DATAPAGESIZE
TIDY_ATTR_DATASRC
TIDY_ATTR_DATETIME
TIDY_ATTR_DECLARE
TIDY_ATTR_DEFER
TIDY_ATTR_DIR
TIDY_ATTR_DISABLED
TIDY_ATTR_ENCODING
TIDY_ATTR_ENCTYPE
TIDY_ATTR_FACE

定数
TIDY_ATTR_FOR
TIDY_ATTR_FRAME
TIDY_ATTR_FRAMEBORDER
TIDY_ATTR_FRAMESPACING
TIDY_ATTR_GRIDX
TIDY_ATTR_GRIDY
TIDY_ATTR_HEADERS
TIDY_ATTR_HEIGHT
TIDY_ATTR_HREF
TIDY_ATTR_HREFLANG
TIDY_ATTR_HSPACE
TIDY_ATTR_HTTP_EQUIV
TIDY_ATTR_ID
TIDY_ATTR_ISMAP
TIDY_ATTR_LABEL
TIDY_ATTR_LANG
TIDY_ATTR_LANGUAGE
TIDY_ATTR_LAST_MODIFIED
TIDY_ATTR_LAST_VISIT
TIDY_ATTR_LEFTMARGIN
TIDY_ATTR_LINK
TIDY_ATTR_LONGDESC
TIDY_ATTR_LOWSRC
TIDY_ATTR_MARGINHEIGHT
TIDY_ATTR_MARGINWIDTH
TIDY_ATTR_MAXLENGTH
TIDY_ATTR_MEDIA
TIDY_ATTR_METHOD
TIDY_ATTR_MULTIPLE
TIDY_ATTR_NAME
TIDY_ATTR_NOHREF
TIDY_ATTR_NORESIZE
TIDY_ATTR_NOSHADE
TIDY_ATTR_NOWRAP
TIDY_ATTR_OBJECT
TIDY_ATTR_OnAFTERUPDATE
TIDY_ATTR_OnBEFOREUNLOAD
TIDY_ATTR_OnBEFOREUPDATE
TIDY_ATTR_OnBLUR
TIDY_ATTR_OnCHANGE
TIDY_ATTR_OnCLICK
TIDY_ATTR_OnDATAAVAILABLE
TIDY_ATTR_OnDATASETCHANGED
TIDY_ATTR_OnDATASETCOMPLETE
TIDY_ATTR_OnDBLCLICK
TIDY_ATTR_OnERRORUPDATE
TIDY_ATTR_OnFOCUS
TIDY_ATTR_OnKEYDOWN
TIDY_ATTR_OnKEYPRESS

定数
TIDY_ATTR_OnKEYUP
TIDY_ATTR_OnLOAD
TIDY_ATTR_OnMOUSEDOWN
TIDY_ATTR_OnMOUSEMOVE
TIDY_ATTR_OnMOUSEOUT
TIDY_ATTR_OnMOUSEOVER
TIDY_ATTR_OnMOUSEUP
TIDY_ATTR_OnRESET
TIDY_ATTR_OnROWENTER
TIDY_ATTR_OnROWEXIT
TIDY_ATTR_OnSELECT
TIDY_ATTR_OnSUBMIT
TIDY_ATTR_OnUNLOAD
TIDY_ATTR_PROFILE
TIDY_ATTR_PROMPT
TIDY_ATTR_RBSPAN
TIDY_ATTR_READONLY
TIDY_ATTR_REL
TIDY_ATTR_REV
TIDY_ATTR_RIGHTMARGIN
TIDY_ATTR_ROWS
TIDY_ATTR_ROWSPAN
TIDY_ATTR_RULES
TIDY_ATTR_SCHEME
TIDY_ATTR_SCOPE
TIDY_ATTR_SCROLLING
TIDY_ATTR_SELECTED
TIDY_ATTR_SHAPE
TIDY_ATTR_SHOWGRID
TIDY_ATTR_SHOWGRIDX
TIDY_ATTR_SHOWGRIDY
TIDY_ATTR_SIZE
TIDY_ATTR_SPAN
TIDY_ATTR_SRC
TIDY_ATTR_STANDBY
TIDY_ATTR_START
TIDY_ATTR_STYLE
TIDY_ATTR_SUMMARY
TIDY_ATTR_TABINDEX
TIDY_ATTR_TARGET
TIDY_ATTR_TEXT
TIDY_ATTR_TITLE
TIDY_ATTR_TOPMARGIN
TIDY_ATTR_TYPE
TIDY_ATTR_USEMAP
TIDY_ATTR_VALIGN
TIDY_ATTR_VALUE
TIDY_ATTR_VALUETYPE
TIDY_ATTR_VERSION

定数
TIDY_ATTR_VLINK
TIDY_ATTR_VSPACE
TIDY_ATTR_WIDTH
TIDY_ATTR_WRAP
TIDY_ATTR_XML_LANG
TIDY_ATTR_XML_SPACE
TIDY_ATTR_XMLNS

tidy ノード型定数

定数	説明
TIDY_NODETYPE_ROOT	ルートノード
TIDY_NODETYPE_DOCTYPE	文章型
TIDY_NODETYPE_COMMENT	HTML コメント
TIDY_NODETYPE_PROCINS	処理命令
TIDY_NODETYPE_TEXT	テキスト
TIDY_NODETYPE_START	開始タグ
TIDY_NODETYPE_END	終了タグ
TIDY_NODETYPE_STARTEND	空タグ
TIDY_NODETYPE_CDATA	CDATA
TIDY_NODETYPE_SECTION	XML セクション
TIDY_NODETYPE_ASP	ASP コード
TIDY_NODETYPE_JSTE	JSTE コード
TIDY_NODETYPE_PHP	PHP コード
TIDY_NODETYPE_XMLDECL	XML 宣言

例

このシンプルな例は、Tidy の基本的な使い方を示しています。

Example#2 Tidy の基本的な使用法

```
<?php
ob_start();
?>
<html>a  html  document</html>
<?php
$html = ob_get_clean();

// とある設定
$config = array(
            'indent'          => true,
            'output-xhtml'    => true,
            'wrap'            => 200);

// Tidy
$tidy = new tidy;
$tidy->parseString($html, $config, 'utf8');
$tidy->cleanRepair();

// 出力
echo $tidy;
?>
```

---

ob_tidyhandler

(PHP 5)

ob_tidyhandler — バッファを修正するための ob_start コールバック関数

説明

string **ob_tidyhandler** (string $input [, int $mode])

ob_tidyhandler() は、バッファを修正するための ob_start() として使用されることを意図しています。

Example#1 ob_tidyhandler() の例

```
<?php
ob_start('ob_tidyhandler');
```

```
echo '<p>test</i>';
?>
```

上の例の出力は以下となります。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>test</p>
</body>
</html>
```

ob_start() も参照ください。

---

tidy_access_count

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_access_count — 指定したドキュメントについて発生したTidyアクセシビリティ警告の数を返す

説明

int **tidy_access_count** (tidy $object)

tidy_access_count() は、 指定したドキュメントについて発生した Tidy アクセシビリティ警告の数を返します。

> **注意**: TidyLib の設計のため、**tidy_access_count()** の前に tidy_diagnose() をコールしなければなりません。 さもないと、常に 0 が返されます。 また、accessibility-check オプションを有効にしなければなりません。

Example#1 tidy_access_count() の例

```
<?php

$html ='<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html><head><title>Title</title></head>
<body>

<p><img src="img.png"></p>

</body></html>';


// アクセシビリティチェックのレベルを選択する: 1, 2 あるいは 3
$config = array('accessibility-check' => 3);

$tidy = new tidy();
$tidy->parseString($html, $config);
$tidy->CleanRepair();

/* これをコールすることを忘れないように! */
$tidy->diagnose();

echo tidy_access_count($tidy); //5

?>
```

tidy_error_count(), tidy_warning_count() も参照ください。

---

tidy_clean_repair

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_clean_repair — パースされたマークアップに設定に基く誤りの修正を行う

説明

bool **tidy_clean_repair** (tidy $object)

Object oriented style:

bool **tidy->cleanRepair** (void)

この関数は、与えられた Tidy オブジェクト object を修正します。

Example#1 tidy_clean_repair() の例

```
<?php
$html = '<p>test</I>';

$tidy = tidy_parse_string($html);
tidy_clean_repair($tidy);

echo $tidy;
?>
```

上の例の出力は以下となります。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>test</p>
</body>
</html>
```

tidy_repair_file(), tidy_repair_string() も参照ください。

---

tidy_config_count

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_config_count — 指定したドキュメントについて発生した Tidy 設定エラーの数を返す

説明

int **tidy_config_count** (tidy $object)

tidy_config_count() は、 指定した Tidy オブジェクト object について発生した Tidy エラーの数を返します。

Example#1 tidy_config_count() の例

```
<?php
$html = '<p>test</I>';

$config = array('doctype' => 'bogus');

$tidy = tidy_parse_string($html, $config);

/* 'bogus' は有効な文章型でないので、1 を出力する */
echo tidy_config_count($tidy);
?>
```

---

tidy::__construct

(No version information available, might be only in CVS)

tidy::__construct — 新規 Tidy オブジェクトを生成する

説明

tidy **tidy::__construct** ([string $filename [, mixed $config [, string $encoding [, bool $use_include_path]]]])

tidy::__construct() constructs a new tidy object.

もし filename パラメータが与えられた場合、 この関数はファイルを読み込み、tidy_parse_file() のように実行してファイルに基づいたオブジェクトを初期化します。

config パラメータは、配列または文字列として指定することができます。文字列として指定した場合には設定ファイルの名前として解釈され、そうでない場合はオプション自体として解釈されます。各オプションに関する説明については、 » http://tidy.sourceforge.net/docs/quickref.html を参照してください。

encoding パラメータは、文書を入力/出力する際のエンコーディングを設定します。encoding には次の値を使用可能です。 ascii, latin1, raw, utf8, iso2022, mac, win1252, utf16le, utf16be, utf16, big5, shiftjis

Example#1 tidy::__construct() の例

```
<?php

$html = <<< HTML

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head><title>title</title></head>
<body>
<p>paragraph <bt />
text</p>
</body></html>

HTML;

$tidy = new tidy;
$tidy->parseString($html);

$tidy->CleanRepair();

if ($tidy->errorBuffer) {
     echo "The following errors were detected:¥n";
     echo $tidy->errorBuffer;
}
```

```
?>
```

上の例の出力は以下となります。

```
The following errors were detected:
line 8 column 14 - Error: <bt> is not recognized!
line 8 column 14 - Warning: discarding unexpected <bt>
```

tidy_parse_file(), tidy_parse_string() も参照ください。

---

tidy_diagnose

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_diagnose — パース、修正されたマークアップの診断を行う

説明

手続き言語型スタイル:

```
bool tidy_diagnose ( tidy $object )
```

オブジェクト指向言語型スタイル:

```
bool tidy->diagnose ( void )
```

tidy_diagnose() は、 与えられて Tidy オブジェクト object に対して診断テストを実行し、 エラーバッファにドキュメントについての情報を追加します。

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

Example#1 tidy_diagnose() の例

```
<?php

$html = <<< HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<p>paragraph</p>
HTML;

$tidy = tidy_parse_string($html);
$tidy->CleanRepair();

// 2 つの出力の違いに注意
echo tidy_get_error_buffer($tidy) . "¥n";

$tidy->diagnose();
echo tidy_get_error_buffer($tidy);

?>
```

上の例の出力は以下となります。

```
line 5 column 1 - Warning: <p> isn't allowed in <head> elements
line 5 column 1 - Warning: inserting missing 'title' element

line 5 column 1 - Warning: <p> isn't allowed in <head> elements
line 5 column 1 - Warning: inserting missing 'title' element
Info: Doctype given is "-//W3C//DTD XHTML 1.0 Strict//EN"
Info: Document content looks like XHTML 1.0 Strict
2 warnings, 0 errors were found!
```

tidy_get_error_buffer() も参照ください。

---

tidy_error_count

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_error_count — 指定したドキュメントについて発生した Tidy エラーの数を返す

説明

```
int tidy_error_count ( tidy $object )
```

tidy_error_count() は、 指定したドキュメントについて発生した Tidy エラーの数を返します。

Example#1 tidy_error_count() の例

```
<?php
$html = '<p>test</i>
<bogustag>bogus</bogustag>';
```

```
$tidy = tidy_parse_string($html);

echo tidy_error_count($tidy) . "¥n"; //1

echo $tidy->errorBuffer;
?>
```

上の例の出力は以下となります。

```
1
line 1 column 1 - Warning: missing <!DOCTYPE> declaration
line 1 column 8 - Warning: discarding unexpected </i>
line 2 column 1 - Error: <bogustag> is not recognized!
line 2 column 1 - Warning: discarding unexpected <bogustag>
line 2 column 16 - Warning: discarding unexpected </bogustag>
line 1 column 1 - Warning: inserting missing 'title' element
```

tidy_access_count(), tidy_warning_count() も参照ください。

---

tidy_get_body

(PHP 5, PECL tidy:0.5.2-1.0)

tidy_get_body — Tidy パースツリーの <body> タグから始まる tidyNode オブジェクトを返す

説明

手続き言語型スタイル:

tidyNode **tidy_get_body** (tidy $object)

オブジェクト指向言語型スタイル:

tidyNode **tidy->body** (void)

この関数は Tidy パースツリーの <body> タグから始まる tidyNode オブジェクトを返します。

Example#1 tidy_get_body() の例

```
<?php
$html = '
<html>
   <head>
      <title>test</title>
   </head>
   <body>
      <p>paragraph</p>
   </body>
</html>';

$tidy = tidy_parse_string($html);

$body = tidy_get_body($tidy);
echo $body->value;
?>
```

上の例の出力は以下となります。

```
<body>
<p>paragraph</p>
</body>
```

> **注意**: この関数は、Zend Engine 2、つまり、PHP >= 5.0.0でのみ利用可能です。

tidy_get_head(), tidy_get_html() も参照ください。

---

tidy_get_config

(PHP 5, PECL tidy:0.7-1.2)

tidy_get_config — 現在の Tidy の設定を取得する

説明

手続き言語型スタイル:

array **tidy_get_config** (tidy $object)

オブジェクト指向言語型スタイル:

array **tidy->getConfig** (void)

tidy_get_config() は 与えられた Tidy オブジェクト object で使用されている設定オプションを配列で返します。

それぞれのオプションの説明については、» http://tidy.sourceforge.net/docs/quickref.html を参照ください。

Example#1 tidy_get_config() の例

```
<?php
$html = '<p>test</p>';
$config = array('indent' => TRUE,
                'output-xhtml' => TRUE,
                'wrap' => 200);

$tidy = tidy_parse_string($html, $config);

print_r(tidy_get_config($tidy));
?>
```

上の例の出力は以下となります。

```
Array
(
     [indent-spaces] => 2
     [wrap] => 200
     [tab-size] => 8
     [char-encoding] => 1
     [input-encoding] => 3
     [output-encoding] => 1
     [newline] => 1
     [doctype-mode] => 1
     [doctype] =>
     [repeated-attributes] => 1
     [alt-text] =>
     [slide-style] =>
     [error-file] =>
     [output-file] =>
     [write-back] =>
     [markup] => 1
     [show-warnings] => 1
     [quiet] =>
     [indent] => 1
     [hide-endtags] =>
     [input-xml] =>
     [output-xml] => 1
     [output-xhtml] => 1
     [output-html] =>
     [add-xml-decl] =>
     [uppercase-tags] =>
     [uppercase-attributes] =>
     [bare] =>
     [clean] =>
     [logical-emphasis] =>
     [drop-proprietary-attributes] =>
     [drop-font-tags] =>
     [drop-empty-paras] => 1
     [fix-bad-comments] => 1
     [break-before-br] =>
     [split] =>
     [numeric-entities] =>
     [quote-marks] =>
     [quote-nbsp] => 1
     [quote-ampersand] => 1
     [wrap-attributes] =>
     [wrap-script-literals] =>
     [wrap-sections] => 1
     [wrap-asp] => 1
     [wrap-jste] => 1
     [wrap-php] => 1
     [fix-backslash] => 1
     [indent-attributes] =>
     [assume-xml-procins] =>
     [add-xml-space] =>
     [enclose-text] =>
     [enclose-block-text] =>
     [keep-time] =>
     [word-2000] =>
     [tidy-mark] =>
     [gnu-emacs] =>
     [gnu-emacs-file] =>
     [literal-attributes] =>
     [show-body-only] =>
     [fix-uri] => 1
     [lower-literals] => 1
     [hide-comments] =>
     [indent-cdata] =>
     [force-output] => 1
     [show-errors] => 6
     [ascii-chars] => 1
     [join-classes] =>
     [join-styles] => 1
     [escape-cdata] =>
     [language] =>
     [ncr] => 1
     [output-bom] => 2
     [replace-color] =>
     [css-prefix] =>
     [new-inline-tags] =>
     [new-blocklevel-tags] =>
     [new-empty-tags] =>
     [new-pre-tags] =>
     [accessibility-check] => 0
     [vertical-space] =>
     [punctuation-wrap] =>
     [merge-divs] => 1
)
```

tidy_reset_config(), tidy_save_config() も参照ください。

---

tidy_get_error_buffer

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_get_error_buffer — 指定したドキュメントのパースで発生した警告とエラーを返す

説明

手続き言語型スタイル:

string **tidy_get_error_buffer** (tidy $object)

オブジェクト指向言語型スタイル (property):

tidy
string$errorBuffer;

tidy_get_error_buffer() は、 指定したドキュメントのパースで発生した警告とエラーを返します。

Example#1 tidy_get_error_buffer() の例

```
<?php
$html = '<p>paragraph</p>';

$tidy = tidy_parse_string($html);

echo tidy_get_error_buffer($tidy);
/* オブジェクト指向の場合 */
echo $tidy->errorBuffer;
?>
```

上の例の出力は以下となります。

```
line 1 column 1 - Warning: missing <!DOCTYPE> declaration
line 1 column 1 - Warning: inserting missing 'title' element
```

tidy_access_count(), tidy_error_count(), tidy_warning_count() も参照ください。

---

tidy_get_head

(PHP 5, PECL tidy:0.5.2-1.0)

tidy_get_head — Tidy パースツリーの <head> タグから始まる tidyNode オブジェクトを返す

説明

手続き言語型スタイル:

tidyNode **tidy_get_head** (tidy $object)

オブジェクト指向言語型スタイル:

tidyNode **tidy->head** (void)

この関数は Tidy パースツリーの <head> タグから始まる tidyNode オブジェクトを返します。

Example#1 tidy_get_head() の例

```
<?php
$html = '
<html>
   <head>
      <title>test</title>
   </head>
   <body>
      <p>paragraph</p>
   </body>
</html>';

$tidy = tidy_parse_string($html);

$head = tidy_get_head($tidy);
echo $head->value;
?>
```

上の例の出力は以下となります。

```
<head>
<title>test</title>
</head>
```

注意: この関数は、Zend Engine 2、つまり、PHP >= 5.0.0でのみ利用可能です。

tidy_get_body(), tidy_get_html() も参照ください。

---

tidy_get_html_ver

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_get_html_ver — 指定したドキュメントで検出された HTML バージョンを取得する

説明

手続き言語型スタイル:

int **tidy_get_html_ver** (tidy $object)

オブジェクト指向言語型スタイル:

int **tidy->getHtmlVer** (void)

tidy_get_html_ver() は、 指定した Tidy オブジェクト object で検出された HTML バージョンを返します。

警告

この関数はまだ Tidylib 自身に実装されていません。 なので、常に 0 を返します。

---

tidy_get_html

(PHP 5, PECL tidy:0.5.2-1.0)

tidy_get_html — Tidy パースツリーの <html> タグから始まる tidyNode オブジェクトを返す

説明

手続き言語型スタイル:

tidyNode **tidy_get_html** (tidy $object)

オブジェクト指向言語型スタイル:

tidyNode **tidy->html** (void)

この関数は Tidy パースツリーの <html> タグから始まる tidyNode オブジェクトを返します。

Example#1 tidy_get_html() の例

```
<?php
$html = '
<html>
   <head>
      <title>test</title>
   </head>
   <body>
      <p>paragraph</p>
   </body>
</html>';

$tidy = tidy_parse_string($html);

$html = tidy_get_html($tidy);
echo $html->value;
?>
```

上の例の出力は以下となります。

```
<html>
<head>
<title>test</title>
</head>
<body>
<p>paragraph</p>
</body>
</html>
```

注意: この関数は、Zend Engine 2、つまり、PHP >= 5.0.0でのみ利用可能です。

tidy_get_body(), tidy_get_head() も参照ください。

---

tidy_get_opt_doc

(PHP 5 >= 5.1.0)

tidy_get_opt_doc — 与えられたオプション名に対するドキュメントを返す

説明

Procedural style:

string **tidy_get_opt_doc** (tidy $object , string $optname)

Object oriented style:

string **tidy->getOptDoc** (string $optname)

tidy_get_opt_doc() は、 与えられたオプション名に対するドキュメントを返します。

注意: この関数を有効にするには、少なくとも2005年4月25日以降の libtidy が必要です。

パラメータ

object

Tidy オブジェクトを指定します。

optname

オプション名を指定します。

返り値

オプションが存在し、かつドキュメントが利用可能であれば、文字列を返します。 その他の場合は **FALSE** を返します。

例

Example#1 全てのオプションについてのドキュメントとデフォルト値をプリントする

```
<?php

$tidy = new tidy;
$config = $tidy->getConfig();

ksort($config);

foreach ($config as $opt => $val) {

    if (!$doc = $tidy->getOptDoc($opt))
        $doc = 'no documentation available!';

    $val = ($tidy->getOpt($opt) === true)  ? 'true'  : $val;
    $val = ($tidy->getOpt($opt) === false) ? 'false' : $val;

    echo "<p><b>$opt</b> (default: '$val')<br />".
         "$doc</p><hr />¥n";
}

?>
```

参考

- tidy_get_config()
- tidy_getopt()

---

tidy_get_output

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_get_output — パースされた Tidy マークアップを表す文字列を返す

説明

string **tidy_get_output** (tidy $object)

tidy_get_output() は、修正された HTML を文字列で返します。

Example#1 tidy_get_output() の例

```
<?php

$html = '<p>paragraph</i>';
$tidy = tidy_parse_string($html);

$tidy->CleanRepair();

echo tidy_get_output($tidy);
?>
```

上の例の出力は以下となります。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title></title>
</head>
<body>
<p>paragraph</p>
```

```
</body>
</html>
```

---

tidy_get_release

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_get_release — Tidy ライブラリのリリース日 (バージョン) を取得する

説明

手続き言語型スタイル:

```
string tidy_get_release ( void )
```

オブジェクト指向言語型スタイル:

```
string tidy->getRelease ( void )
```

この関数は、Tidy ライブラリのリリース日を文字列で返します。

---

tidy_get_root

(PHP 5, PECL tidy:0.5.2-1.0)

tidy_get_root — Tidy パースツリーのルートを表す tidyNode を返す

説明

手続き言語型スタイル:

```
tidyNode tidy_get_root ( tidy $object )
```

オブジェクト指向言語型スタイル:

```
tidyNode tidy->root ( void )
```

Tidy パースツリーのルートを表す tidyNode を返します。

Example#1 ノードをダンプする

```
<?php

$html = <<< HTML
<html><body>

<p>paragraph</p>
<br/>

</body></html>
HTML;

$tidy = tidy_parse_string($html);
dump_nodes($tidy->root(), 1);


function dump_nodes($node, $indent) {

    if($node->hasChildren()) {
        foreach($node->child as $child) {
            echo str_repeat('.', $indent*2) . ($child->name ? $child->name : '"'.$child->value.'"'). "¥n";

            dump_nodes($child, $indent+1);
        }
    }
}
?>
```

上の例の出力は以下となります。

```
..html
....head
......title
....body
......p
........"paragraph"
......br
```

注意: この関数は、Zend Engine 2、つまり、PHP >= 5.0.0でのみ利用可能です。

---

tidy_get_status

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_get_status — 指定したドキュメントのステータスを取得する

説明

手続き言語型スタイル:

```
int tidy_get_status ( tidy $object )
```

オブジェクト指向言語型スタイル:

```
int tidy->getStatus ( void )
```

tidy_get_status() は、 指定した Tidy オブジェクト object のステータスを返します。エラー/警告が発生しなかった場合は 0、 警告やアクセイシビリティエラーの場合は 1、 エラーの場合は 2 を返します。

Example#1 tidy_get_status() の例

```
<?php
$html = '<p>paragraph</i>';
$tidy = tidy_parse_string($html);

$html2 = '<bogus>test</bogus>';
$tidy2 = tidy_parse_string($html2);

echo tidy_get_status($tidy); //1

echo tidy_get_status($tidy2); //2
?>
```

---

tidy_getopt

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_getopt — Tidy ドキュメントについて指定した設定オプションの値を返す

説明

手続き言語型スタイル:

```
mixed tidy_getopt ( tidy $object , string $option )
```

オブジェクト指向言語型スタイル:

```
mixed tidy->getOpt ( string $option )
```

tidy_getopt() は、 Tidy オブジェクト object について指定した設定オプション option の値を返します。 返される型は、設定オプション option の型に依存します。 それぞれの設定オプションと型の一覧は、» http://tidy.sourceforge.net/docs/quickref.html にあります。

Example#1 tidy_getopt() の例

```
<?php

$html ='<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html><head><title>Title</title></head>
<body>

<p><img src="img.png"></p>

</body></html>';

$config = array('accessibility-check' => 3,
                'alt-text' => 'some text');

$tidy = new tidy();
$tidy->parseString($html, $config);


var_dump($tidy->getOpt('accessibility-check')); //integer
var_dump($tidy->getOpt('lower-literals')); //boolean
var_dump($tidy->getOpt('alt-text')); //string

?>
```

上の例の出力は以下となります。

```
int(3)
bool(true)
string(9) "some text"
```

---

tidy_is_xhtml

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_is_xhtml — ドキュメントが XHTML ドキュメントかどうかを示す

説明

手続き言語型スタイル:

bool **tidy_is_xhtml** (tidy $object)

オブジェクト指向言語型スタイル:

bool **tidy->isXhtml** (void)

指定した Tidy オブジェクト object が XHTML ドキュメントの場合、この関数は **TRUE** を返します。 その他の場合は **FALSE** を返します。

警告

この関数はまだ Tidylib 自身に実装されていません。 なので、常に **FALSE** を返します。

---

tidy_is_xml

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_is_xml — ドキュメントが一般的な XML ドキュメント (非 HTML/XHTML) かどうかを示す

説明

手続き言語型スタイル:

bool **tidy_is_xml** (tidy $object)

オブジェクト指向言語型スタイル:

bool **tidy->isXml** (void)

指定した Tidy オブジェクト object が一般的な XML ドキュメント (非 HTML/XHTML) の場合、 この関数は **TRUE** を返します。 その他の場合は **FALSE** を返します。

警告

この関数はまだ Tidylib 自身に実装されていません。 なので、常に **FALSE** を返します。

---

tidy_load_config

(PECL tidy:0.5.2-1.2)

tidy_load_config — 指定したエンコーディングで ASCII コードの Tidy 設定ファイルをロードする

説明

void **tidy_load_config** (string $filename , string $encoding)

この関数は、指定したエンコーディング encoding で Tidy 設定ファイルをロードします。

注意: この関数は、Tidy 1.0 でのみ利用可能です。この関数は、Tidy 2.0 では古い関数として削除されているためです。

---

tidy_node->get_attr

(PECL tidy:0.7-1.0)

tidy_node->get_attr — 指定された属性 ID を持つ属性を返す

説明

tidy_attr **tidy_node->get_attr** (int $attrib_id)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

tidy_node->get_nodes

(PECL tidy:0.7-1.0)

tidy_node->get_nodes — 指定された ID を持つノード以下のノードの配列を返す

説明

array **tidy_node->get_nodes** (int $node_id)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

tidy_node->next

(PECL tidy:0.7-1.0)

tidy_node->next — このノードの次の兄弟を返す

説明

tidy_node **tidy_node->next** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

tidy_node->prev

(PECL tidy:0.7-1.0)

tidy_node->prev — このノードの前の兄弟を返す

説明

tidy_node **tidy_node->prev** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

tidy_parse_file

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_parse_file — ファイルまたは URI にあるマークアップをパースする

説明

手続き言語型スタイル:

tidy **tidy_parse_file** (string $filename [, mixed $config [, string $encoding [, bool $use_include_path]]])

オブジェクト指向言語型スタイル:

bool **tidy->parseFile** (string $filename [, mixed $config [, string $encoding [, bool $use_include_path]]])

この関数は与えられたファイルをパースします。

config パラメータは、配列または文字列として指定することができます。文字列として指定した場合には設定ファイルの名前として解釈され、そうでない場合はオプション自体として解釈されます。各オプションに関する説明については、 » http://tidy.sourceforge.net/docs/quickref.html を参照してください。

encoding パラメータは、文書を入力/出力する際のエンコーディングを設定します。encoding には次の値を使用可能です。 ascii, latin1, raw, utf8, iso2022, mac, win1252, utf16le, utf16be, utf16, big5, shiftjis

Example#1 tidy_parse_file() の例

```
<?php
$tidy = tidy_parse_file('file.html');

$tidy->cleanRepair();

if(!empty($tidy->errorBuffer)) {
     echo "The following errors or warnings occured:¥n";
     echo $tidy->errorBuffer;
}
?>
```

注意: オプションのパラメータ config_options と enconding は Tidy 2.0 で追加されました。

tidy_parse_string(), tidy_repair_file(), tidy_repair_string() も参照ください。

---

tidy_parse_string

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_parse_string — 文字列にストアされたドキュメントをパースする

説明

手続き言語型スタイル:

tidy **tidy_parse_string** (string $input [, mixed $config [, string $encoding]])

オブジェクト指向言語型スタイル:

bool **tidy->parseString** (string $input [, mixed $config [, string $encoding]])

tidy_parse_string() は、 文字列にストアされたドキュメントをパースします。

config パラメータは、配列または文字列として指定することができます。文字列として指定した場合には設定ファイルの名前として解釈され、そうでない場合はオプション自体として解釈されます。各オプションに関する説明については、 » http://tidy.sourceforge.net/docs/quickref.html を参照してください。

encoding パラメータは、文書を入力/出力する際のエンコーディングを設定します。encoding には次の値を使用可能です。 ascii, latin1, raw, utf8, iso2022, mac, win1252, utf16le, utf16be, utf16, big5, shiftjis

Example#1 tidy_parse_string() の例

```
<?php
ob_start();
?>

<html>
   <head>
     <title>test</title>
   </head>
   <body>
     <p>error<br>another line</i>
   </body>
</html>

<?php

$buffer = ob_get_clean();
$config = array('indent' => TRUE,
                'output-xhtml' => TRUE,
                'wrap' => 200);

$tidy = tidy_parse_string($buffer, $config, 'UTF8');

$tidy->cleanRepair();
echo $tidy;
?>
```

上の例の出力は以下となります。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
     <title>
       test
     </title>
   </head>
   <body>
     <p>
       error<br />
       another line
     </p>
   </body>
</html>
```

注意: オプションのパラメータ config_options と enconding は Tidy 2.0 で追加されました。

tidy_parse_file(), tidy_repair_file(), tidy_repair_string() も参照ください。

---

tidy_repair_file

(PHP 5, PECL tidy:0.7-1.2)

tidy_repair_file — ファイルを修正し、それを文字列として返す

説明

string **tidy_repair_file** (string $filename [, mixed $config [, string $encoding [, bool $use_include_path]]])

この関数は、与えられたファイルを修正し、それを文字列として返します。

config パラメータは、配列または文字列として指定することができます。文字列として指定した場合には設定ファイルの名前として解釈され、そうでない場合はオプション自体として解釈されます。各オプションに関する説明については、 » http://tidy.sourceforge.net/docs/quickref.html を参照してください。

encoding パラメータは、文書を入力/出力する際のエンコーディングを設定します。encoding には次の値を使用可能です。 ascii, latin1, raw, utf8, iso2022, mac, win1252, utf16le, utf16be, utf16, big5, shiftjis

Example#1 tidy_repair_file() の例

```
<?php
$file = 'file.html';

$repaired = tidy_repair_file($file);
rename($file, $file . '.bak');

file_put_contents($file, $repaired);
?>
```

注意: オプションのパラメータ config_options と enconding は Tidy 2.0 で追加されました。

tidy_parse_file(), tidy_parse_string(), tidy_repair_string() も参照ください。

tidy_repair_string

(PHP 5, PECL tidy:0.7-1.2)

tidy_repair_string — 別途提供される設定ファイルを使用して文字列を修正する

説明

string **tidy_repair_string** (string $data [, mixed $config [, string $encoding]])

この関数は、与えられた文字列を修正します。 This function repairs the given string.

config パラメータは、配列または文字列として指定することができます。文字列として指定した場合には設定ファイルの名前として解釈され、そうでない場合はオプション自体として解釈されます。各オプションに関する説明については、 » http://tidy.sourceforge.net/docs/quickref.html を参照してください。

encoding パラメータは、文書を入力/出力する際のエンコーディングを設定します。encoding には次の値を使用可能です。 ascii, latin1, raw, utf8, iso2022, mac, win1252, utf16le, utf16be, utf16, big5, shiftjis

Example#1 tidy_repair_string() の例

```
<?php
ob_start();
?>

<html>
   <head>
      <title>test</title>
   </head>
   <body>
      <p>error</i>
   </body>
</html>

<?php

$buffer = ob_get_clean();
$tidy = tidy_repair_string($buffer);

echo $tidy;
?>
```

上の例の出力は以下となります。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<title>test</title>
</head>
<body>
<p>error</p>
</body>
</html>
```

注意: オプションのパラメータ config_options と enconding は Tidy 2.0 で追加されました。

tidy_parse_file(), tidy_parse_string(), tidy_repair_file() も参照ください。

tidy_reset_config

(PECL tidy:0.7-1.2)

tidy_reset_config — Tidy の設定をデフォルト値に戻す

説明

bool **tidy_reset_config** (void)

この関数は、Tidy の設定をデフォルト値に戻します。

注意: この関数は、Tidy 1.0 でのみ利用可能です。この関数は、Tidy 2.0 では古い関数として削除されているためです。

tidy_save_config

(PECL tidy:0.5.2-1.2)

tidy_save_config — 現在の設定を名前が付けられたファイルに保存する

説明

bool **tidy_save_config** (string $filename)

tidy_save_config() は、 現在の設定を指定したファイルに保存します。 デフォルトではない値だけが書き込まれます。

tidy_get_config(), tidy_getopt(), tidy_reset_config(), tidy_setopt() も参照ください。

注意: この関数は、Tidy 1.0 でのみ利用可能です。この関数は、Tidy 2.0 では古い関数として削除されているためです。

---

tidy_set_encoding

(PECL tidy:0.5.2-1.2)

tidy_set_encoding — マークアップをパースする際の入力／出力エンコーディングを設定する

説明

bool **tidy_set_encoding** (string $encoding)

入力／出力ドキュメントのエンコーディングを設定します。 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 encoding に使用可能な値は、 ascii, latin0, latin1, raw, utf8, iso2022, mac, win1252, ibm858, utf16, utf16le, utf16be, big5, shiftjis です。

注意: この関数は、Tidy 1.0 でのみ利用可能です。この関数は、Tidy 2.0 では古い関数として削除されているためです。

---

tidy_setopt

(PECL tidy:0.5.2-1.2)

tidy_setopt — 指定した Tidy ドキュメントについての設定を更新する

説明

bool **tidy_setopt** (string $option , mixed $value)

tidy_setopt() は、 指定したオプション option の値を 新しい値 value に更新します。 各設定オプションの一覧を » http://tidy.sourceforge.net/docs/quickref.html でご覧いただけます。

Example#1 tidy_setopt() の例

```
<?php
$html = '<p>test</i>';

$tidy = tidy_parse_string($html);

tidy_setopt('indent', FALSE);
?>
```

tidy_getopt(), tidy_get_config(), tidy_reset_config(), tidy_save_config() も参照ください。

注意: この関数は、Tidy 1.0 でのみ利用可能です。この関数は、Tidy 2.0 では古い関数として削除されているためです。

---

tidy_warning_count

(PHP 5, PECL tidy:0.5.2-1.2)

tidy_warning_count — 指定したドキュメントについて発生した Tidy 警告の数を返す

説明

int **tidy_warning_count** (tidy $object)

tidy_warning_count() は、 指定したドキュメントについて発生した Tidy 警告の数を返します。

Example#1 tidy_warning_count() の例

```
<?php
$html = '<p>test</i>
<bogustag>bogus</bogustag>';

$tidy = tidy_parse_string($html);

echo tidy_error_count($tidy) . "¥n"; //1
echo tidy_warning_count($tidy) . "¥n"; //5
?>
```

tidy_access_count(), tidy_error_count() も参照ください。

---

tidyNode->hasChildren

(PHP 5 >= 5.0.1)

tidyNode->hasChildren — このノードが子を持つ場合 true を返す

説明

bool **tidyNode->hasChildren** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->has_children()** という名前でした。

---

tidyNode->hasSiblings

(PHP 5 >= 5.0.1)

tidyNode->hasSiblings — このノードが兄弟ノードを持つ場合 true を返す

説明

bool **tidyNode->hasSiblings** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->has_siblings()** という名前でした。

---

tidyNode->isAsp

(PHP 5 >= 5.0.1)

tidyNode->isAsp — このノードが ASP コードの場合 true を返す

説明

bool **tidyNode->isAsp** (void)

現在のノードが ASP コードの場合、この関数は **TRUE** を返します。 それ以外の場合 **FALSE** を返します。

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->is_asp()** という名前でした。

---

tidyNode->isComment

(PHP 5 >= 5.0.1)

tidyNode->isComment — このノードがコメントである場合 true を返す

説明

bool **tidyNode->isComment** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->is_comment()** という名前でした。

---

tidyNode->isHtml

(PHP 5 >= 5.0.1)

tidyNode->isHtml — このノードが HTML ドキュメントの場合 true を返す

説明

bool **tidyNode->isHtml** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->is_html()** という名前でした。

---

tidyNode->isJste

(PHP 5 >= 5.0.1)

tidyNode->isJste — このノードが JSTE であるの場合 true を返す

説明

bool **tidyNode->isJste** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->is_jste()** という名前でした。

tidyNode->isPhp

(PHP 5 >= 5.0.1)

tidyNode->isPhp — このノードが PHP コードの場合 true を返す

説明

bool **tidyNode->isPhp** (void)

もし現在のノードが PHP コードであれば **TRUE** を返します。 そうでない場合は **FALSE** を返します。

Example#1 HTML と PHP が混ざったドキュメントから PHP コードを取得する

```
<?php

$html = <<< HTML
<html><head>
<?php echo '<title>title</title>'; ?>
</head>
<body>

<?php
echo 'hello world!';
?>

</body></html>
HTML;


$tidy = tidy_parse_string($html);
$num = 0;

get_php($tidy->html());


function get_php($node) {

    // 現在のノードが PHP コードかどうかをチェックする
    if($node->isPhp()) {
        echo "¥n¥n# PHP node #" . ++$GLOBALS['num'] . "¥n";
        echo $node->value;
    }

    // 現在のノードが子を持つかどうかをチェックする
    if($node->hasChildren()) {
        foreach($node->child as $child) {
            get_php($child);
        }
    }
}

?>
```

上の例の出力は以下となります。

```
# PHP node #1
<?php echo '<title>title</title>'; ?>

# PHP node #2
<?php
echo 'hello world!';
?>
```

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->is_php()** という名前でした。

tidyNode->isText

(PHP 5 >= 5.0.1)

tidyNode->isText — このノードがテキスト（マークアップでない）の場合 true を返す

説明

bool **tidyNode->isText** (void)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意: この関数は、PHP 4/Tidy 1 では **tidy_node->is_text()** という名前でした。

tidyNode::getParent

(PHP 5 >= 5.2.2)

tidyNode::getParent — カレントノードの親ノードを返す

説明

tidyNode **tidyNode::getParent** (void)

カレントノードの親ノードを返します。

返り値

そのノードが親を持っている場合に tidyNode、 それ以外の場合に **NULL** を返します。

目次

Tokenizer 関数

導入

tokenizer 関数は、Zend Engine に組み込まれた PHP tokenizer への インターフェイスを提供します。以下の関数により、 字句解析レベルの言語処理を行うことなく、PHP ソースを解析/修正する ツールを作成することが可能となります。

トークンに関する付録も参照ください。

要件

外部ライブラリを必要としません。

インストール手順

PHP 4.3.0 以降、以下の関数はデフォルトで有効となっています。 これ以前のバージョンの場合、 --enable-tokenizer を指定して PHP をコンパイルする必要があります。 --disable-tokenizer を指定すること により、tokenizer サポートを無効とすることができます。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

注意: tokenizer の組込みサポートは PHP 4.3.0 で利用可能となりました。

定義済み定数

この拡張モジュールを組み込んで PHP をコンパイルするか、あるいは実行時に 動的にモジュールを読み込むと、パーサトークンの一覧 に挙げられている トークンが定数として定義されます。

例

以下に tokenizer を用いた簡単な PHP スクリプトの例を示します。この例は、 PHP ファイルを読み込み、ソースから全てのコメントを削除し、コードのみを 出力するものです。

Example#1 tokenizer によりコメントを削除する

```
<?php
/*
 * T_ML_COMMENT は PHP 5 には存在しません。
 * 以下の 3 行で、古いバージョンとの互換性を確保するために
 * これらを定義しています。
 *
 * その次の 2 行では、PHP 5 にのみ存在する T_DOC_COMMENT を定義しています。
 * T_ML_COMMENT が存在するかどうかで PHP 4 かどうかを判断しています。
 */
if (!defined('T_ML_COMMENT')) {
    define('T_ML_COMMENT', T_COMMENT);
} else {
    define('T_DOC_COMMENT', T_ML_COMMENT);
}

$source = file_get_contents('example.php');
$tokens = token_get_all($source);

foreach ($tokens as $token) {
    if (is_string($token)) {
        // 簡単な1文字毎のトークン
        echo $token;
    } else {
        // トークン配列
        list($id, $text) = $token;

        switch ($id) {
            case T_COMMENT:
            case T_ML_COMMENT: // we've defined this
            case T_DOC_COMMENT: // and this
                // コメントの場合は何もしない
                break;

            default:
                // それ以外の場合 -> "そのまま"出力
                echo $text;
                break;
        }
    }
}
?>
```

---

token_get_all

(PHP 4 >= 4.2.0, PHP 5)

token_get_all — 指定したソースを PHP トークンに分割する

説明

array **token_get_all** (string $source)

token_get_all() は指定した文字列 source をパースし、Zend engine の 字句解析スキャナにより PHP 言語のトークンに分割します。

パーサトークンの一覧を得るには、パーサトークンの一覧 を参照するか、 あるいは token_name() でトークン値を文字列表現に 変換します。

パラメータ

source

パースする PHP ソース。

返り値

トークン ID の配列を返します。配列の各要素には、1 文字単位の文字列 (例: ;、 .、 >、! など...)、 またはトークンのインデックスを 0 番目

の要素、トークンの文字列表現を 1 番目の要素、 行番号を 2 番目の要素とする配列が含まれます。

例

Example#1 token_get_all() の例

```
<?php
$tokens = token_get_all('<?php echo; ?>'); /* => array(
                                                array(T_OPEN_TAG, '<?php'),
                                                array(T_ECHO, 'echo'),
                                               ';',
                                                array(T_CLOSE_TAG, '?>') ); */

/* 以下の例では、文字列が T_COMMENT (PHP <5 では T_ML_COMMENT)
   ではなく T_INLINE_HTML とパースされていることに注意しましょう。
   これは、指定した "code" の中に開始タグ/終了タグが含まれていないからです。
   通常のファイルで、コメントを <?php ?> タグの外部に書いた場合にも同じようになります。 */
$tokens = token_get_all('/* comment */'); // => array(array(T_INLINE_HTML, '/* comment */'));
?>
```

変更履歴

バージョン	説明
5.2.2	2 番目の要素として行番号を返すようになりました。

---

token_name

(PHP 4 >= 4.2.0, PHP 5)

token_name — 指定した PHP トークンのシンボル名を取得する

説明

string **token_name** (int $token)

token_name()は、PHP token 値のシンボル名を返します。

パラメータ

token

トークンの値。

返り値

token に対応するシンボル名を返します。 返される名前は、検索用に指定したトークン定数の名前に一致します。

例

Example#1 token_name() の例

```
<?php
// 260は T_REQUIRE トークンのトークン値
echo token_name(260);         // -> "T_REQUIRE"

// 自身の名前にマップするトークン定数
echo token_name(T_FUNCTION); // -> "T_FUNCTION"
?>
```

参考

- パーサトークンの一覧

目次

Unicode 関数

導入

Unicode をサポートします。

警告

この拡張モジュールは現在開発中のものであり、まだ一般には公開されていません。

要件

» ICU 3.4 あるいはそれ以降のバージョンが必要です。

インストール手順

まずはじめに ICU をダウンロードしてインストールします。

Example#1 Unix での ICU のインストール

```
./configure --disable-threads --enable-extras --enable-icuio --enable-layout
make && make install
```

それから、最新の PHP をチェックアウトして、 --with-icu-dir=<dir> オプション付きで configure します。<dir> には ICU をインストールしたディレクトリを指定します。 ICU を標準的な場所にインストールした場合は、 ディレクトリを明示的に指定する必要はありません。

実行時設定

php.ini の設定により動作が変化します。

Unicode 設定オプション

名前	デフォルト	変更の可否	変更履歴
unicode.fallback_encoding	NULL	PHP_INI_ALL	PHP 6.0.0 以降で使用可能です。
unicode.from_error_mode	"2"	PHP_INI_ALL	PHP 6.0.0 以降で使用可能です。
unicode.from_error_subst_char	"3f"	PHP_INI_ALL	PHP 6.0.0 以降で使用可能です。
unicode.http_input_encoding	NULL	PHP_INI_ALL	PHP 6.0.0 以降で使用可能です。
unicode.output_encoding	NULL	PHP_INI_ALL	PHP 6.0.0 以降で使用可能です。
unicode.runtime_encoding	NULL	PHP_INI_ALL	PHP 6.0.0 以降で使用可能です。
unicode.script_encoding	NULL	PHP_INI_ALL	PHP 6.0.0 以降で使用可能です。
unicode.semantics	"0"	PHP_INI_SYSTEM	PHP 6.0.0 以降で使用可能です。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

unicode.output_encoding string

出力用のデフォルトエンコーディング。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

定数	値	説明
U_INVALID_STOP	0	無効な文字が現れた時点で停止します。
U_INVALID_SKIP	1	無効な文字は読み飛ばします。
U_INVALID_SUBSTITUTE	2	無効な文字を置き換えます。
U_INVALID_ESCAPE	3	無効な文字をエスケープします。

---

unicode_decode

(No version information available, might be only in CVS)

unicode_decode — バイナリ文字列を Unicode 文字列に変換する

説明

unicode **unicode_decode** (string $input , string $encoding [, int $errmode])

encoding でエンコードされたバイナリ文字列を unicode 文字列に変換します。

パラメータ

input

変換する文字列。

encoding

input のエンコード方式。

errmode

変換エラーモード。このパラメータで、 コンバータが文字を変換できなかったときの動作を指定します。 使用できるモードについては unicode_set_error_mode() を参照ください。このパラメータを省略した場合は、グローバルなエラーモードを使用します。

返り値

unicode 文字列を返します。失敗した場合は FALSE を返します。

エラー / 例外

指定した encoding に対応するコンバータが作成できない場合は E_WARNING レベルのエラーが発生します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- unicode_set_error_mode()
- unicode_semantics()
- unicode_encode()

---

unicode_encode

(No version information available, might be only in CVS)

unicode_encode — unicode 文字列を任意のエンコーディングに変換する

説明

string **unicode_encode** (unicode $input , string $encoding [, int $errmode])

unicode 文字列を受け取り、それを指定したエンコーディング encoding の文字列に変換します。

パラメータ

input

エンコードする unicode 文字列。

encoding

input の新しいエンコーディング。

errmode

変換エラーモード。このパラメータで、 コンバータが文字を変換できなかったときの動作を指定します。 使用できるモードについては unicode_set_error_mode() を参照ください。このパラメータを省略した場合は、グローバルなエラーモードを使用します。

返り値

成功した場合に文字列、失敗した場合に FALSE を返します。

エラー / 例外

指定したエンコーディングの変換器が作成できない場合に E_WARNING レベルのエラーが発生します。

例

Example#1 unicode_encode() の例

注意: 出力は、エンティティではなく文字になります。

```
<?php
header ('Content-Type: text/plain; charset=ISO-8859-2');

$encoded = unicode_encode ('¥u0150¥u0179', 'ISO-8859-2');

echo 'Unicode semantics: ', ini_get ('unicode_semantics'), PHP_EOL;
echo 'The string itself:', $encoded, PHP_EOL;
echo 'The length of the string: ', strlen ($encoded);
?>
```

上の例の出力は、たとえば以下のようになります。

```
Unicode semantics: 1
The string itself: &#336;&#377;
The length of the string: 2
```

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- unicode_set_error_mode()
- unicode_semantics()
- unicode_decode()

---

unicode_get_error_mode

(No version information available, might be only in CVS)

unicode_get_error_mode — 文字列変換のエラーモードを取得する

説明

int **unicode_get_error_mode** (int $direction)

direction で指定した方向の、 現在の文字列変換のエラーモードを返します。

パラメータ

direction

変換の方向を設定します。これは、 **FROM_UNICODE** あるいは **TO_UNICODE** のいずれかとなります。詳細は unicode_set_error_mode() を参照ください。

返り値

この関数は、モード定数のいずれかを返します。失敗した場合は **FALSE** を返します。

エラー / 例外

direction に不正な値を指定した場合は **E_WARNING** レベルのエラーが発生します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- unicode_set_error_mode()
- unicode_encode()
- unicode_decode()

---

unicode_get_subst_char

(No version information available, might be only in CVS)

unicode_get_subst_char — 文字列変換エラー時に使用する置換文字を取得する

説明

unicode **unicode_get_subst_char** (void)

unicode_set_subst_char() で設定した、 文字列変換エラー時に使用する置換文字を返します。

返り値

unicode 型の置換文字を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- unicode_set_error_mode()
- unicode_set_subst_char()

---

unicode_semantics

(No version information available, might be only in CVS)

unicode_semantics — unicode 機能が使用可能かどうかを調べる

説明

bool ***unicode_semantics*** *(void)*

unicode_semantics() *は、 unicode 機能が使用可能かどうかを調べます。*

返り値

TRUE *あるいは* ***FALSE*** *を返します。*

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- *unicode_encode()*
- *unicode_decode()*

---

unicode_set_error_mode

(No version information available, might be only in CVS)

unicode_set_error_mode — 文字列変換のエラーモードを設定する

説明

bool ***unicode_set_error_mode*** *(int $direction , int $mode)*

この関数は、異なるエンコーディング間での文字列変換の際のエラーモードを設定します。 文字列変換の際にエラーが発生する原因としては、 不正な文字があらわれた場合や 変換先のエンコードで表現できない文字があらわれた場合などがあります。 デフォルトでは、エラーが発生したらその場で変換を停止します。

パラメータ

direction

direction は、 そのエラーモードを適用する変換方向を設定します。 ***FROM_UNICODE*** *か* ***TO_UNICODE*** *のいずれかです。* ***FROM_UNICODE*** *の場合は unicode 文字列からバイナリ文字列への変換、* ***TO_UNICODE*** *の場合はバイナリ文字列から unicode 文字列への変換を表します。*

mode

mode は、変換エラーの処理方法を指定します。 以下の定数のいずれかとなります。

使用できるモード

モード	説明
U_CONV_ERROR_STOP	変換を停止する。これがデフォルトのモードです。
U_CONV_ERROR_SKIP	その文字を読み飛ばす。
U_CONV_ERROR_SUBST	特定の文字で置き換える。置き換える文字は unicode_set_subst_char() で設定します。
U_CONV_ERROR_ESCAPE_UNICODE	間違ったバイト列をエスケープし、Unicode 形式の文字フォーマットで表す。
U_CONV_ERROR_ESCAPE_ICU	間違ったバイト列をエスケープし、ICU 形式の文字フォーマットで表す。
U_CONV_ERROR_ESCAPE_JAVA	間違ったバイト列をエスケープし、Java 形式の文字フォーマットで表す。
U_CONV_ERROR_ESCAPE_XML_DEC	間違ったバイト列をエスケープし、十進形式の文字フォーマットで表す。
U_CONV_ERROR_ESCAPE_XML_HEX	間違ったバイト列をエスケープし、十六進形式の文字フォーマットで表す。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

エラー / 例外

direction あるいは mode に不正な値を指定した場合は ***E_WARNING*** *レベルのエラーが発生します。*

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- *unicode_encode()*
- *unicode_decode()*

---

unicode_set_subst_char

(No version information available, might be only in CVS)

unicode_set_subst_char — 文字列変換エラー時に使用する置換文字を設定する

説明

bool **unicode_set_subst_char** (unicode $character)

文字列を unicode から変換したり unicode に変換したりする際には、 変換できない文字があらわれることがあります。変換エラーモードを **U_CONV_ERROR_SUBST** に設定している場合、 このような文字はこの関数で設定した文字で置き換えられます。 デフォルトの置換文字は ? です。

パラメータ

character

使用する置換文字。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

エラー / 例外

character が無効な場合に **E_WARNING** レベルのエラーが発生します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

参考

- unicode_set_error_mode()
- unicode_get_error_mode()
- unicode_encode()
- unicode_decode()

目次

URL 関数

導入

URL文字列を処理(エンコード、デコード、パース)します。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

リソース型は定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

以下の定数は parse_url() で使用するもので、 PHP 5.1.2 以降で使用可能です。

PHP_URL_SCHEME (integer)
PHP_URL_HOST (integer)
PHP_URL_PORT (integer)
PHP_URL_USER (integer)
PHP_URL_PASS (integer)
PHP_URL_PATH (integer)
PHP_URL_QUERY (integer)
PHP_URL_FRAGMENT (integer)

---

base64_decode

(PHP 4, PHP 5)

base64_decode — MIME base64 方式によりエンコードされたデータをデコードする

説明

string **base64_decode** (string $data [, bool $strict])

base64 でエンコードされた data をデコードします。

パラメータ

data

デコードされるデータ。

strict

入力に空白あるいはその他の区切り文字が含まれる場合に **FALSE** を返す。

返り値

もとのデータを返します。失敗した場合は **FALSE** を返します。 返り値はバイナリになることもあります。

変更履歴

バージョン	説明
5.2.0	strict が追加されました。

例

Example#1 base64_decode() の例

```
<?php
$str = 'VGhpcyBpcyBhbiBlbmNvZGVkIHN0cmluZw==';
echo base64_decode($str);
?>
```

上の例の出力は以下となります。

```
This is an encoded string
```

参考

- base64_encode()
- » RFC 2045 6.8 節

---

base64_encode

(PHP 4, PHP 5)

base64_encode — MIME base64 方式でデータをエンコードする

説明

string **base64_encode** (string $data)

指定した data を base64 でエンコードします。

このエンコードは、メールの本体のように 8 ビットクリーンではないトランスポート層を通じても、 バイナリデータが生き残れるように設計されています。

Base64 でエンコードされたデータは、エンコード前のデータにくらべて 33% 余計に容量が必要です。

パラメータ

data

エンコードするデータ。

返り値

エンコードされたデータを文字列で返します。

例

Example#1 base64_encode() の例

```
<?php
$str = 'This is an encoded string';
echo base64_encode($str);
?>
```

上の例の出力は以下となります。

```
VGhpcyBpcyBhbiBlbmNvZGVkIHN0cmluZw==
```

参考

- base64_decode()
- chunk_split()
- convert_uuencode()
- » RFC 2045 6.8 節

---

get_headers

(PHP 5)

get_headers — HTTP リクエストに対するレスポンス内で サーバによって送出された全てのヘッダを取得する

説明

array **get_headers** (string $url [, int $format])

get_headers() は、HTTP リクエストに対するレスポンス内で サーバによって送出されたヘッダの配列を返します。

パラメータ

url

対象となる URL。

format

オプションの format パラメータが 1 にセットされた場合、**get_headers()** はレスポンスをパースし、配列のキーをセットします。

返り値

数値添字配列あるいは連想配列でヘッダを返します。 失敗した場合は **FALSE** を返します。

変更履歴

バージョン	説明
5.1.3	この関数はデフォルトのストリームコンテキストを使用します。 これは、stream_context_get_default() 関数を使用して設定/変更することが可能です。

例

Example#1 get_headers() の例

```
<?php
$url = 'http://www.example.com';

print_r(get_headers($url));

print_r(get_headers($url, 1));
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [0] => HTTP/1.1 200 OK
     [1] => Date: Sat, 29 May 2004 12:28:13 GMT
     [2] => Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
     [3] => Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
     [4] => ETag: "3f80f-1b6-3e1cb03b"
     [5] => Accept-Ranges: bytes
     [6] => Content-Length: 438
     [7] => Connection: close
     [8] => Content-Type: text/html
)
```

```
Array
(
     [0] => HTTP/1.1 200 OK
     [Date] => Sat, 29 May 2004 12:28:14 GMT
     [Server] => Apache/1.3.27 (Unix)  (Red-Hat/Linux)
     [Last-Modified] => Wed, 08 Jan 2003 23:11:55 GMT
     [ETag] => "3f80f-1b6-3e1cb03b"
     [Accept-Ranges] => bytes
     [Content-Length] => 438
     [Connection] => close
     [Content-Type] => text/html
)
```

---

get_meta_tags

(PHP 4, PHP 5)

get_meta_tags — ファイル上のすべてのメタタグ情報を配列に展開する

説明

array **get_meta_tags** (string $filename [, bool $use_include_path])

filename 内の各行をパースし <meta> タグ内の情報を配列として返します。 </head> でパースを終了します。

パラメータ

filename

HTML ファイルへのパスを表す文字列。 ローカルファイルのほか URL も指定できます。

Example#1 get_meta_tags() が何をパースするのか

```
<meta name="author" content="name">
<meta name="keywords" content="php documentation">
<meta name="DESCRIPTION" content="a php manual">
<meta name="geo.position" content="49.33;-86.59">
</head> <!-- ここでパースを終了します -->
```

(改行コードに注意してください。PHP は入力をパースするためにネイティブ関数を使用するため、 Macintosh のファイルは Unix 上では動作しません)。

use_include_path

use_include_path を **TRUE** に指定すると、 include_path ディレクティブの内容にしたがってファイルを探します。 これはローカルファイルにのみ有効で、URL の場合は使用できません。

返り値

パースされたメタタグを含む配列を返します。

name 属性の値が配列のキーとなります。content 属性の値が配列の要素となります。 標準の配列関数を利用することでこれらの値に簡単にアクセスすることができます。 name 属性で特別な文字が使われている場合は '_' で代用されます。 それ以外は小文字に変換されます。もしも同じ名前のメタタグがある場合には 最後のもののみが返されます。

変更履歴

バージョン	説明
4.0.5	クォートされていない HTML 属性もサポートするようになりました。

例

Example#2 get_meta_tags() が何を返すのか

```
<?php
// 上のタグは www.example.com 上のものだと仮定します
$tags = get_meta_tags('http://www.example.com/');

// すべてのキーが小文字であり、.（ピリオド）が _ に
// 置換されていることに注目してください。
echo $tags['author'];       // name
echo $tags['keywords'];     // php documentation
echo $tags['description'];  // a php manual
echo $tags['geo_position']; // 49.33;-86.59
?>
```

参考

- htmlentities()
- urlencode()

---

http_build_query

(PHP 5, PECL pecl_http:0.1.0-0.9.0)

http_build_query — URL エンコードされたクエリ文字列を生成する

説明

string **http_build_query** (array $formdata [, string $numeric_prefix [, string $arg_separator]])

与えられた連想配列（もしくは添字配列）から URL エンコードされたクエリ文字列を生成します。

パラメータ

formdata

プロパティを含む配列もしくはオブジェクト。

配列の形式は、単純な一次元構造、 もしくは配列の配列（言い換えると、他の配列を含む配列）となります。

numeric_prefix

もし数値インデックスが基底となる配列に使用されたり numeric_prefix が指定された場合、 基底となる配列の要素に対する数値インデックスの前にこれが追加されます。

これは、後で PHP や他の CGI アプリケーションによってデータがデコードされる際、 正当な変数名になるよう考慮したものです。

arg_separator

arg_separator.output が区分のためのセパレータとして使用されます。ただし、 このパラメータが指定されていた場合は それが使用されます。

返り値

URL エンコードされた文字列を返します。

変更履歴

バージョン	説明
5.1.2	パラメータ arg_separator が追加されました。
5.1.3	角括弧がエスケープされるようになりました。

例

Example#1 http_build_query() の簡単な使用法

```
<?php
$data = array('foo'=>'bar',
              'baz'=>'boom',
              'cow'=>'milk',
              'php'=>'hypertext processor');

echo http_build_query($data); //  foo=bar&baz=boom&cow=milk&php=hypertext+processor
echo http_build_query($data, '', '&amp;'); //  foo=bar&amp;baz=boom&amp;cow=milk&amp;php=hypertext+processor

?>
```

Example#2 数値インデックス要素の場合での http_build_query()

```
<?php
$data = array('foo', 'bar', 'baz', 'boom', 'cow' => 'milk', 'php' =>'hypertext processor');

echo http_build_query($data) . "¥n";
echo http_build_query($data, 'myvar_');
?>
```

上の例の出力は以下となります。

```
0=foo&1=bar&2=baz&3=boom&cow=milk&php=hypertext+processor
myvar_0=foo&myvar_1=bar&myvar_2=baz&myvar_3=boom&cow=milk&php=hypertext+processor
```

Example#3 複雑な配列の場合での http_build_query()

```
<?php
$data = array('user'=>array('name'=>'Bob Smith',
                            'age'=>47,
                            'sex'=>'M',
                            'dob'=>'5/12/1956'),
              'pastimes'=>array('golf', 'opera', 'poker', 'rap'),
              'children'=>array('bobby'=>array('age'=>12,
                                               'sex'=>'M'),
                                'sally'=>array('age'=>8,
                                               'sex'=>'F')),
              'CEO');

echo http_build_query($data, 'flags_');
?>
```

この例は以下を出力します:（可読性のため適宜折り返しています）

```
user%5Bname%5D=Bob+Smith&user%5Bage%5D=47&user%5Bsex%5D=M&
user%5Bdob%5D=5%2F12%2F1956&pastimes%5B0%5D=golf&pastimes%5B1%5D=opera&
pastimes%5B2%5D=poker&pastimes%5B3%5D=rap&children%5Bbobby%5D%5Bage%5D=12&
children%5Bbobby%5D%5Bsex%5D=M&children%5Bsally%5D%5Bage%5D=8&
```

```
children%5Bsally%5D%5Bsex%5D=F&flags_0=CEO
```

注意: 基底の配列内の数値インデックス要素 "CEO" のみ、 接頭辞を受け取ります。pastimes 以下にある他の数値インデックスは、正当な変数名にするための文字列の接頭辞を要求しません。

Example#4 オブジェクトの場合の http_build_query() の使用

```
<?php
class myClass {
     var $foo;
     var $baz;

     function myClass() {
         $this->foo = 'bar';
         $this->baz = 'boom';
     }
}

$data = new myClass();

echo http_build_query($data); //  foo=bar&baz=boom

?>
```

参考

- parse_str()
- parse_url()
- urlencode()
- array_walk()

---

parse_url

(PHP 4, PHP 5)

parse_url — URL を解釈し、その構成要素を返す

説明

mixed **parse_url** (string $url [, int $component])

この関数は、URL の様々な構成要素のうち特定できるものに関して 連想配列にして返します。

この関数は、指定された URL が有効かどうかを調べるためのもの ではなく、単に URL を上で示した 要素に分解するだけのものです。不完全な URL であっても受け入れられますし、 そのような場合でも **parse_url()** は可能な限り 正しく解析しようとします。

パラメータ

url

パースする URL

component

PHP_URL_SCHEME、 **PHP_URL_HOST**、**PHP_URL_PORT**、 **PHP_URL_USER**、**PHP_URL_PASS**、 **PHP_URL_PATH**、**PHP_URL_QUERY** あるいは **PHP_URL_FRAGMENT** のうちのいずれかを指定し、 特定の URL コンポーネントのみを 文字列で取得するようにします。

返り値

完全におかしな形式の URL については、**parse_url()** は **FALSE** を返し、**E_WARNING** を発生します。それ以外の場合は 連想配列が返され、その中には以下の要素（のうち少なくともひとつ）が含まれます。

- scheme - 例: http
- host
- port
- user
- pass
- path
- query - クエスチョンマーク ? 以降
- fragment - ハッシュマーク # 以降

component が指定されている場合、結果は array ではなく文字列で返されます。

変更履歴

バージョン	説明
5.1.2	パラメータ component が追加されました。

例

Example#1 parse_url() の例

```
<?php
$url = 'http://username:password@hostname/path?arg=value#anchor';

print_r(parse_url($url));
?>
```

上の例の出力は以下となります。

```
Array
(
    [scheme] => http
    [host] => hostname
    [user] => username
    [pass] => password
    [path] => /path
    [query] => arg=value
    [fragment] => anchor
)
```

注意

注意: この関数は相対 URL では動作しません。

注意: **parse_url()** は URL をパースするための関数であり、 URI をパースするものではありません。しかし、PHP の後方互換性を満たすため、 例外として file:// スキームについては 3 重スラッシュ(file:///...) が認められています。他のスキームにおいては、これは無効な形式となります。

参考

- pathinfo()
- parse_str()
- dirname()
- basename()

---

rawurldecode

(PHP 4, PHP 5)

rawurldecode — URL エンコードされた文字列をデコードする

説明

string **rawurldecode** (string $str)

文字列の中にパーセント記号 (%) に続いて 2 つの 16 進数があるような表現形式を、文字定数に置き換えて返します。

パラメータ

str

デコードする URL。

返り値

デコードされた URL を文字列で返します。

例

Example#1 rawurldecode() の例

```
<?php

echo rawurldecode('foo%20bar%40baz'); // foo bar@baz

?>
```

注意

注意: **rawurldecode()** は、プラス記号 ('+') をスペースに変換しません。urldecode() は変換します。

参考

- rawurlencode()
- urldecode()
- urlencode()

---

rawurlencode

(PHP 4, PHP 5)

rawurlencode — RFC 1738 に基づき URL エンコードを行う

説明

string **rawurlencode** (string $str)

指定した文字列を » RFC 1738 にもとづいてエンコードします。

パラメータ

str

エンコードする URL。

返り値

-_.

を除くすべての非アルファベット文字をパーセント 記号 (%)に続いて 2 つの 16 進数がある表現形式に 置き換えた文字列を返します。これは、文字定数が特殊な URL デリミタと して解釈されたり、URL デリミタが(いくつかの電子メールシステムのような) 転送メディアにより文字変換されて失われてしまったりすることが ないように、RFC 1738 で定められたエンコーディング方法です。

例

Example#1 FTP URL へのパスワードの埋め込み

```
<?php
echo '<a href="ftp://user:', rawurlencode('foo @+%/'),
     '@ftp.example.com/x.txt">';
?>
```

上の例の出力は以下となります。

```
<a href="ftp://user:foo%20%40%2B%25%2F@ftp.example.com/x.txt">
```

あるいは、URL の PATH_INFO 中の情報を渡した場合は、

Example#2 rawurlencode() の例 2

```
<?php
echo '<a href="http://example.com/department_list_script/',
    rawurlencode('sales and marketing/Miami'), '">';
?>
```

上の例の出力は以下となります。

```
<a href="http://example.com/department_list_script/sales%20and%20marketing%2FMiami">
```

参考

- rawurldecode()
- urldecode()
- urlencode()
- » RFC 1738

---

urldecode

(PHP 4, PHP 5)

urldecode — URL エンコードされた文字列をデコードする

説明

string **urldecode** (string $str)

与えられた文字列中のあらゆるエンコード文字 %## をデコードします。

パラメータ

str

デコードする文字列。

返り値

デコードした文字列を返します。

例

Example#1 urldecode() の例

```
<?php
$a = explode('&', $QUERY_STRING);
$i = 0;
while ($i < count($a)) {
    $b = split('=', $a[$i]);
    echo 'Value for parameter ', htmlspecialchars(urldecode($b[0])),
         ' is ', htmlspecialchars(urldecode($b[1])), "<br />¥n";
    $i++;
}
?>
```

参考

- urlencode()
- rawurlencode()
- rawurldecode()

---

urlencode

(PHP 4, PHP 5)

urlencode — 文字列を URL エンコードする

説明

string **urlencode** (string $str)

この関数は、URL の問い合わせ部分に使用する文字列のエンコードや 次のページへ変数を渡す際に便利です。

パラメータ

str

エンコードする文字列。

返り値

-_. を除くすべての非英数文字が % 記号 (%)に続く二桁の数字で置き換えられ、 空白は + 記号(+)にエンコードされます。 同様の方法で、WWW のフォームからポストされたデータはエンコードされ、 application/x-www-form-urlencoded メディア型も同様です。歴史的な理由により、この関数は RFC1738 エンコード(rawurlencode() を参照してください) とは異なり、 空白を + 記号にエンコードします。

例

Example#1 urlencode() の例

```
<?php
echo '<a href="mycgi?foo=', urlencode($userinput), '">';
?>
```

Example#2 urlencode() および htmlentities() の例

```
<?php
$query_string = 'foo=' . urlencode($foo) . '&bar=' . urlencode($bar);
echo '<a href="mycgi?' . htmlentities($query_string) . '">';
?>
```

注意

> **注意**: HTML エンティティにマッチする変数については注意が必要です。 &、&copy、&pound のようなものがブラウザから送信された 場合、エンティティの実体がその変数名の代わりに使用されます。 これは明らかな問題点であり、W3C が何年も指摘し続けてきたことです。 リファレンスは、» http://www.w3.org/TR/html4/appendix/notes.html#h-B.2.2 にあります。
> PHP では、 .ini ディレクティブの arg_separator により引数のセパレータを W3C が推奨するセミコロンに変更することが可能です。残念なことに、多くの ユーザエージントはこのセミコロン区切り形式でデータを送信しません。 よりポータブルな方法としては、セパレータに & の代わりに & を使用するというものがあります。この場合、PHP の arg_separator を変更する必要はありません。セパレータを & のままにし、htmlentities() あるいは htmlspecialchars() で URL をエンコードしてください。

参考

- urldecode()
- htmlentities()
- rawurlencode()
- rawurldecode()

目次

- parse_url — URL を解釈し、その構成要素を返す
- rawurldecode — URL エンコードされた文字列をデコードする
- rawurlencode — RFC 1738 に基づき URL エンコードを行う
- urldecode — URL エンコードされた文字列をデコードする
- urlencode — 文字列を URL エンコードする

変数操作関数(Variable Handling)

導入

変数の動作に関する情報については、このマニュアルの 言語リファレンスセクションにある 変数のエントリを参照下さ い。

要件

外部ライブラリを必要としません。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

php.ini の設定により動作が変化します。

変数設定オプション

名前	デフォルト	変更の可否	変更履歴
unserialize_callback_func	NULL	PHP_INI_ALL	PHP 4.2.0 から利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

unserialize_callback_func string

アンシリアライザがインスタンスを作成する必要がある未定義のクラス を見付けた場合に(未定義のクラス名をパラメータとして) unserialize() コールバック関数がコールされます。 指定した関数が定義されていない場合やその関数が足りないクラスを読 み込み/実装しない場合に警告が発生します。 コールバック関数のようなものが本当に必要な場合は、このエントリを 設定してください。

unserialize()も参照してください。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

debug_zval_dump

(PHP 4 >= 4.2.0, PHP 5)

debug_zval_dump — 内部的な Zend の値を表す文字列をダンプする

説明

void **debug_zval_dump** (mixed $variable)

内部的な Zend の値を表す文字列をダンプします。

パラメータ

variable

評価される変数

返り値

値を返しません。

例

Example#1 debug_zval_dump() の例

```
<?php
$var1 = 'Hello World';
$var2 = '';

$var2 =& $var1;
```

```
debug_zval_dump(&$var1);
?>
```

上の例の出力は以下となります。

```
&string(11) "Hello World" refcount(3)
```

注意: **refcount に注意** この関数によって返される refcount の値は、 特定の状況下では自明ではありません。例えば、とある開発者は上記の例で refcount は 2 を示すと予想したかも知れません。 実際に **debug_zval_dump()** がコールされるとき、 3 番目の参照が生成されます。
この動作は、変数が **debug_zval_dump()** に参照渡しされない場合、よりその度合いを増します。 例えば、上記の例を僅かに修正したバージョンを考えてみます。

```
<?php
$var1 = 'Hello World';
$var2 = '';

$var2 =& $var1;

debug_zval_dump($var1); // この時、参照で渡されない
?>
```

上の例の出力は以下となります。

```
string(11) "Hello World" refcount(1)
```

なぜ refcount(1) でしょうか? なぜなら関数がコールされるとき、 $var1 のコピーが作成されるためです。
この関数は、refcount が 1 の変数が渡される（コピーもしくは値渡し）とき、より 混乱させます。

```
<?php
$var1 = 'Hello World';

debug_zval_dump($var1);
?>
```

上の例の出力は以下となります。

```
string(11) "Hello World" refcount(2)
```

ここで refcount の 2 は自明ではありません。特に上記の例を考える場合はそうです。 では、何が起こっているのでしょうか ?
変数が単一の参照（**debug_zval_dump()** の引数として使用される前の $var1）を持つ場合、PHP のエンジンは関数に渡される様式を最適化します。内部的に PHP は参照（refcount がこの関数のスコープのために増加されます）のように $var1 を扱い、参照渡しされた変数に 書き込みが発生するかどうかの警告を伴ってコピーが作成されます。 ただし、書き込みの瞬間だけです。これは "copy on write" として知られます。
そのため、もし **debug_zval_dump()** がただ一つのパラメータ（もしくはない）に書き込みが発生した場合、コピーが作成されます。
その時までパラメータは参照を保持し、関数呼び出しのスコープに対して refcount が 2 にインクリメントされる原因になります。

参考

- var_dump()
- debug_backtrace()
- リファレンスの説明
- » リファレンスの説明 (Derick Rethans による)

---

doubleval

(PHP 4, PHP 5)

doubleval — floatval() のエイリアス

説明

この関数は次の関数のエイリアスです。 floatval().

変更履歴

バージョン	説明
4.2.0	doubleval() は floatval() のエイリアスになりました。 これ以前では doubleval() のみ存在していました。

---

empty

(PHP 4, PHP 5)

empty — 変数が空であるかどうかを検査する

説明

bool **empty** (mixed $var)

変数が空であるかどうかを検査する

パラメータ

var

チェックする変数

> **注意**: **empty()** は、変数のみをチェックし、それ以外の値を チェックすると文法エラーを生成します。つまり、次の例は動作しません: **empty(trim($name))**.

empty()は、変数が設定されていないときに警告が生成 されないことを除けば、(boolean) var の逆です。

返り値

var が空でないか、0でない値であれば **FALSE** を返します。

次のような値は空であると考えられます。:

- "" (空文字列)
- 0 (0 は整数)
- "0" (0は文字列)
- **NULL**
- **FALSE**
- array() (空の配列)
- var $var;(変数が宣言されているが、クラスの中で値が設定されていない)

変更履歴

バージョン	説明
PHP 5	PHP5では、プロパティがないオブジェクトは空でないと見なされます。
PHP 4	PHP 4では、文字列 "0"は空であると見なされます。

例

Example#1 簡単な empty() / isset() の比較

```
<?php
$var = 0;

// $var が空なのでtrueと評価されます
if (empty($var)) {
    echo '$var is either 0, empty, or not set at all';
}

// $var が設定されているのでtrueと評価されます
if (isset($var)) {
    echo '$var is set even though it is empty';
}
?>
```

注意

> **注意**: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

参考

- isset()
- unset()
- array_key_exists()
- count()
- strlen()
- PHP 型の比較表

---

floatval

(PHP 4 >= 4.2.0, PHP 5)

floatval — 変数の float 値を取得する

説明

float **floatval** (mixed $var)

var の float 値を返します。

パラメータ

var

あらゆるスカラ型を指定できます。配列あるいはオブジェクトに **floatval()** を使用することはできません。

返り値

指定した変数の *float* 値を返します。

例

Example#1 floatval() の例

```
<?php
$var = '122.34343The';
$float_value_of_var = floatval($var);
echo $float_value_of_var; //  122.34343
?>
```

参考

- intval()
- strval()
- settype()
- 型の相互変換

---

get_defined_vars

(PHP 4 >= 4.0.4, PHP 5)

get_defined_vars — 全ての定義済の変数を配列で返す

説明

array **get_defined_vars** *(void)*

この関数は、環境変数、サーバ変数、**get_defined_vars()** がコールされたスコープ内でユーザが定義した変数を含む、全ての の定義済の変数のリストを有する多次元の配列を返します。

返り値

すべての変数を含む多次元の配列を返します。

例

Example#1 get_defined_vars() の例

```
<?php
$b = array(1, 1, 2, 3, 5, 8);

$arr = get_defined_vars();

// $b を出力
print_r($arr["b"]);

// PHPインタプリタのパスを出力 (CGIとして使用された場合)
//  例えば、/usr/local/bin/php
echo $arr["_"];

// コマンドラインパラメータがある場合に出力
print_r($arr["argv"]);

// サーバ変数を全て表示
print_r($arr["_SERVER"]);

// 変数の配列で利用可能なキーを全て出力
print_r(array_keys(get_defined_vars()));
?>
```

変更履歴

バージョン	説明
5.0.0	返される配列に *$GLOBALS* 変数が 含まれます。

参考

- isset()
- get_defined_functions()
- get_defined_constants()

---

get_resource_type

(PHP 4 >= 4.0.2, PHP 5)

get_resource_type — リソース型を返す

説明

string **get_resource_type** (resource $handle)

この関数は、指定したリソースの型を取得します。

パラメータ

handle

評価されるリソースハンドル。

返り値

指定された handle がリソースであった場合、 この関数はその型を表す文字列を返します。この関数で型が判別できなかった 場合は、返り値は文字列 Unknown となります。

もし handle がリソースでない場合、 この関数は **FALSE** を返し、エラーを発生させます。

例

Example#1 get_resource_type() の例

```
<?php
// mysql link を出力
$c = mysql_connect();
echo get_resource_type($c) . "¥n";

// file を出力
$fp = fopen("foo", "w");
echo get_resource_type($fp) . "¥n";

// domxml document を出力
$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc) . "¥n";
?>
```

---

gettype

(PHP 4, PHP 5)

gettype — 変数の型を取得する

説明

string **gettype** (mixed $var)

PHP 変数 var の型を返します。

警告

返された文字列は将来のバージョンで変更される可能性があるので、 **gettype()** を使用して型を調べることはしないでください。 更に、この関数は文字列比較を行うため、処理が遅くなります。

代わりに is_* 関数を使用してください。

返り値

返された文字列は、以下のいずれかの値を持ちます。

- "boolean" (PHP 4 以降)
- "integer"
- "double" (歴史的な理由により、float の場合には "double"が返されます。"float" とはなりません)
- "string"
- "array"
- "object"
- "resource" (PHP 4 以降)
- "null" (PHP 4 以降)
- "user function" (PHP 3 のみ。古い形式)
- "unknown type"

PHP 4 の場合、ある関数に関する **gettype()** の以前の使用法を置き換えるには、function_exists() および method_exists() を使用する必要があります。

参考

- settype()

- is_array()
- is_bool()
- is_float()
- is_int()
- is_null()
- is_numeric()
- is_object()
- is_resource()
- is_scalar()
- is_string()

---

import_request_variables

(PHP 4 >= 4.0.7, PHP 5)

import_request_variables — GET/POST/Cookie 変数をグローバルスコープにインポートする

説明

bool **import_request_variables** (string $types [, string $prefix])

GET/POST/Cookie 変数をグローバルスコープにインポートします。 この関数は、register_globals を無効としているが、いくつかの変数をグローバルスコープで参照したいといった場合に有用です。

SERVER のような他の変数をグローバルスコープへインポートすることを考えている場合には、 extract() の使用を検討してください。

パラメータ

types

types パラメータを使用すると、インポートする リクエスト変数の種類を指定可能です。文字 'G'、'P'、'C' がそれぞれ GET、POST、Cookie を表します。これらは大文字小文字を区別しないため、 'g'、'p'、'c' の組み合せも使用することが可能です。 POST には、アップロードされたファイルに関する情報も含まれます。

> **注意**: 文字の順番には注意してください。"gp" とすると、POST 変数は同名の GET 変数を上書きします。GPC 以外の文字は無視されます。

prefix

変数名の接頭辞として使用され、 グローバルスコープにインポートされる全ての変数名の前に付加されます。 このため、"userid" という名前の GET 値があり、 接頭辞 "pref_" を指定した場合、 $pref_userid という名前のグローバル変数が作成されます。

> **注意**: prefix パラメータはオプションですが、 接頭辞を指定しないか接頭辞として空の文字列を指定した場合、 E_NOTICE レベルのエラーが発生します。 これは、セキュリティ上の問題を発生する可能性があります。 NOTICE レベルのエラーは、デフォルトの error reporting レベルでは表示されません。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 import_request_variables() の例

```
<?php
// GET および POST 変数を、接頭辞 "rvar_" を付けて
// インポートします
import_request_variables("gp", "rvar_");

echo $rvar_foo;
?>
```

参考

- $_REQUEST
- register_globals
- 定義済みの変数
- extract()

---

intval

(PHP 4, PHP 5)

intval — 変数の整数としての値を取得する

説明

int **intval** (mixed $var [, int $base])

指定された値 base を基数（デフォルトは 10）とする、 var の integer としての値を返します。

パラメータ

var

整数に変換するスカラー値

base

変換のための基数（デフォルトは 10）

返り値

成功時は var の整数値、失敗時は 0。 空の配列やオブジェクトの場合は 0、空でない配列やオブジェクトの場合は 1 を返します。

最大値はシステムに依存します。32 ビットシステムでは、 最大の符号付き整数の範囲 -2147483648 ～ 2147483647 となります。 このため、そのようなシステムでは intval('1000000000000') は 2147483647 を返します。 64 ビットシステムにおける最大の符号付き整数は 9223372036854775807 となります。

文字列の場合、文字列の最左の文字に依存しますが、ほとんどの場合で 0 を返します。 整数への変換 の一般的なルールが適用されます。

例

Example#1 intval() の例

以下の例は 32 ビットシステムに基づきます。

```php
<?php
echo intval(42);                      // 42
echo intval(4.2);                     // 4
echo intval('42');                    // 42
echo intval('+42');                   // 42
echo intval('-42');                   // -42
echo intval(042);                     // 34
echo intval('042');                   // 42
echo intval(1e10);                    // 1410065408
echo intval('1e10');                  // 1
echo intval(0x1A);                    // 26
echo intval(42000000);                // 42000000
echo intval(420000000000000000000);   // 0
echo intval('420000000000000000000'); // 2147483647
echo intval(42, 8);                   // 42
echo intval('42', 8);                 // 34
?>
```

注意

> **注意**: パラメータ base はパラメータ var が文字列でない限り意味がありません。

参考

- floatval()
- strval()
- settype()
- is_numeric()
- 型の相互変換
- BCMath任意精度数学関数

---

is_array

(PHP 4, PHP 5)

is_array — 変数が配列かどうかを検査する

説明

bool **is_array** (mixed $var)

与えられた変数が配列かどうかを検査します。

パラメータ

var

評価する変数

返り値

var が 配列型 の場合 **TRUE**、 そうでない場合 **FALSE** を返します。

例

Example#1 変数が配列かどうか検査する

```php
<?php
$yes = array('this', 'is', 'an array');

echo is_array($yes) ? 'Array' : 'not an Array';
```

```
echo "¥n";

$no = 'this is a string';

echo is_array($no) ? 'Array' : 'not an Array';
?>
```

上の例の出力は以下となります。

```
Array
not an Array
```

参考

- is_float()
- is_int()
- is_string()
- is_object()

---

is_binary

(No version information available, might be only in CVS)

is_binary — 変数がネイティブバイナリ文字列かどうかを調べる

説明

bool **is_binary** (mixed $var)

指定した変数が、ネイティブのバイナリ文字列かどうかを調べます。

パラメータ

var

調べる変数。

返り値

var がネイティブバイナリ文字列である場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- is_buffer()
- is_string()
- is_unicode()

---

is_bool

(PHP 4, PHP 5)

is_bool — 変数が boolean であるかを調べる

説明

bool **is_bool** (mixed $var)

指定した変数が boolean であるかどうかを調べます。

パラメータ

var

評価する変数。

返り値

var が boolean である場合に **TRUE** 、それ以外の場合に **FALSE** を返します。

例

Example#1 is_bool() の例

```
<?php
$a = false;
$b = 0;

// $a は boolean なので、これは true です
if (is_bool($a)) {
```

```
    echo "Yes, this is a boolean";
}

// $b は boolean ではないので、これは true ではありません
if (is_bool($b)) {
    echo "Yes, this is a boolean";
}
?>
```

参考

- is_float()
- is_int()
- is_string()
- is_object()
- is_array()

---

is_buffer

(No version information available, might be only in CVS)

is_buffer — 変数がネイティブ unicode あるいはバイナリ文字列かどうかを調べる

説明

bool **is_buffer** (mixed $var)

指定した変数が、ネイティブの unicode あるいはバイナリ文字列かどうかを調べます。

パラメータ

var

調べる変数。

返り値

var がネイティブの unicode あるいはバイナリ文字列である場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- is_binary()
- is_string()
- is_unicode()

---

is_callable

(PHP 4 >= 4.0.6, PHP 5)

is_callable — 引数が、関数としてコール可能な構造であるかどうかを調べる

説明

bool **is_callable** (mixed $var [, bool $syntax_only [, string &$callable_name]])

引数の内容が、関数としてコール可能かどうかを調べます。 変数が有効な関数名かどうかを調べたり、配列の中に適切に エンコードされたオブジェクトと関数名が格納されているかどうかを 調べたりすることが可能です。

パラメータ

var

関数名を格納した文字列か、あるいは 以下のようにオブジェクトとメソッド名を格納したオブジェクトとすることが可能です。

array($SomeObject, 'MethodName')

syntax_only

TRUE の場合、この関数は単に var が関数またはメソッドであるかどうかだけを調べます。 文字列以外の型の変数や不正な形式の配列は、 引数として受け付けられません。有効な配列の形式は、 最初のエントリがオブジェクトあるいは文字列で、2 番目のエントリが文字列である 2 つのエントリからなるものです。

callable_name

"呼び出し名" を受け取ります。下の例では "someClass:someMethod" です。これは someClass::SomeMethod() が static メソッドであるかのようにみえますが、 そうではないことに注意しましょう。

返り値

var がコール可能な場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

例

Example#1 is_callable() の例

```
<?php
//  変数が、関数としてコール可能かどうかを確かめます。

//
//  関数名を含む単純な配列
//

function  someFunction()
{
}

$functionVariable  = 'someFunction';

var_dump(is_callable($functionVariable, false, $callable_name));  //  bool(true)

echo  $callable_name, "¥n";  //  someFunction

//
//  メソッドを含む配列
//

class  someClass  {

   function  someMethod()
   {
   }

}

$anObject  = new  someClass();

$methodVariable  =  array($anObject, 'someMethod');

var_dump(is_callable($methodVariable, true, $callable_name));  //   bool(true)

echo  $callable_name, "¥n";  //    someClass:someMethod

?>
```

参考

- function_exists()
- method_exists()

---

is_double

(PHP 4, PHP 5)

is_double — is_float() のエイリアス

説明

この関数は次の関数のエイリアスです。 is_float().

---

is_float

(PHP 4, PHP 5)

is_float — 変数の型が float かどうか調べる

説明

bool **is_float** (mixed $var)

与えられた変数の型が float かどうかを調べます。

> **注意**: 変数が数値もしくは数値文字列の場合（フォームからの入力の場合は 常に文字列となります）、is_numeric() を使用する必要があります。

パラメータ

var

評価する変数

返り値

もし var が float 型 の場合 **TRUE**、 そうでない場合は **FALSE** を返します。

例

Example#1 is_float() の例

```
<?php
if(is_float(27.25)) {
```

```
 echo "float です\n";
}else {
 echo "float ではありません\n";
}
var_dump(is_float('abc'));
var_dump(is_float(23));
var_dump(is_float(23.5));
var_dump(is_float(1e7));  // 科学記法
var_dump(is_float(true));
?>
```

上の例の出力は以下となります。

```
float です
bool(false)
bool(false)
bool(true)
bool(true)
bool(false)
```

参考

- is_bool()
- is_int()
- is_numeric()
- is_string()
- is_array()
- is_object()

---

is_int

(PHP 4, PHP 5)

is_int — 変数が整数型かどうかを検査する

説明

bool **is_int** (mixed $var)

与えられた変数の型が整数型かどうかを検査します。

> **注意**: 変数が数値もしくは数値文字列の場合（フォームからの入力の場合は 常に文字列となります）、is_numeric() を使用する必要があります。

パラメータ

var

評価する変数

返り値

もし var が 整数型 の場合 **TRUE**、 そうでない場合は **FALSE** を返します。

例

Example#1 is_int() の例

```
<?php
if (is_int(23)) {
 echo "integer です\n";
} else {
 echo "integer ではありません\n";
}
var_dump(is_int(23));
var_dump(is_int("23"));
var_dump(is_int(23.5));
var_dump(is_int(true));
?>
```

上の例の出力は以下となります。

```
integer です
bool(true)
bool(false)
bool(false)
bool(false)
```

参考

- is_bool()
- is_float()

- is_numeric()
- is_string()
- is_array()
- is_object()

---

is_integer

(PHP 4, PHP 5)

is_integer — is_int() のエイリアス

説明

この関数は次の関数のエイリアスです。 is_int().

---

is_long

(PHP 4, PHP 5)

is_long — is_int() のエイリアス

説明

この関数は次の関数のエイリアスです。 is_int().

---

is_null

(PHP 4 >= 4.0.4, PHP 5)

is_null — 変数が **NULL** かどうか調べる

説明

bool **is_null** (mixed $var)

指定した変数が **NULL** かどうかを調べます。

パラメータ

var

評価する変数。

返り値

var が null の場合に **TRUE**、 それ以外の場合に **FALSE** を返します。

参考

- **NULL** 型
- isset()
- is_bool()
- is_numeric()
- is_float()
- is_int()
- is_string()
- is_object()
- is_array()

---

is_numeric

(PHP 4, PHP 5)

is_numeric — 変数が数字または数値文字列であるかを調べる

説明

bool **is_numeric** (mixed $var)

指定した変数が数値であるかどうかを調べます。数値文字列は以下の要素から なります。（オプションの）符号、任意の数の数字、（オプションの）小数部、 そして（オプションの）指数部。つまり、+0123.45e6 は数値として有効な値です。16 進表記（0xFF）も 認められますが、この場合は符号や小数部、指数部を含めることはできません。

パラメータ

var

評価する変数。

返り値

var が数値または数値文字列である場合に **TRUE**、それ以外の場合に **FALSE** を返します。

参考

- ctype_digit()
- is_bool()
- is_null()
- is_float()
- is_int()
- is_string()
- is_object()
- is_array()

---

is_object

(PHP 4, PHP 5)

is_object — 変数がオブジェクトかどうかを検査する

説明

bool **is_object** (mixed $var)

与えられた変数がオブジェクトかどうかを調べます。

パラメータ

var

評価する変数

返り値

もし *var* が *object* 型 の場合 **TRUE**、 そうでない場合は **FALSE** を返します。

注意

> **注意**: クラス定義が存在しない場合に、 シリアライズされていないオブジェクトに対して使用すると (gettype() は object を返すにもかかわらず) この関数は **FALSE** を返します。

参考

- is_bool()
- is_int()
- is_float()
- is_string()
- is_array()

---

is_real

(PHP 4, PHP 5)

is_real — is_float() のエイリアス

説明

この関数は次の関数のエイリアスです。 is_float().

---

is_resource

(PHP 4, PHP 5)

is_resource — 変数がリソースかどうかを調べる

説明

bool **is_resource** (mixed $var)

指定した変数がリソースかどうかを調べます。

パラメータ

var

評価する変数。

返り値

var が resource の場合に TRUE、 それ以外の場合に FALSE を返します。

例

Example#1 is_resource() の例

```
<?php

$db_link = @mysql_connect('localhost', 'mysql_user', 'mysql_pass');
if (!is_resource($db_link)) {
    die('Can¥'t connect : ' . mysql_error());
}

?>
```

参考

- リソース型のドキュメント
- get_resource_type()

---

is_scalar

(PHP 4 >= 4.0.5, PHP 5)

is_scalar — 変数がスカラかどうかを調べる

説明

bool **is_scalar** (mixed $var)

指定した変数がスカラかどうかを調べます。

スカラ変数には integer、float、string あるいは boolean が含まれます。 array、object および resource はスカラではありません。

> **注意**: リソース型は現在整数に基づく抽象型であるため、 **is_scalar()** は resource 型の値を スカラ値と判定しません。この実装の詳細は変更される可能性があるため、 前提にするべきではありません。

パラメータ

var

評価する変数。

返り値

var がスカラの場合に TRUE、 それ以外の場合に FALSE を返します。

例

Example#1 is_scalar() の例

```
<?php
function show_var($var)
{
    if (is_scalar($var)) {
        echo $var;
    } else {
        var_dump($var);
    }
}
$pi = 3.1416;
$proteins = array("hemoglobin", "cytochrome c oxidase", "ferredoxin");

show_var($pi);
show_var($proteins)

?>
```

上の例の出力は以下となります。

```
3.1416
array(3) {
  [0]=>
  string(10) "hemoglobin"
  [1]=>
  string(20) "cytochrome c oxidase"
  [2]=>
  string(10) "ferredoxin"
}
```

参考

- is_float()
- is_int()
- is_numeric()
- is_real()
- is_string()
- is_bool()
- is_object()
- is_array()

---

is_string

(PHP 4, PHP 5)

is_string — 変数の型が文字列かどうかを調べる

説明

bool **is_string** (mixed $var)

指定した変数の型が文字列かどうかを調べます。

パラメータ

var

評価する変数。

返り値

var の型が string である場合に **TRUE**、それ以外の場合に **FALSE** を返します。

例

Example#1 is_string() の例

```
<?php
if (is_string("23")) {
 echo "文字列です\n";
} else {
 echo "文字列ではありません\n";
}
var_dump(is_string('abc'));
var_dump(is_string("23"));
var_dump(is_string(23.5));
var_dump(is_string(true));
?>
```

上の例の出力は以下となります。

```
文字列です
bool(true)
bool(true)
bool(false)
bool(false)
```

参考

- is_float()
- is_int()
- is_bool()
- is_object()
- is_array()

---

is_unicode

(No version information available, might be only in CVS)

is_unicode — 変数が unicode 文字列かどうかを調べる

説明

bool **is_unicode** (mixed $var)

指定した変数が、unicode 文字列かどうかを調べます。

パラメータ

var

調べる変数。

返り値

var が unicode 文字列である場合に TRUE、それ以外の場合に FALSE を返します。

参考

- is_binary()
- is_buffer()
- is_string()
- unicode_encode()

---

isset

(PHP 4, PHP 5)

isset — 変数がセットされているかどうかを検査する

説明

bool **isset** (mixed $var [, mixed $var [, $...]])

変数がセットされているかどうかを調べます。

変数が、unset() により割当を解除された場合、 何も値が設定されていない状態になります。 NULLに設定されている変数を調べた場合、 isset() はFALSEを返します。 NULLバイト("¥0")はPHPの定数 NULLと等価ではないことにも注意してください。

複数のパラメータを渡した場合は、isset() はそれらすべてがセットされている場合にのみ TRUE を返します。 左から順に評価を行い、セットされていない変数があった時点で処理を終了します。

パラメータ

var

調べたい変数。

var

別の変数。

...

返り値

var が存在すればTRUE、 そうでなければFALSEを返します。

例

Example#1 isset() の例

```php
<?php

$var = '';

// これは TRUE と評価されるので、テキストが出力される
if (isset($var)) {
    echo "This var is set so I will print.";
}

// 次の例では、isset() の戻り値を出力するため var_dump を
// 使用している

$a = "test";
$b = "anothertest";

var_dump(isset($a));      // TRUE
var_dump(isset($a, $b)); // TRUE

unset ($a);

var_dump(isset($a));     // FALSE
var_dump(isset($a, $b)); // FALSE

$foo = NULL;
var_dump(isset($foo));   // FALSE

?>
```

この関数は配列の要素に対しても動作します:

```php
<?php
```

```
$a = array ('test' => 1, 'hello' => NULL);

var_dump(isset($a['test']));            // TRUE
var_dump(isset($a['foo']));             // FALSE
var_dump(isset($a['hello']));           // FALSE

// キー 'hello' は NULL と等しいのでセットされていないと見なされる。
// もし NULL なキー値に対してチェックを行いたい場合、次を試してみること:
var_dump(array_key_exists('hello', $a)); // TRUE

?>
```

注意

警告

isset() は何らかの値が渡された 変数の場合のみ動作します。そうでない場合、パースエラーとなります。 もし 定数 が設定されているかどうかをチェックする場合は、 defined() を使用してください。

> **注意**: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

参考

- empty()
- unset()
- defined()
- PHP 型の比較表
- array_key_exists()
- is_null()
- エラー制御演算子 @

---

print_r

(PHP 4, PHP 5)

print_r — 指定した変数に関する情報を解りやすく出力する

説明

mixed **print_r** (mixed $expression [, bool $return])

print_r() は、 変数の値に関する情報を解り易い形式で表示します。

print_r()、var_dump() および var_export() は、PHP 5 においてオブジェクトの protected および private のプロパティも表示します。

print_r() は、配列ポインタを終端まで移動する ことに注意してください。ポインタを最初に戻すために reset() を使用してください。

パラメータ

expression

表示したい式。

return

print_r() の結果を取得したい場合には return 引数を試用してください。 **print_r()** はデフォルトでは結果を直接表示してしまいますが この引数が **TRUE** の場合には結果を戻します。

返り値

string、integer、float を指定した場合はその値が出力されます。 array を指定した場合、キーと要素を表す形式で値が 表示されます。object に関しても同様の表示形式となります。

注意

> **注意**: この関数は、このパラメータに対して内部的に出力バッファリングを使用しています。そのため、ob_start() コールバック関数の中で使用することはできません。

変更履歴

バージョン	説明
4.3.0	return 引数が PHP 4.3.0 以降追加されました。 PHP4.3.0 より前のバージョンを使っている場合で print_r() の結果を取得したい場合には、 出力制御関数を使用してください。
4.0.4	PHP 4.0.4 より前のバージョンでは、自分自身への直接または間接の 参照を含む array または object が指定された 場合には、無限に print_r() が実行されてしまいました。 *print_r($GLOBALS)* が例で、 *$GLOBALS* はそれ自体自分自身への参照を有する グローバル変数です。

例

Example#1 print_r() の例

```
<pre>
<?php
$a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x', 'y', 'z'));
print_r ($a);
?>
</pre>
```

上の例の出力は以下となります。

```
<pre>
Array
(
    [a] => apple
    [b] => banana
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
</pre>
```

Example#2 return 引数の例

```
<?php
$b = array ('m' => 'monkey', 'foo' => 'bar', 'x' => array ('x', 'y', 'z'));
$results = print_r($b, true); // print_r の結果が $results に格納されます
?>
```

参考

- ob_start()
- var_dump()
- var_export()

---

serialize

(PHP 4, PHP 5, PECL axis2:0.1.0-0.1.1)

serialize — 値の保存可能な表現を生成する

説明

string **serialize** (mixed $value)

値の保存可能な表現を生成します。

型や構造を失わずに PHP の値を保存または渡す際に有用です。

シリアル化された文字列を PHP の値に戻すには、 unserialize() を使用してください。

パラメータ

value

シリアル化する値。 **serialize()** は、resource 以外の全ての型を処理します。自分自身への参照を含む配列を **serialize()** することも可能です。 **serialize()** している配列/オブジェクト内の 循環参照も保存されます。その他の参照は失われます。

PHP は、シリアル化の前にまずメンバ関数 __sleep のコールを試みます。ここで、シリアル化の前のオブジェクトの後始末処理 などを行います。同様に、unserialize() で オブジェクトを復元した際にはメンバ関数 __wakeup がコールされます。

返り値

value の保存可能なバイトストリーム表現を含む文字列を返します。

例

Example#1 serialize() の例

```
<?php
// $session_data はカレントユーザーのセッション情報を含む多次元配列を
// 保持しています。リクエストの最後にこれをデータベースに保存するために
// serialize() を使用します。

$conn = odbc_connect("webdb", "php", "chicken");
$stmt = odbc_prepare($conn,
        "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array (serialize($session_data), $_SERVER['PHP_AUTH_USER']);
if (!odbc_execute($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
      "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
        /* 何か問題があったようです.. */
    }
}
?>
```

変更履歴

バージョン	説明
4.0.7	オブジェクトのシリアル化処理の問題が修正されました。
4.0.0	オブジェクトをシリアル化しても、メソッドが失われなくなりました。 詳細は オブジェクトのシリアル化 を参照ください。

注意

注意: PHP の組み込みオブジェクトをシリアル化することはできません。

参考

- unserialize()
- オブジェクトのシリアル化

---

settype

(PHP 4, PHP 5)

settype — 変数の型をセットする

説明

bool **settype** (mixed &$var , string $type)

変数 var の型を type にセットします。

パラメータ

var

変換する変数。

type

type の値は以下の命令のいずれかです。

- "boolean" (または、PHP 4.2.0以降は"bool")
- "integer" (または、PHP 4.2.0以降は"int")
- "float" (PHP 4.2.0以降でのみ可能、古いバージョンでは、"double" を使用します)
- "string"
- "array"
- "object"
- "null" (PHP 4.2.0以降)

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 settype() の例

```
<?php
$foo = "5bar"; // string
$bar = true;   // boolean

settype($foo, "integer"); // ここでは、$foo は 5です (整数)
settype($bar, "string");  // ここでは、$bar は "1" です (文字列)
?>
```

注意

注意: "int" の最大値は **PHP_INT_MAX** です。

参考

- gettype()
- 型キャスト
- 型の相互変換

---

strval

(PHP 4, PHP 5)

strval — 変数の文字列としての値を得ます

説明

string **strval** (mixed $var)

var の string としての値を 返します。文字列への変換の詳細については、string の ドキュメントを参照ください。

var は、全てのスカラー値にできます。 **strval()** に配列あるいはオブジェクトは使用できません。

参考

- floatval()
- intval()
- settype()
- 型の相互変換

---

unserialize

(PHP 4, PHP 5)

unserialize — 保存用表現から PHP の値を生成する

説明

mixed **unserialize** (string $str)

unserialize() は、シリアル化された変数を PHP 変数値に戻す変換を行います。

パラメータ

str

シリアル化された文字列。

もしアンシリアライズする変数がオブジェクトの場合、 オブジェクトが無事再作成された後、PHP は自動的にメンバ関数 **__wakeup()**（存在していれば）をコールしようとします。

> **注意**: **unserialize_callback_func ディレクティブ** コールバック関数を設定することが可能です。(不完全な object "__PHP_Incomplete_Class"を得ることを防ぐため) コールバック関数は、非シリアル化する際に未定義のクラスをインスタ ンス化する必要がある場合にコールされます。 'unserialize_callback_func'を定義するためには、 php.ini, ini_set(), .htaccess を使用し てください。未定義のクラスをインスタンス化する度に、コールバック関 数がコールされます。この機能を無効とするには、 単純にこの設定を空にしてください。

返り値

変換された値が返されます。その値は、 boolean, integer, float, string, array, object のいずれかとなります。

渡された文字列が復元できなかった場合、**FALSE** を返して E_NOTICE を発生します。

変更履歴

バージョン	説明
4.2.0	unserialize_callback_func ディレクティブが追加されました。
4.0.0	オブジェクトをシリアル化しても、メソッドが失われなくなりました。 詳細は オブジェクトのシリアル化 を参照ください。

例

Example#1 unserialize() の例

```
<?php
// ここで、データベースから $session_data にセッションデータをロード
// するために unserialize() を使用します。
// この例は、serialize() で記述された例を補足するものです。

$conn = odbc_connect("webdb", "php", "chicken");
$stmt = odbc_prepare($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array($_SERVER['PHP_AUTH_USER']);
if (!odbc_execute($stmt, &$sqldata) || !odbc_fetch_into($stmt, &$tmp)) {
    // 実行または取得が失敗した場合、空の配列で初期化します
    $session_data = array();
} else {
    // tmp[0] にシリアル化されたデータを保持している必要があります。
    $session_data = unserialize($tmp[0]);
    if (!is_array($session_data)) {
        // 何か問題があったため、空の配列で初期化します。
        $session_data = array();
    }
}
?>
```

Example#2 unserialize_callback_func の例

```
<?php
$serialized_object='O:1:"a":1:{s:5:"value";s:3:"100";}';

// unserialize_callback_func ディレクティブは PHP 4.2.0 以降で利用可能
ini_set('unserialize_callback_func', 'mycallback'); //  独自のコールバック関数を設定する

function mycallback($classname)
{
    // クラスが定義されているファイルをインクルードするだけです。
    // どのクラス定義が必要になるのかを $classname で判断します。
```

```
}
?>
```

注意

警告

エラーやシリアライズされた **FALSE** 値をアンシリアライズする場合、 **FALSE** が返されます。この特殊なケースは str を serialize(false) で比較する、もしくは E_NOTICE をキャッチすることで区別することができます。

参考

- serialize()

---

unset

(PHP 4, PHP 5)

unset — 指定した変数の割当を解除する

説明

void **unset** (mixed $var [, mixed $var [, mixed $...]])

unset() は指定した変数を破棄します。

関数 **unset()** の内部動作は、 破棄しようとする変数の型に依存します。

あるグローバル変数が関数の中で **unset()** された場合、ローカル変数のみが破棄されます。呼出側の環境にある変数は、 **unset()** がコールされる前と同じ値を保持します。

```
<?php
function destroy_foo()
{
    global $foo;
    unset($foo);
}

$foo = 'bar';
destroy_foo();
echo $foo;
?>
```

上の例の出力は以下となります。

```
bar
```

グローバル変数を関数の内部で **unset()** したい場合は、 $GLOBALS 配列を使用することが可能です。

```
<?php
function foo()
{
    unset($GLOBALS['bar']);
}

$bar = "something";
foo();
?>
```

参照渡しされた変数が関数内で **unset()** された場合に、 ローカル変数のみが破棄されます。呼出側の環境でその変数は、 **unset()** がコールされる前と同じ値を保持します。

```
<?php
function foo(&$bar)
{
    unset($bar);
    $bar = "blah";
}

$bar = 'something';
echo "$bar¥n";

foo($bar);
echo "$bar¥n";
?>
```

上の例の出力は以下となります。

```
something
something
```

静的変数が関数の内部で **unset()** された場合、 **unset()** は、その関数の残りのコンテキスト内においてのみ 変数を破棄します。関数を再度コールすると、破棄する前の値が復元されます。

```
<?php
function foo()
{
    static $bar;
```

```
    $bar++;
    echo "Before unset: $bar, ";
    unset($bar);
    $bar = 23;
    echo "after unset: $bar¥n";
}

foo();
foo();
foo();
?>
```

上の例の出力は以下となります。

```
Before unset: 1, after unset: 23
Before unset: 2, after unset: 23
Before unset: 3, after unset: 23
```

パラメータ

var

破棄する変数。

var

別の変数。

...

返り値

値を返しません。

変更履歴

バージョン	説明
4.0.0	**unset()** は、関数ではなく、式となりました (PHP 3 では、**unset()** は常に 1 を返していました)。

例

Example#1 unset() の例

```
<?php
// 変数を一つ破棄する
unset($foo);

// 配列の要素の一つを破棄する
unset($bar['quux']);

// 複数の変数を破棄する
unset($foo1, $foo2, $foo3);
?>
```

注意

注意: これは、関数ではなく言語構造のため、可変関数 を用いてコールすることはできません。

注意: 現在のコンテキストで見えるものであれば、 オブジェクトのプロパティでさえも破棄することが可能です。

注意: PHP 5 以降では、オブジェクトのメソッド内で $this を破棄することはできません。

参考

- isset()
- empty()
- array_splice()

---

var_dump

(PHP 4, PHP 5)

var_dump — 変数に関する情報をダンプする

説明

```
void var_dump ( mixed $expression [, mixed $expression [, $... ]] )
```

この関数は、指定した式に関してその型や値を含む構造化された情報を 返します。配列の場合、その構造を表示するために各値について再帰的に 探索されます。

PHP 5 では、オブジェクトのすべての public、private および protected なプロパティが出力されます。

ヒント

ブラウザに直接結果を出力するすべてのものと同様に、出力制御関数 を使用してこの関数の出力をキャプチャーし、(例えば)文字列 (string)に保存することが可能です。

パラメータ

expression

展開したい変数。

返り値

値を返しません。

例

Example#1 var_dump() の例

```
<?php
$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
?>
```

上の例の出力は以下となります。

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}
```

```
<?php

$b = 3.1;
$c = true;
var_dump($b, $c);

?>
```

上の例の出力は以下となります。

```
float(3.1)
bool(true)
```

参考

- var_export()
- print_r()

---

var_export

(PHP 4 >= 4.2.0, PHP 5)

var_export — 変数の文字列表現を出力または返す

説明

mixed **var_export** (mixed $expression [, bool $return])

var_export() は、 渡された変数に関する構造化された情報を返します。この関数は var_dump() に似ていますが、 返される表現が有効な PHP コードであるところが異なります。

パラメータ

expression

エクスポートしたい変数

return

使用されかつ **TRUE** に設定された場合、**var_export()** は変数表現を出力する代わりに返します。

> **注意**: この関数は、このパラメータに対して内部的に出力バッファリングを使用しています。そのため、ob_start() コールバック関数の中で使用することはできません。

返り値

return パラメータが使用され TRUE と評価される場合、 変数表現を返します。そうでない場合、この関数は NULL を返します。

変更履歴

バージョン	説明
5.1.0	マジックメソッド __set_state を使用することで、 クラスを含む配列やクラスをエクスポートできるようになりました。

例

Example#1 var_export() の例

```php
<?php
$a = array (1, 2, array ("a", "b", "c"));
var_export($a);
?>
```

上の例の出力は以下となります。

```
array (
  0 => 1,
  1 => 2,
  2 =>
  array (
    0 => 'a',
    1 => 'b',
    2 => 'c',
  ),
)
```

```php
<?php

$b = 3.1;
$v = var_export($b, true);
echo $v;

?>
```

上の例の出力は以下となります。

```
3.1
```

Example#2 PHP 5.1.0 以降でのクラスのエクスポート

```php
<?php
class A { public $var; }
$a = new A;
$a->var = 5;
var_export($a);
?>
```

上の例の出力は以下となります。

```
A::__set_state(array(
   'var' => 5,
))
```

Example#3 __set_state の使用法 (PHP 5.1.0 以降)

```php
<?php
class A
{
     public $var1;
     public $var2;

      public static function __set_state($an_array)
     {
          $obj = new A;
          $obj->var1 = $an_array['var1'];
          $obj->var2 = $an_array['var2'];
           return $obj;
     }
}

$a = new A;
$a->var1 = 5;
$a->var2 = 'foo';

eval('$b = ' . var_export($a, true) . ';'); // $b = A::__set_state(array(
                                            //      'var1' => 5,
                                            //      'var2' => 'foo',
                                            // ));
var_dump($b);
?>
```

上の例の出力は以下となります。

```
object(A)#2 (2) {
   ["var1"]=>
   int(5)
   ["var2"]=>
   string(3) "foo"
}
```

注意

注意: リソース型 の変数は、 この関数ではエクスポートする事ができません。

注意: **var_export()** では循環参照を扱うことができません。 循環参照を表す解析可能な PHP コードを生成することは、不可能に近いからです。 配列やオブジェクトを完全な形式で扱いたい場合は serialize() を使用してください。

参考

- print_r()
- serialize()
- var_dump()

目次

Verisign Payflow Pro 関数

導入

この拡張モジュールにより、以前は Signio として知られていた Verisign Payment Services (» http://www.verisign.com/products-services/payment-processing/online-payment/payflow-pro/index.html) を使用してクレジットカードおよび他の金融トランザクションを 処理することが可能になります。

これらの関数を使用する場合、pfpro_init() および pfpro_cleanup() のコールを省略することが可能です。 これは、この拡張モジュールが必要に応じて自動的にこれらをコールする ためです。しかし、複数のトランザクションを処理し、ライブラリ全体を 意のままに制御したい場合には、これらの関数を利用することが可能です。 これら二つの関数コールの間に、pfpro_process() を 使用してトランザクションを何回でも行うことが可能です。

これらの関数は、PHP 4.0.2 で追加されました。

> **注意**: これらの関数は、Verisign Payment Services へのリンクのみ提供します。 必要なパラメータの詳細については、Payflow Pro Developers Guide を 参照ください。
>
> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.0.
>
> **注意**: この拡張モジュールは Windows 環境では利用できません。

要件

使用するプラットフォーム用の適当な SDK が必要です。これは、登録後に » manager interface から ダウンロードすることが可能です。

SDK をダウンロードした後、配布ファイルのディレクトリ lib からファイルをコピーする必要があります。 ヘッダファイル pfpro.h を /usr/local/include に、 ライブラリファイル libpfpro.so を /usr/local/lib にコピーして ください。

または、Verisign から取得した tarball をひとつの場所に展開し、 ビルド設定時に --with-pfpro[=DIR] オプションを指定してその場所を参照することも可能です。

Example#1 明示的な設定

```
tar -zxf pfpro_sunsparc.tar.gz -C /usr/local/

./configure --with-pfpro=/usr/local/verisign/payflowpro/sunsparc
```

> **注意**: 上で示した例のパスの最後の部分、つまりここでは sunsparc は、Verisign SDK のビルド対象のアーキテクチャに依存します。

インストール手順

以下の関数は、PHP がオプション --with-pfpro[=DIR] を付けて コンパイルされている場合にのみ利用可能です。

警告

この拡張モジュールを OpenSSL 拡張モジュールあるいは ModSSL とともに使用することを考えている場合、--with-pfpro=shared,/usr/local を指定して、この拡張モジュールを共有モジュールとしてコンパイルする 必要があります。

実行時設定

php.ini の設定により動作が変化します。

Verisign Payflow Pro 設定オプション

名前	デフォルト	変更の可否	変更履歴
pfpro.defaulthost/PFPRO_VERSION < 3	"test.signio.com"	PHP_INI_ALL	
pfpro.defaulthost	"test-payflow.verisign.com"	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
pfpro.defaultport	"443"	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
pfpro.defaulttimeout	"30"	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
pfpro.proxyaddress	""	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
pfpro.proxyport	""	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
pfpro.proxylogon	""	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。
pfpro.proxypassword	""	PHP_INI_ALL	PHP 4.0.2 以降で使用可能です。PHP 5.1.0 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

---

pfpro_cleanup

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5)

pfpro_cleanup — Payflow Pro ライブラリをシャットダウンする

説明

bool **pfpro_cleanup** (void)

Used to shutdown the Payflow Pro library cleanly.

この関数は、トランザクションを 終了した後スクリプトが終了する前にコールしなければなりません。 しかしこの関数のコールは省略することができ、その場合 この拡張モジュールはスクリプト終了時に **pfpro_cleanup()** を自動的にコールします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- pfpro_init()

---

pfpro_init

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5)

pfpro_init — Payflow Pro ライブラリを初期化する

説明

bool **pfpro_init** (void)

Payflow Pro ライブラリを初期化するために使用します。

この関数のコールは省略することができます。 その場合、この拡張モジュールは最初のトランザクションの前に **pfpro_init()** を自動的にコールします。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- pfpro_cleanup()

---

pfpro_process_raw

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5)

pfpro_process_raw — Payflow Pro により素のトランザクションを処理する

説明

string **pfpro_process_raw** (string $parameters [, string $address [, int $port [, int $timeout [, string $proxy_address [, int $proxy_port [, string $proxy_logon [, string $proxy_password]]]]]]])

pfpro_process_raw() は、生のトランザクション 文字列を Payflow Pro で処理します。実際には、 これらのトランザクションのエンコーディング規則が非標準であるため、 pfpro_process() を代わりに使用するべきです。

> **注意**: 必要なパラメータおよびエンコーディング規約の詳細については、 Payflow Pro Developers Guide を参照ください。代わりに pfpro_process() を使用することを強く推奨します。

パラメータ

parameters

生のトランザクション要求を有する文字列です。

address

接続するホストを指定します。デフォルトは test.signio.com です。実際のトランザクションを処理するには、 これを connect.signio.com に変更します。

port

接続するポートを指定します。デフォルトは 443 で、 これは標準の SSL のポート番号です。

timeout

タイムアウト秒数を指定します。デフォルトは 30 秒です。 このタイムアウトは、プロセッサへのリンクが確立されてからのものであることに注意しましょう。 DNS やネットワークに問題がある場合、非常に長い時間がかかってしまうことがあります。

proxy_address

必要に応じて、SSL プロキシのホスト名を指定します。

proxy_port

必要に応じて、SSL プロキシのポート番号を指定します。

proxy_logon

必要に応じて、SSL プロキシにログオンする際の ID を指定します。

proxy_password

必要に応じて、SSL プロキシにログオンする際のパスワードを指定します。

返り値

生のレスポンスを文字列で返します。

例

Example#1 Payflow Pro の例

```
<?php

pfpro_init();

$response = pfpro_process_raw(
"USER=mylogin&PWD[5]=m&ndy&PARTNER=VeriSign&TRXTYPE=S" .
"&TENDER=C&AMT=1.50&ACCT=4111111111111111&EXPDATE=0904");

if (!$response) {
   die("Couldn't establish link to Verisign.¥n");
}

echo "Verisign raw response was " . $response;

pfpro_cleanup();

?>
```

参考

- *pfpro_process()*

---

pfpro_process

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5)

pfpro_process — Payflow Pro でトランザクションを処理する

説明

```
array pfpro_process ( array $parameters [, string $address [, int $port [, int $timeout [, string $proxy_address [, int $proxy_port [, string $proxy_logon [, string $proxy_password ]]]]]] )
```

***pfpro_process()** は、Payflow Pro でトランザクションを処理します。*

> ***注意**: 必要なパラメータの詳細については、Payflow Pro Developers Guide を参照ください。*

パラメータ

parameters

エンコードされてプロセッサに渡されたキーおよび値を有する連想配列です。

address

接続するホストを指定します。 デフォルトでは、この値は "test.signio.com" になっており、 実際のトランザクションを処理するためには "connect.signio.com" に変更したいと思うことでしょう。

port

接続するポートを指定します。デフォルトは 443、つまり標準 SSL ポートです。

timeout

使用されるタイムアウトを秒数で指定します。 この値のデフォルトは 30 秒です。タイムアウトは、プロセッサへの リンクが確立されてから計測が開始されるので、DNS またはネットワークの 問題によりスクリプトが非常に長い時間実行されたままになる 可能性があることに注意してください。

proxy_address

必要に応じて、SSL プロキシのホスト名を指定します。

proxy_port

必要に応じて、SSL プロキシのポート番号を指定します。

proxy_logon

必要に応じて、SSL プロキシにログオンする際の ID を指定します。

proxy_password

必要に応じて、SSL プロキシにログオンする際のパスワードを指定します。

返り値

レスポンスのキーおよび値を含む連想配列を返します。

例

Example#1 Payflow Pro の例

```
<?php

pfpro_init();

$transaction = array('USER'     => 'mylogin',
                     'PWD'      => 'mypassword',
                     'PARTNER'  => 'VeriSign',
                     'TRXTYPE'  => 'S',
                     'TENDER'   => 'C',
                     'AMT'      => 1.50,
                     'ACCT'     => '4111111111111111',
                     'EXPDATE'  => '0909'
                    );

$response = pfpro_process($transaction);

if (!$response) {
   die("Couldn't establish link to Verisign.¥n");
}

echo "Verisign response code was " . $response['RESULT'];
echo ", which means: " . $response['RESPMSG'] . "¥n";

echo "¥nThe transaction request: ";
print_r($transaction);

echo "¥nThe response: ";
print_r($response);

pfpro_cleanup();

?>
```

---

pfpro_version

(PHP 4 >= 4.0.2, PHP 5 <= 5.0.5)

pfpro_version — Payflow Pro ソフトウエアのバージョンを返す

説明

string **pfpro_version** (void)

pfpro_version() は、Payflow Pro ライブラリの バージョンを文字列で返します。この文書を書いている時点では、 このバージョンは L211 です。

返り値

バージョン番号を文字列で返します。

目次

---

vpopmail 関数

導入

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

この拡張モジュールは、PHP 4.3.0 以降 PHP 本体から別れ、 現在は » PECL にあります。

インストール手順

PHP 4 では、これらの関数は、PHP が --with-vpopmail[=DIR] を指定して 設定されている場合にのみ使用可能です。

---

vpopmail_add_alias_domain_ex

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_add_alias_domain_ex — 既存の仮想ドメインにエイリアスを追加する

説明

bool **vpopmail_add_alias_domain_ex** (string $olddomain , string $newdomain)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_add_alias_domain

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_add_alias_domain — 仮想ドメインへのエイリアスを追加する

説明

bool **vpopmail_add_alias_domain** (string $domain , string $aliasdomain)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_add_domain_ex

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_add_domain_ex — 新規に仮想ドメインを追加する

説明

bool **vpopmail_add_domain_ex** (string $domain , string $passwd [, string $quota [, string $bounce [, bool $apop]]])
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_add_domain

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_add_domain — 仮想ドメインを新たに追加する

説明

bool **vpopmail_add_domain** (string $domain , string $dir , int $uid , int $gid)
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_add_user

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_add_user — 指定した仮想ドメインに新規ユーザを追加する

説明

bool ***vpopmail_add_user*** *(string $user , string $domain , string $password [, string $gecos [, bool $apop]])*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_alias_add

(PHP 4 >= 4.0.7, PECL vpopmail:0.2)

vpopmail_alias_add — 仮想エイリアスを追加する

説明

bool ***vpopmail_alias_add*** *(string $user , string $domain , string $alias)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_alias_del_domain

(PHP 4 >= 4.0.7, PECL vpopmail:0.2)

vpopmail_alias_del_domain — あるドメインに関する仮想エイリアスを全て削除する

説明

bool ***vpopmail_alias_del_domain*** *(string $domain)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_alias_del

(PHP 4 >= 4.0.7, PECL vpopmail:0.2)

vpopmail_alias_del — あるユーザの仮想エイリアスを全て削除する

説明

bool ***vpopmail_alias_del*** *(string $user , string $domain)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_alias_get_all

(PHP 4 >= 4.0.7, PECL vpopmail:0.2)

vpopmail_alias_get_all — あるドメインに関するエイリアスを全て取得する

説明

array ***vpopmail_alias_get_all*** *(string $domain)*
警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_alias_get

(PHP 4 >= 4.0.7, PECL vpopmail:0.2)

vpopmail_alias_get — あるドメインに関するエイリアスを取得する

説明

array **vpopmail_alias_get** (string $alias , string $domain)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_auth_user

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_auth_user — ユーザ名/ドメイン/パスワードの認証を試みる

説明

bool **vpopmail_auth_user** (string $user , string $domain , string $password [, string $apop])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_del_domain_ex

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_del_domain_ex — 仮想ドメインを削除する

説明

bool **vpopmail_del_domain_ex** (string $domain)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_del_domain

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_del_domain — 仮想ドメインを削除する

説明

bool **vpopmail_del_domain** (string $domain)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_del_user

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_del_user — 仮想ドメインからユーザを削除する

説明

bool ***vpopmail_del_user*** *(string $user , string $domain)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_error

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_error — 直近の vpopmail エラーに関するエラーメッセージを取得する

説明

string ***vpopmail_error*** *(void)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_passwd

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_passwd — 仮想ユーザのパスワードを変更する

説明

bool ***vpopmail_passwd*** *(string $user , string $domain , string $password [, bool $apop])*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

vpopmail_set_user_quota

(PHP 4 >= 4.0.5, PECL vpopmail:0.2)

vpopmail_set_user_quota — 仮想ユーザの容量制限(クオータ)を設定する

説明

bool ***vpopmail_set_user_quota*** *(string $user , string $domain , string $quota)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

目次

- vpopmail_auth_user — ユーザ名/ドメイン/パスワードの認証を試みる
- vpopmail_del_domain_ex — 仮想ドメインを削除する
- vpopmail_del_domain — 仮想ドメインを削除する
- vpopmail_del_user — 仮想ドメインからユーザを削除する
- vpopmail_error — 直近の vpopmail エラーに関するエラーメッセージを取得する
- vpopmail_passwd — 仮想ユーザのパスワードを変更する
- vpopmail_set_user_quota — 仮想ユーザの容量制限(クオータ)を設定する

W32api 関数

導入

この拡張モジュールは、DLL への一般的な拡張 API です。この拡張モジュールは もともと PHP から Win32 API にアクセスできるようにするために 作成されましたが、他の DLL によりエクスポートされている他の関数にも アクセスすることができます。

現在サポートされている型は、一般の PHP 型 (文字列, boolean, float, 整数, NULL) と w32api_deftype() で定義した型です。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。
> PHP 5.1.0.

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

要件

この拡張モジュールは Windows システムでのみ動作します。

インストール手順

PHP コアに含まれるため、追加のインストール無しで使用できます。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールは 1 種類のリソース型を定義し、ユーザ定義型で 使用されます。このリソースの名前は "dynaparm" です。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

DC_MICROSOFT (integer)
DC_BORLAND (integer)
DC_CALL_CDECL (integer)
DC_CALL_STD (integer)
DC_RETVAL_MATH4 (integer)
DC_RETVAL_MATH8 (integer)
DC_CALL_STD_BO (integer)
DC_CALL_STD_MS (integer)
DC_CALL_STD_M8 (integer)
DC_FLAG_ARGPTR (integer)

例

以下の例は、システムの実行時間を取得し、メッセージボックスに 表示するものです。

Example#1 uptime を取得し、メッセージボックスに表示する

```
<?php
// 必要な定数を定義します。これは
//  Visual  Studio/Tools/Winapi/WIN32API.txt  から取得できます。
define("MB_OK", 0);

// 拡張モジュールを読み込みます。
dl("php_w32api.dll");

//  kernel32.dll  の  GetTickCount  関数を登録します。
w32api_register_function("kernel32.dll",
                         "GetTickCount",
                         "long");

// User32.dll  の  MessageBoxA  関数を登録します。
w32api_register_function("User32.dll",
                         "MessageBoxA",
                         "long");

// uptime 情報を取得します。
$ticks  =  GetTickCount();

// それを読みやすい形式に変換します。
```

```
$secs  = floor($ticks / 1000);
$mins  = floor($secs / 60);
$hours = floor($mins / 60);

$str = sprintf("You have been using your computer for:" .
               "¥r¥n %d Milliseconds, or ¥r¥n %d Seconds" .
               "or ¥r¥n %d mins or¥r¥n %d hours %d mins.",
               $ticks,
               $secs,
               $mins,
               $hours,
               $mins - ($hours*60));

// OK ボタンと uptime のみのメッセージボックスを表示します。
MessageBoxA(NULL,
            $str,
            "Uptime Information",
            MB_OK);
?>
```

---

w32api_deftype

(PHP 4 >= 4.2.0)

w32api_deftype — 他の w32api_functions で使用するために型を定義する

説明

bool **w32api_deftype** (string $typename , string $member1_type , string $member1_name [, string $... [, string $...]])

w32api のコールのために型を定義したい場合には、 この関数をコールする必要があります。

パラメータ

typename

型の名前。

member1_type

メンバの型はユーザ定義型とすることが可能です。 型の名前は大文字小文字を区別します。 組み込みの型の名前はすべて小文字となります。

member1_name

member1_type のメンバ名。

...

...

この関数は 2n+1 個の引数をとります。n はその型が保持するメンバの 数です。最初の引数が型の名前となります。それ以降には、メンバの型と その名前が（対になって）続きます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

w32api_init_dtype

(PHP 4 >= 4.2.0)

w32api_init_dtype — データ型 typename のインスタンスを作成し、そこに渡された値を代入する

説明

resource **w32api_init_dtype** (string $typename , mixed $value [, mixed $...])

この関数は、typename という型のデータのインスタンスを作成し、値を代入します。

パラメータ

typename

パラメータ typename は大文字小文字を区別します。

value

w32api_deftype() で定義したのと同じ順番で値を指定する必要があります。

...

返り値

dynaparm リソースを返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

w32api_invoke_function

(PHP 4 >= 4.2.0)

w32api_invoke_function — 関数名の後ろに指定された引数を指定し、関数 funcname を実行する

説明

mixed **w32api_invoke_function** (string $funcname , mixed $argument [, mixed $...])

w32api_invoke_function() は、 funcname という名前の事前に登録された関数を探し、 指定したパラメータを渡します。

パラメータ

funcname

関数の名前。

argument

引数の型は PHP 型のいずれかか、必要に応じて w32api_deftype() で定義された型となります。

...

返り値

返り値の型は、その関数を登録した際に定義したもので、 値はその関数が返す値そのものとなります。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

w32api_register_function

(PHP 4 >= 4.2.0)

w32api_register_function — ライブラリの関数 function_name を PHP に登録する

説明

bool **w32api_register_function** (string $library , string $function_name , string $return_type)

この関数は、ライブラリ libary から function_name という名前の関数を探し、PHP にインポートしようとします。

パラメータ

library

ライブラリ名を表す文字列。

function_name

関数名を表す文字列。

return_type

関数は、指定した return_type で登録されます。 この型は、PHP の組み込み型かあるいは w32api_deftype() で定義した型とすることが可能です。 型の名前は大文字小文字を区別します。組み込みの型の名前は すべて小文字となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

---

w32api_set_call_method

(PHP 4 >= 4.2.0)

w32api_set_call_method — 使用するコール方式を設定する

説明

void **w32api_set_call_method** (int $method)

この関数は、メソッドのコール方式を設定します。

パラメータ

method

DC_CALL_CDECL あるいは **DC_CALL_STD** のいずれかを指定します。 デフォルト値は **DC_CALL_STD** です。

返り値

値を返しません。

注意

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

目次

WDDX 関数

導入

以下の関数は、» WDDX と組み合わせて 動作することを想定しています。

要件

WDDX を使用するには、(Apache 1.3.7 以降に付属する) expat ライブラリを インストールする必要があります。

インストール手順

expat をインストールした後、 --enable-wddx を指定して PHP を コンパイルする必要があります。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールでは、WDDX パケット ID を定義しています。これは wddx_packet_start() が返すものです。

定義済み定数

定数は定義されていません。

例

変数をシリアル化する全ての関数は配列の最初要素をその配列が配列と 構造体のどちらでシリアル化されるのかを定義するために使用するということに 注意してください。最初の要素が文字列をキーとして有する場合は 構造体でシリアル化され、その他の場合は配列でシリアル化されます。

Example#1 WDDX を使用した単一の値のシリアル化

```
<?php
echo wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

この例は次の出力を行います。

```
<wddxPacket version='1.0'><header comment='PHP packet'/><data>
<string>PHP to WDDX packet example</string></data></wddxPacket>
```

Example#2 WDDX を使用してパケットを追加する例

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* $cities はデータベースから取得するものと仮定します */
$cities = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "cities");

$packet = wddx_packet_end($packet_id);
echo $packet;
?>
```

この例は次のような出力を行います。

```
<wddxPacket version='1.0'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='cities'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

> **注意**: ASCII以外の文字をシリアル化したい場合、まず最初に データを UTF-8 に変換する必要があります (utf8_encode() および iconv() を参照ください)。

---

wddx_add_vars

(PHP 4, PHP 5)

wddx_add_vars — 指定した ID の WDDX パケットを追加する

説明

bool **wddx_add_vars** (resource $packet_id , mixed $var_name [, mixed $...])

渡された変数をシリアライズし、 指定したパケットに結果を追加します。

パラメータ

この関数は、可変長のパラメータを受け取ります。

packet_id

wddx_packet_start() が返す WDDX パケット。

var_name

変数名を表す文字列、あるいは配列。 配列の中身は、変数名を表す文字列あるいは別の配列などとなります。

...

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

wddx_deserialize

(PHP 4, PHP 5)

wddx_deserialize — wddx_unserialize() のエイリアス

説明

この関数は次の関数のエイリアスです。 wddx_unserialize()

---

wddx_packet_end

(PHP 4, PHP 5)

wddx_packet_end — 指定した ID の WDDX パケットを終了する

説明

string **wddx_packet_end** (resource $packet_id)

指定した WDDX パケットを終了し、 パケットを含む文字列を返します。

パラメータ

packet_id

wddx_packet_start() が返す WDDX パケット。

返り値

WDDX パケットを含む文字列を返します。

---

wddx_packet_start

(PHP 4, PHP 5)

wddx_packet_start — 新規の WDDX パケットを内部の構造体を用いて開始する

説明

resource **wddx_packet_start** ([string $comment])

変数を順に追加していくための新しい WDDX パケットを開始します。 この関数は、パケットの内部に自動的に構造体の定義を作成し、 そこに変数を保持します。

パラメータ

comment

オプションのコメント文字列。

返り値

後で他の関数で使用するためのパケット ID、あるいはエラー時に **FALSE** を返します。

---

wddx_serialize_value

(PHP 4, PHP 5)

wddx_serialize_value — 単一の値を WDDX パケットにシリアライズする

説明

string **wddx_serialize_value** (mixed $var [, string $comment])

指定した単一の値をもとにして WDDX パケットを作成します。

パラメータ

var

シリアライズする値。

comment

オプションのコメント文字列。これはヘッダ内で用いられます。

返り値

WDDX パケット、あるいはエラー時に **FALSE** を返します。

---

wddx_serialize_vars

(PHP 4, PHP 5)

wddx_serialize_vars — 変数を WDDX パケットにシリアライズする

説明

string **wddx_serialize_vars** (mixed $var_name [, mixed $...])

指定された変数をシリアライズしたものを含む構造体を用いて WDDX パケットを作成します。

パラメータ

この関数は、可変長のパラメータを受け取ります。

var_name

変数名を表す文字列、あるいは配列。 配列の中身は、変数名を表す文字列あるいは別の配列などとなります。

...

返り値

WDDX パケット、あるいはエラー時に **FALSE** を返します。

例

Example#1 wddx_serialize_vars() の例

```
<?php
$a = 1;
$b = 5.5;
$c = array("blue", "orange", "violet");
$d = "colors";

$clvars = array("c", "d");
echo wddx_serialize_vars("a", "b", $clvars);
?>
```

上の例の出力は以下となります。

```
<wddxPacket version='1.0'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>blue</string><string>orange</string><string>violet</string></array></var>
<var name='d'><string>colors</string></var></struct></data></wddxPacket>
```

---

wddx_unserialize

(No version information available, might be only in CVS)

wddx_unserialize — シリアライズされた WDDX パケットを元に戻す

説明

mixed **wddx_unserialize** (string $packet)

シリアライズされた WDDX パケット packet を元に戻します。

パラメータ

packet

WDDX パケットを表す文字列あるいはストリーム。

返り値

シリアライズされたパケットを元に戻したものを返します。 これは文字列、数字、または配列となります。 シリアライズされた構造体を復元すると、連想配列となることに注意しましょう。

目次

---

win32ps 関数

導入

win32ps は Windows 限定の拡張モジュールで、PHP によって プロセスやメモリの使用状況を取得することを可能とします。

要件

Windows NT、Windows 2000、Windows XP あるいは Windows Server 2003 が必要です。Windows NT 系のすべての Windows で動作します。

インストール手順

PECL からのインストール

1. » pecl4win.php.net から php_win32ps.dll をダウンロードします。
2. php_win32ps.dll を、 extension_dir にコピーします。
3. php.ini で、この拡張モジュールを読み込みます。

```
extension=php_win32ps.dll
```

例

Example#1 現在の PHP プロセスについての統計情報

```
<?php
print_r(win32_ps_stat_proc());
/*
    Array
    (
        [pid] => 936
          [exe] => D:¥Daten¥Source¥php-5.1¥Debug_TS¥php.exe
        [mem] => Array
            (
                [page_fault_count] => 2062
                [peak_working_set_size] => 8396800
                [working_set_size] => 8396800
                [quota_peak_paged_pool_usage] => 32080
                [quota_paged_pool_usage] => 31876
                [quota_peak_non_paged_pool_usage] => 4240
                [quota_non_paged_pool_usage] => 3888
                [pagefile_usage] => 5865472
                [peak_pagefile_usage] => 5865472
            )

        [tms] => Array
            (
                [created] => 0.093
                [kernel] => 0.015
                [user] => 0.062
            )

    )
*/
?>
```

Example#2 グローバルメモリの使用状況

```
<?php
print_r(win32_ps_stat_mem());
/*
    Array
    (
        [load] => 37
        [unit] => 1024
         [total_phys] => 1048096
         [avail_phys] => 649960
         [total_pagefile] => 2521368
         [avail_pagefile] => 2237940
         [total_virtual] => 2097024
         [avail_virtual] => 2057848
    )
*/
?>
```

---

win32_ps_list_procs

(PECL win32ps:1.0.1)

win32_ps_list_procs — 稼動中のプロセスの一覧を取得する

説明

array **win32_ps_list_procs** (void)

稼動しているすべてのプロセスについての統計データを取得します。

返り値

失敗した場合、**FALSE** を返します。 成功した場合、実行中のすべてのプロセスについての統計情報を 含む配列を返します。統計情報の形式は win32_ps_stat_proc() が返すものと同じです。

参考

- win32_ps_stat_proc()

---

win32_ps_stat_mem

(PECL win32ps:1.0.1)

win32_ps_stat_mem — メモリ使用量の統計情報を取得する

説明

array **win32_ps_stat_mem** (void)

グローバルメモリの使用状況についての統計情報を取得します。

返り値

失敗した場合は、**FALSE** を返します。 成功した場合は、以下の情報を含む配列を返します。

load

現在のメモリ使用量の、物理メモリに対するパーセンテージ。

unit

これは常に *1024* で、以下の値が *1024* バイト単位であることを示します。

total_phys

物理メモリの総容量。

avail_phys

物理メモリの残容量。

total_pagefile

ページング可能メモリ（物理メモリ + ページングファイル）の総容量。

avail_pagefile

ページング可能メモリ（物理メモリ + ページングファイル）の残容量。

total_virtual

プロセスに対する仮想メモリの総容量。

avail_virtual

プロセスに対する仮想メモリの残容量。

---

win32_ps_stat_proc

(PECL win32ps:1.0.1)

win32_ps_stat_proc — プロセスの統計情報を取得する

説明

array **win32_ps_stat_proc** *([int $pid])*

プロセス *ID pid* のプロセスについての 統計情報を取得します。

パラメータ

pid

統計情報を取得するプロセスの *ID*。 指定しなかった場合は、現在のプロセスの *ID* が使用されます。

返り値

失敗した場合は、**FALSE** を返します。 成功した場合は、以下の情報を含む配列を返します。

pid

プロセス *ID*。

exe

実行イメージへのパス。

mem

メモリの使用量を示す以下の情報を含む配列。 *page_fault_count* 、 *peak_working_set_size* 、 *working_set_size* 、 *quota_peak_paged_pool_usage* 、 *quota_paged_pool_usage* 、 *quota_peak_non_paged_pool_usage* 、 *quota_non_paged_pool_usage* 、 *pagefile_usage* および *peak_pagefile_usage* 。

tms

CPU 時間の使用状況を示す以下の情報を含む配列。 *created* 、 *kernel* および *user* 。

参考

- *win32_ps_list_procs()*

目次

win32service 関数

導入

win32service は Windows 限定の拡張モジュールです。サービスコントロールマネージャと 連携してサービスの開始・停止・登録・解除を行ったり、PHP スクリプトを サービスとして実行させる機能を提供します。

要件

Windows NT、Windows 2000、Windows XP あるいは Windows Server 2003 が必要です。Windows NT 系の Windows に対応しています。

インストール手順

PECL からのインストール

1. http://snaps.php.net/win32/ から php_win32service.dll がダウンロード可能です。 PHP のバージョンに一致する PECL_X_X フォルダを選択します。
2. php_win32service.dll を extension_dir にコピーします。
3. php.ini で拡張モジュールを読み込みます。

   ```
   extension=php_win32service.dll
   ```

例

Example#1 PHP スクリプトをサービスとして登録する

```
<?php
win32_create_service(array(
        'service' => 'dummyphp',                # サービスの名前
        'display' => 'sample dummy PHP service', # 説明
        'params' => 'c:¥path¥to¥script.php run', # スクリプトへのパスとパラメータ
    ));
?>
```

Example#2 サービスの登録を解除する

```
<?php
win32_delete_service('dummyphp');
?>
```

Example#3 サービスとして実行する

```
<?php
if ($argv[1] == 'run') {
  win32_start_service_ctrl_dispatcher('dummyphp');

  while (WIN32_SERVICE_CONTROL_STOP  != win32_get_last_control_message()) {
    # ここに実際の作業内容を書きます。
    # 1 回のループに 30 秒以上かからないように心がけてください。
  }
}
?>
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

WIN32_SERVICE_CONTROL_CONTINUE (integer)
WIN32_SERVICE_CONTROL_INTERROGATE (integer)
WIN32_SERVICE_CONTROL_PAUSE (integer)
WIN32_SERVICE_CONTROL_STOP (integer)
WIN32_SERVICE_CONTROL_HARDWAREPROFILECHANGE (integer)
WIN32_SERVICE_CONTROL_POWEREVENT (integer)
WIN32_SERVICE_CONTROL_SESSIONCHANGE (integer)
WIN32_ERROR_CALL_NOT_IMPLEMENTED (integer)
WIN32_NO_ERROR (integer)
WIN32_SERVICE_RUNNING (integer)
WIN32_SERVICE_STOPPED (integer)
WIN32_SERVICE_STOP_PENDING (integer)
WIN32_SERVICE_WIN32_OWN_PROCESS (integer)
WIN32_SERVICE_INTERACTIVE_PROCESS (integer)
WIN32_SERVICE_STOPPED (integer)
WIN32_SERVICE_START_PENDING (integer)
WIN32_SERVICE_STOP_PENDING (integer)
WIN32_SERVICE_RUNNING (integer)
WIN32_SERVICE_CONTINUE_PENDING (integer)
WIN32_SERVICE_PAUSE_PENDING (integer)
WIN32_SERVICE_PAUSED (integer)
WIN32_SERVICE_ACCEPT_NETBINDCHANGE (integer)
WIN32_SERVICE_ACCEPT_PARAMCHANGE (integer)
WIN32_SERVICE_ACCEPT_PAUSE_CONTINUE (integer)
WIN32_SERVICE_ACCEPT_SHUTDOWN (integer)
WIN32_SERVICE_ACCEPT_STOP (integer)
WIN32_SERVICE_ACCEPT_HARDWAREPROFILECHANGE (integer)
WIN32_SERVICE_ACCEPT_POWEREVENT (integer)

WIN32_SERVICE_ACCEPT_SESSIONCHANGE (integer)
WIN32_SERVICE_FILE_SYSTEM_DRIVER (integer)
WIN32_SERVICE_KERNEL_DRIVER (integer)
WIN32_SERVICE_WIN32_SHARE_PROCESS (integer)
WIN32_SERVICE_RUNS_IN_SYSTEM_PROCESS (integer)

---

win32_create_service

(No version information available, might be only in CVS)

win32_create_service — SCM データベースに新しいサービスのエントリを作成する

説明

mixed **win32_create_service** (array $details [, string $machine])

パラメータ

details

サービスの詳細情報を含む配列。

service

サービスの短い名前。net コマンドでサービスを 制御する際に、この名前を使用します。この名前は一意である（同名の サービスが 2 つ存在することがない）必要があり、スペースを含む 名前は可能な限り避けるべきです。

display

サービスの表示名。これは、サービスアプレットに表示される名前です。

user

サービスを実行するユーザ名。指定しなかった場合、サービスは LocalSystem アカウントで実行されます。ユーザ名が指定された場合、password も指定する必要があります。

password

user に対応するパスワード。

path

サービスの開始時に起動される実行モジュールのフルパス。 指定しなかった場合、現在の PHP プロセスへのパスが使用されます。

params

サービスの開始時に渡されるコマンドラインパラメータ。 PHP スクリプトをサービスとして実行したい場合は、最初のパラメータは 実行するスクリプトへのフルパスとなります。

load_order

load_order を制御します。現時点では完全にはサポートされていません。

svc_type

サービスの型を指定します。指定しなかった場合、デフォルト値は **WIN32_SERVICE_WIN32_OWN_PROCESS** です。 よくわからない場合はこの値を変更しないでください。

start_type

サービスをどのように開始させるかを指定します。デフォルトは **WIN32_SERVICE_AUTO_START** で、これは マシンの起動時にサービスを開始させることを意味します。

error_control

サービスに問題が発生した際にとるべき行動を SCM に指示します。 デフォルトは **WIN32_SERVER_ERROR_IGNORE** です。 この値を変更することは、現時点では完全にはサポートされていません。

machine

オプションで、サービスを作成したいマシン名を指定します。 指定しなかった場合は、ローカルマシンを使用します。

返り値

成功した場合に **TRUE** 、それ以外の場合に win32 エラーコードを返します。

例

Example#1 win32_create_service() の例

```
<?php
$x = win32_create_service(array(
        'service'  => 'dummyphp',
        'display'  => 'sample dummy PHP service',
        'params'  => __FILE__ . ' run',
));
debug_zval_dump($x);
?>
```

参考

- win32_delete_service()

win32_delete_service

(No version information available, might be only in CVS)

win32_delete_service — SCM データベースからサービスのエントリを削除する

説明

int **win32_delete_service** (string $servicename [, string $machine])

SCM データベースからサービスの削除を試みます。管理者権限が必要です。

この関数は、単にサービスに削除マークをつけるだけです。もし他のプロセス（サービスアプレットなど）が開かれた場合、削除処理はそのアプリケーションが 終了するまで延期されます。サービスに削除マークがつけられると、それ以降は 同じサービスに対する削除の試行は失敗します。また同名の新規サービスを作成 しようとする処理も失敗します。

パラメータ

servicename

サービスの短い名前。

machine

オプションのマシン名。指定しなかった場合、ローカルマシンが使用されます。

返り値

成功した場合に **TRUE** 、失敗した場合に win32 エラーコードを返します。

例

Example#1 win32_delete_service() の例

dummyphp サービスを削除します。

```
<?php
win32_delete_service('dummyphp');
?>
```

win32_get_last_control_message

(No version information available, might be only in CVS)

win32_get_last_control_message — サービスに送信された直近の制御メッセージを返す

説明

int **win32_get_last_control_message** (void)

サービスプロセスに送信された、直近の制御コードを返します。 サービスの実行中は、サービスを停止させる必要がないかを調べるために 定期的にこれをチェックすべきです。

返り値

制御コード定数を返します。以下のうちのひとつです。 **WIN32_SERVICE_CONTROL_CONTINUE**、 **WIN32_SERVICE_CONTROL_INTERROGATE**、**WIN32_SERVICE_CONTROL_PAUSE**、 **WIN32_SERVICE_CONTROL_STOP**、 **WIN32_SERVICE_CONTROL_HARDWAREPROFILECHANGE**、**WIN32_SERVICE_CONTROL_POWEREVENT**、 **WIN32_SERVICE_CONTROL_SESSIONCHANGE**。

参考

- win32_start_service_ctrl_dispatcher()

win32_query_service_status

(No version information available, might be only in CVS)

win32_query_service_status — サービスの状態を問い合わせる

説明

mixed **win32_query_service_status** (string $servicename [, string $machine])

サービスの稼動状況を問い合わせ、情報を配列で返します。

パラメータ

servicename

サービスの短い名前。

machine

オプションのマシン名。指定しなかった場合、ローカルマシンが使用されます。

返り値

失敗した場合に ***FALSE*** *、それ以外の場合に以下の情報を含む配列を返します。*

ServiceType

dwServiceType

CurrentState

dwCurrentState

ControlsAccepted

サービスが許可しているコントロール。

Win32ExitCode

サービスが終了した際にプロセスが返すコード。

ServiceSpecificExitCode

サービスが異常終了した際にイベントログに記録されるコード。

CheckPoint

サービスが終了する際に、現在のチェックポイント番号を保持します。 SCM はこれをいわゆる心拍のように扱い、反応しなくなったサービスを 検出するために使用します。この値は、WaitHint の値と組み合わせて 用いられます。

WaitHint

サービスが終了する際に、WaitHint を checkpoint の値に設定します。 この処理が終了した時点でサービスの終了処理が 100% 完了したことを示します。 これは、進捗状況のインジケータを実装する際に使用します。

ProcessId

Windows のプロセス ID 。0 の場合、プロセスは実行していません。

ServiceFlags

dwServiceFlags

---

win32_set_service_status

(No version information available, might be only in CVS)

win32_set_service_status — サービスの状態を更新する

説明

bool ***win32_set_service_status*** *(int $status)*

SCM に、実行中のサービスの現在の状態を通知します。 この関数のコールは、現在実行中のサービスについてのみ有効です。

パラメータ

status

以下のステータスコードのうちのひとつ。 ***WIN32_SERVICE_RUNNING***、 ***WIN32_SERVICE_STOPPED***、 ***WIN32_SERVICE_STOP_PENDING***、 ***WIN32_SERVICE_START_PENDING***、 ***WIN32_SERVICE_CONTINUE_PENDING***、 ***WIN32_SERVICE_PAUSE_PENDING***、 ***WIN32_SERVICE_PAUSED***。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *win32_start_service_ctrl_dispatcher()*

---

win32_start_service_ctrl_dispatcher

(No version information available, might be only in CVS)

win32_start_service_ctrl_dispatcher — スクリプトを SCM に登録し、指定した名前でサービスとして稼動させる ようにする

説明

mixed ***win32_start_service_ctrl_dispatcher*** *(string $name)*

サービスコントロールマネージャ経由で起動させる際、サービスプロセスは 「チェックイン」を要求され、これによってサービスのモニタリングや 通信の機能を確立します。この関数は、サービスコントロールマネージャとの 低レベル通信を処理するスレッドを生成し、チェックインを実行します。

サービスの開始後は、常にサービスコントロールマネージャにチェックイン し続け、サービスをいつ終了させるべきかを確認するべきです。 これは、

定期的に win32_get_last_control_message() をコールして返り値を適切に処理することで実現できます。 code appropriately.

パラメータ

name

サービスの短い名前。win32_create_service() で登録されたもの。

返り値

成功した場合に TRUE、それ以外の場合は FALSE か win32 エラーコードを返します。

例

Example#1 win32_start_service_ctrl_dispatcher() の例

```
<?php
if (!win32_start_service_ctrl_dispatcher('dummyphp')) {
  die("I'm probably not running under the service control manager");
}

while (WIN32_SERVICE_CONTROL_STOP != win32_get_last_control_message()) {
  # ここでなんらかの処理をします。1 回のループに 30 秒以上かからない
  # ように心がけてください。
}
?>
```

参考

- win32_get_last_control_message()

---

win32_start_service

(No version information available, might be only in CVS)

win32_start_service — サービスを開始する

説明

int **win32_start_service** (string $servicename [, string $machine])

指定したサービスの開始を試みます。通常は管理者権限を必要とします。

パラメータ

servicename

サービスの短い名前。

machine

オプションのマシン名。指定しなかった場合、ローカルマシンが使用されます。

返り値

成功した場合に WIN32_NO_ERROR 、失敗した場合に それ以外の win32 エラーコードを返します。

参考

- win32_stop_service()

---

win32_stop_service

(No version information available, might be only in CVS)

win32_stop_service — サービスを停止する

説明

int **win32_stop_service** (string $servicename [, string $machine])

サービスを停止します。管理者権限を必要とします。

パラメータ

servicename

サービスの短い名前。

machine

オプションのマシン名。指定しなかった場合、ローカルマシンが使用されます。

返り値

成功した場合に **WIN32_NO_ERROR** 、失敗した場合に それ以外の win32 エラーコードを返します。

参考

- win32_start_service()

目次

---

xattr 関数

導入

xattr 拡張モジュールは、ファイルシステムの拡張属性を操作します。

要件

xattr を使用するには、libattr がインストールされている必要があります。 これは » http://oss.sgi.com/projects/xfs/ から取得可能です。

> **注意**: これらの関数は、ファイルシステムが拡張属性をサポートしており、 マウント時にそれを有効にしている場合にのみ動作します。一般に使用されているファイルシステムのうち、拡張属性を サポートしているものには ext2、ext3、reiserfs、jfs および xfs などがあります。

インストール手順

xattr は、現在 PECL » http://pecl.php.net/package/xattr から取得可能です。

*nix 系のシステムを使用しており » PEAR が使用可能な場合は、pear インストーラを使用して xattr 拡張モジュールを インストールすることができます。そのためには以下のコマンド **pear -v install xattr** を使用します。

tar.gz パッケージをダウンロードし、手動でインストールすることも可能です。

Example#1 手動での xattr のインストール

```
gunzip  xattr-xxx.tgz
tar  -xvf  xattr-xxx.tar
cd  xattr-xxx
phpize
./configure  &&  make  &&  make  install
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

XATTR_ROOT (integer)
root (信頼された) 名前空間に属性を設定します。 root 権限が必要です。
XATTR_DONTFOLLOW (integer)
シンボリックリンクのリンク先をたどらず、リンクそのものを操作します。
XATTR_CREATE (integer)
拡張属性が既に存在する場合、関数は失敗します。
XATTR_REPLACE (integer)
拡張属性が存在しない場合、関数は失敗します。

---

xattr_get

(PECL xattr:0.9-1.0)

xattr_get — 拡張属性を取得する

説明

string **xattr_get** (string $filename , string $name [, int $flags])

この関数は、ファイルの拡張属性の値を取得します。

拡張属性には二種類の異なる名前空間、つまり、ユーザとルートがあります。ユーザ名前空間は、すべてのユーザで利用可能ですが、ルート名前空間は、ルート権限を有するユーザのみ利用可能です。 xattr はデフォルトでユーザ名前空間で処理を行いますが、 flags 引数によりこれを変更することができます。

パラメータ

filename

属性を取得するファイル。

name

属性の名前。

flags

サポートされる xattr のフラグ

XATTR_DONTFOLLOW	シンボリックリンクのリンク先をたどらず、リンクそのものを操作します。
XATTR_ROOT	root（信頼された）名前空間に属性を設定します。root 権限が必要です。

返り値

値を含む文字列、あるいは属性が存在しない場合には **FALSE** を返します。

例

Example#1 システム管理者がファイルに署名をしているかどうかを調べる

```
<?php
$file = '/usr/local/sbin/some_binary';
$signature = xattr_get($file, 'Root signature', XATTR_ROOT);

/* ... $signature が正しいものかどうかを調べます ... */

?>
```

参考

- xattr_list()
- xattr_set()
- xattr_remove()

---

xattr_list

(PECL xattr:0.9-1.0)

xattr_list — 拡張属性の一覧を取得する

説明

array **xattr_list** (string $filename [, int $flags])

この関数は、ファイルの拡張属性の名前の一覧を取得します。

拡張属性には二種類の異なる名前空間、つまり、ユーザとルートがあります。ユーザ名前空間は、すべてのユーザで利用可能ですが、ルート名前空間は、ルート権限を有するユーザのみ利用可能です。 xattr はデフォルトでユーザ名前空間で処理を行いますが、 flags 引数によりこれを変更することができます。

パラメータ

filename

ファイルのパス。

flags

サポートされる xattr のフラグ

XATTR_DONTFOLLOW	シンボリックリンクのリンク先をたどらず、リンクそのものを操作します。
XATTR_ROOT	root（信頼された）名前空間に属性を設定します。root 権限が必要です。

返り値

この関数は、拡張属性の名前を配列で返します。

例

Example#1 ファイルのすべての拡張属性の名前を表示する

```
<?php
$file = 'some_file';
$root_attributes = xattr_list($file, XATTR_ROOT);
$user_attributes = xattr_list($file);

echo "Root 属性: \n";
foreach ($root_attributes as $attr_name) {
    printf("%s¥n", $attr_name);
}

echo "\n User 属性: \n";
foreach ($attributes as $attr_name) {
```

```
    printf("%s¥n", $attr_name);
}

?>
```

参考

- xattr_get()

---

xattr_remove

(PECL xattr:0.9-1.0)

xattr_remove — 拡張属性を削除する

説明

bool **xattr_remove** (string $filename , string $name [, int $flags])

この関数は、ファイルの拡張属性を削除します。

拡張属性には二種類の異なる名前空間、つまり、ユーザとルートがあります。ユーザ名前空間は、すべてのユーザで利用可能ですが、ルート名前空間は、ルート権限を有するユーザのみ利用可能です。 xattr はデフォルトでユーザ名前空間で処理を行いますが、 flags 引数によりこれを変更することができます。

パラメータ

filename

属性を削除するファイル。

name

削除する属性の名前。

flags

サポートされる xattr のフラグ

XATTR_DONTFOLLOW	シンボリックリンクのリンク先をたどらず、リンクそのものを操作します。
XATTR_ROOT	root （信頼された） 名前空間に属性を設定します。root 権限が必要です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 ファイルのすべての拡張属性を削除する

```
<?php
$file = 'some_file';
$attributes = xattr_list($file);

foreach ($attributes as $attr_name) {
    xattr_remove($file, $attr_name);
}
?>
```

参考

- xattr_list()
- xattr_set()
- xattr_get()

---

xattr_set

(PECL xattr:0.9-1.0)

xattr_set — 拡張属性を設定する

説明

bool **xattr_set** (string $filename , string $name , string $value [, int $flags])

この関数は、ファイルの拡張属性の値を設定します。

拡張属性には二種類の異なる名前空間、つまり、ユーザとルートがあります。ユーザ名前空間は、すべてのユーザで利用可能ですが、ルート名前空間は、ルート権限を有するユーザのみ利用可能です。 xattr はデフォルトでユーザ名前空間で処理を行いますが、 flags 引数によりこれを変更することができます。

パラメータ

filename

属性を設定するファイル。

name

拡張属性の名前。もし属性が存在しない場合は新しく作成され、 存在する場合は上書きされます。この振る舞いは、 flags パラメータを使用することで変更可能です。

value

属性の値。

flags

サポートされる xattr のフラグ

XATTR_CREATE	拡張属性が既に存在する場合、関数は失敗します。
XATTR_REPLACE	拡張属性が存在しない場合、関数は失敗します。
XATTR_DONTFOLLOW	シンボリックリンクのリンク先をたどらず、リンクそのものを操作します。
XATTR_ROOT	root （信頼された） 名前空間に属性を設定します。root 権限が必要です。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 .wav ファイルに拡張属性を設定する

```
<?php
$file = 'my_favourite_song.wav';
xattr_set($file, 'Artist', 'Someone');
xattr_set($file, 'My ranking', 'Good');
xattr_set($file, 'Listen count', '34');

/* ... その他のコード ... */

printf("あなたは、この曲を %d 回再生しました", xattr_get($file, 'Listen count'));
?>
```

参考

- *xattr_get()*
- *xattr_remove()*

---

xattr_supported

(PECL xattr:1.0)

xattr_supported — ファイルシステムが拡張属性をサポートしているかどうかを調べる

説明

*bool **xattr_supported** (string $filename [, int $flags])*

この関数は、指定したファイルを保持するファイルシステムが 拡張属性をサポートしているかどうかを調べます。ファイルに対する 読み込みアクセス権限を必要とします。

パラメータ

filename

調べるファイルのパス。

flags

サポートされる xattr のフラグ

XATTR_DONTFOLLOW	シンボリックリンクのリンク先をたどらず、リンクそのものを操作します。

返り値

*この関数は、ファイルシステムが拡張属性をサポートしている場合に **TRUE**、 していない場合に **FALSE**、そして判断できない場合（たとえばパスが間違っていたりファイルへのアクセス権限がない場合）には **NULL** を返します。*

例

Example#1 xattr_supported() の例

以下のコードは、拡張属性が使用可能かどうかを調べます。

```
<?php
$file = 'some_file';

if (xattr_supported($file)) {
    /* ... xattr_* 関数が使用できます ... */
}

?>
```

参考

- xattr_get()
- xattr_list()

目次

---

xdiff 関数

導入

xdiff 拡張モジュールは、テキストファイルおよびバイナリファイルに対しての パッチの作成および適用を行います。

要件

xdiff を使用するには libxdiff がインストールされている必要があります。 これは libxdiff のホームページ » http://www.xmailserver.org/xdiff-lib.html で取得可能です。

> **注意**: memory_limit を考慮してこれらの関数を使用するには、すくなくとも libxdiff 0.7 が必要です。

インストール手順

xdiff は、現在は PECL » http://pecl.php.net/package/xdiff から取得可能です。

*nix-like システムを使用しており » PEAR が使用可能な場合は、pear インストーラを使用して xdiff 拡張モジュールを インストールすることが可能です。そのためには以下のコマンドを使用します。 **pear -v install xdiff**

tar.gz パッケージをダウンロードし、xdiff を手動でインストールすることも 常に可能です。

Example#1 xdiff の手動インストール

```
gunzip  xdiff-xxx.tgz
tar  -xvf  xdiff-xxx.tar
cd  xdiff-xxx
phpize
./configure  &&  make  &&  make  install
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

XDIFF_PATCH_NORMAL (integer)
XDIFF_PATCH_REVERSE (integer)

---

xdiff_file_diff_binary

(PECL xdiff:0.2-1.4)

xdiff_file_diff_binary — 2 つのファイルのバイナリ diff を作成する

説明

bool **xdiff_file_diff_binary** (string $file1 , string $file2 , string $dest)

ふたつのファイルのバイナリ diff を作成し、それをファイルに保存します。 この関数はテキストファイルとバイナリファイルの両方に適用可能です。

パラメータ

file1

file2

dest

結果のファイルへのパス。このファイルはバイナリ形式となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 xdiff_file_diff_binary() の例

以下のコードは 2 つのアーカイブのバイナリ diff を作成します。

```
<?php
$old_version  = 'my_script_1.0.tgz';
$new_version  = 'my_script_1.1.tgz';

xdiff_file_diff_binary($old_version, $new_version, 'my_script.bdiff');
?>
```

注意

注意: 両方のファイルがメモリに読み込まれるので、 memory_limit が十分大きな値に設定されていることを確認しましょう。

参考

- xdiff_string_diff_binary()

---

xdiff_file_diff

(PECL xdiff:0.2-1.4)

xdiff_file_diff — 2 つのファイルの unified diff を作成する

説明

bool **xdiff_file_diff** (string $file1 , string $file2 , string $dest [, int $context [, bool $minimal]])

ふたつのファイルの diff を作成し、結果をファイルに保存します。

パラメータ

file1

file2

dest

結果のファイルへのパス。

context

diff の結果の前後に含める行の数を指定します。

minimal

このパラメータを **TRUE** にすると、diff のファイルが最小になります（非常に時間がかかります）。結果のファイルは可読形式となります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 xdiff_file_diff() の例

以下のコードは 2 つの php ファイルの unified diff を作成します。

```
<?php
$old_version  = 'my_script.php';
$new_version  = 'my_new_script.php';

xdiff_file_diff($old_version, $new_version, 'my_script.diff', 2);
?>
```

注意

注意: この関数はバイナリファイルに対しては動作しません。バイナリファイルの 差分を作成するには xdiff_file_diff_binary() を使用してください。

参考

- xdiff_string_diff()

---

xdiff_file_merge3

(PECL xdiff:0.2-1.4)

xdiff_file_merge3 — 3 つのファイルをひとつに統合する

説明

mixed **xdiff_file_merge3** (string $file1 , string $file2 , string $file3 , string $dest)

三つのファイルをひとつに統合し、結果をファイルに保存します。

パラメータ

file1

file2

file3

dest

結果のファイルへのパス。

返り値

成功した場合に **TRUE** 、もし拒否された部分がある場合にはその文字列、 内部エラーが発生した場合に **FALSE** を返します。

例

Example#1 xdiff_file_merge3() の例

以下のコードは、3 つのファイルをひとつに統合します。

```
<?php
$old_version = 'original_script.php';
$fix1 = 'script_with_fix1.php';
$fix2 = 'script_with_fix2.php';

$errors = xdiff_file_merge3($old_version, $fix1, $fix2, 'fixed_script.php');
if (is_string($errors)) {
    echo "Rejects:¥n";
    echo $errors;
}
?>
```

参考

- xdiff_string_merge3()

---

xdiff_file_patch_binary

(PECL xdiff:0.2-1.4)

xdiff_file_patch_binary — ファイルにバイナリ diff 形式のパッチを適用する

説明

bool **xdiff_file_patch_binary** (string $file , string $patch , string $dest)

ファイル file にバイナリ形式のパッチファイル patch を適用し、結果をファイル dest に保存します。

パラメータ

file

元のファイル。

patch

バイナリのパッチファイル。

dest

結果のファイルへのパス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 xdiff_file_patch_binary() の例

以下のコードはバイナリ diff 形式のパッチをファイルに適用します。

```
<?php
$old_version = 'archive-1.0.tgz';
$patch = 'archive.bpatch';

$result = xdiff_file_patch_binary($old_version, $patch, 'archive-1.1.tgz');
if ($result) {
   echo "パッチが適用されました";
} else {
    echo "パッチが適用できませんでした";
}

?>
```

注意

注意: 両方のファイル (file および patch) がメモリに読み込まれるので、 memory_limit が十分大きな値に設定されていることを確認しましょう。

参考

- xdiff_string_patch_binary()

---

xdiff_file_patch

(PECL xdiff:0.2-1.4)

xdiff_file_patch — ファイルに unified diff 形式のパッチを適用する

説明

mixed **xdiff_file_patch** (string $file , string $patch , string $dest [, int $flags])

ファイル file に unified 形式のパッチファイル patch を適用し、結果をファイルに保存します。

パラメータ

file

元のファイル。

patch

パッチファイル。

dest

結果のファイルへのパス。

flags

XDIFF_PATCH_NORMAL（デフォルト。通常のパッチ）あるいは **XDIFF_PATCH_REVERSE**（逆パッチ） のいずれか。

返り値

内部エラーが発生した場合に **FALSE** を返します。 パッチの適用に成功した場合には、パッチが拒否された部分の文字列 あるいは **TRUE** を返します。

例

Example#1 xdiff_file_patch() の例

以下のコードは、unified diff 形式のパッチをファイルに適用します。

```
<?php
$old_version = 'my_script-1.0.php';
$patch = 'my_script.patch';

$errors = xdiff_file_patch($old_version, $patch, 'my_script-1.1.php');
if (is_string($errors)) {
    echo "Rejects:¥n";
    echo $errors;
}

?>
```

Example#2 逆パッチの例

以下のコードは、パッチを逆方向に適用します。

```
<?php
$new_version = 'my_script-1.1.php';
$patch = 'my_script.patch';

$errors = xdiff_file_patch($new_version, $patch, 'my_script-1.0.php', XDIFF_PATCH_REVERSE);
if (is_string($errors)) {
    echo "Rejects:¥n";
    echo $errors;
}

?>
```

参考

- xdiff_string_patch()

---

xdiff_string_diff_binary

(PECL xdiff:0.2-1.4)

xdiff_string_diff_binary — 2 つの文字列のバイナリ diff を作成する

説明

string ***xdiff_string_diff_binary*** *(string $str1 , string $str2)*

ふたつの文字列のバイナリ diff を作成します。

パラメータ

str1

str2

返り値

成功した場合に結果の文字列、内部エラーが発生した場合に ***FALSE*** *を返します。*

参考

- *xdiff_file_diff_binary()*

---

xdiff_string_diff

(PECL xdiff:0.2-1.4)

xdiff_string_diff — 2 つの文字列の unified diff を作成する

説明

string ***xdiff_string_diff*** *(string $str1 , string $str2 [, int $context [, bool $minimal]])*

ふたつの文字列の diff を作成します。

パラメータ

str1

str2

context

diff の結果の前後に含める行の数を指定します。

minimal

このパラメータを ***TRUE*** *にすると、diff のファイルが最小になります（非常に時間がかかります）。結果のファイルは可読形式となります。*

返り値

成功した場合に結果の文字列、内部エラーが発生した場合に ***FALSE*** *を返します。*

例

Example#1 xdiff_string_diff() の例

以下のコードは 2 つの記事の unified diff を作成します。

```
<?php
$old_article = file_get_contents('./old_article.txt');
$new_article = $_REQUEST['article']; /* Let's say that someone pasted a new article to html form */

$diff = xdiff_string_diff($old_article, $new_article, 1);
if (is_string($diff)) {
    echo "2 つの記事の差分:\n";
    echo $diff;
}

?>
```

注意

> ***注意***: *この関数はバイナリ文字列に対しては動作しません。バイナリ文字列の 差分を作成するには xdiff_string_diff_binary() を使用してください。*

参考

- *xdiff_file_diff()*

---

xdiff_string_merge3

(PECL xdiff:0.2-1.4)

xdiff_string_merge3 — 3 つの文字列をひとつに統合する

説明

mixed ***xdiff_string_merge3*** *(string $str1 , string $str2 , string $str3 [, string &$error])*

三つの文字列をひとつに結合します。

パラメータ

str1

str2

str3

error

指定した場合、パッチを拒否された部分がこの変数に保存されます。

返り値

統合された文字列を返します。内部エラーが発生した場合は ***FALSE****、 統合された文字列が空の文字列の場合は* ***TRUE*** *を返します。*

参考

- *xdiff_file_merge3()*

---

xdiff_string_patch_binary

(PECL xdiff:0.2-1.4)

xdiff_string_patch_binary — 文字列にバイナリ diff 形式のパッチを適用する

説明

string ***xdiff_string_patch_binary*** *(string $str , string $patch)*

文字列に、バイナリ形式のパッチ patch を適用します。

パラメータ

str

元のバイナリ文字列。

patch

バイナリパッチ文字列。

返り値

パッチ適用後の文字列、あるいはエラー時に ***FALSE*** *を返します。*

参考

- *xdiff_file_patch_binary()*

---

xdiff_string_patch

(PECL xdiff:0.2-1.4)

xdiff_string_patch — 文字列に unified diff 形式のパッチを適用する

説明

string ***xdiff_string_patch*** *(string $str , string $patch [, int $flags [, string &$error]])*

ある文字列に、unified 形式のパッチ文字列 patch を適用します。

パラメータ

str

元の文字列。

patch

unified 形式のパッチ文字列。

flags

flags は ***XDIFF_PATCH_NORMAL*** *（デフォルト。通常のパッチ） あるいは* ***XDIFF_PATCH_REVERSE*** *（逆パッチ） のいずれかです。*

error

これを指定すると、パッチを拒否された部分がここに保存されます。

返り値

パッチ適用後の文字列、あるいはエラー時に **FALSE** を返します。

例

Example#1 xdiff_string_patch() の例

以下のコードは、ある記事に対して変更を適用します。

```
<?php
$old_article = file_get_contents('./old_article.txt');
$diff = $_SERVER['patch']; /* だれかが html フォームからパッチを投稿したとしましょう */

$errors = '';

$new_article = xdiff_string_patch($old_article, $diff, XDIFF_PATCH_NORMAL, $errors);
if (is_string($new_article)) {
    echo "新しい記事:\n";
    echo $new_article;
}

if (strlen($errors)) {
    echo "Rejects: ¥n";
    echo $errors;
}

?>
```

参考

- xdiff_file_patch()

目次

XML パーサ関数

導入

XML (eXtensible Markup Language) は、Web における構造化された ドキュメント交換用のデータフォーマットです。XML は、World Wide Web consortium (W3C) で規定された規格です。XML に関する情報およ び関連する技術は、» http://www.w3.org/XML/ で参照することができます。

このPHPエクステンションは、James Clark氏の expatのサポートをPHPに付加します。 このツールキットは、XML ドキュメントの構文解析をしますが、 検証は行いません。3種類のソース 文字エンコーディング、 US-ASCII, ISO-8859-1 ,UTF-8 がPHPでサポートされます。UTF-16 はサポートさ れません。

この拡張モジュールは、XML パーサの作成 を行い、異なった XML イベントに関してハンドラ を定義します。各XMLパーサーには、設定可能な小数のパラメータ もあります。

要件

このエクステンションは、expat を使用します。 これは、» http://www.jclark.com/xml/expat.htmlにあります。 expatに付属のMakefile は、デフォルトでライブラリを構築しません。こ れを行うmakeルールを次のように指定できます。

```
libexpat.a: $(OBJS)
     ar -rc $@ $(OBJS)
     ranlib $@
```

expat のソース RPM パッケージが » http://sourceforge.net/projects/expat/ にあります。

インストール手順

付属しているexpatライブラリを用いて以下の関数はデフォルトで有効となっ ています。 --disable-xmlを指定してXMLサポート を無効にすることができます。Apache 1.3.9以降でモジュールとしてPHPを コンパイルする場合、PHPは、Apacheから自動的に付属する expatライブラリを使用します。 付属するexpatライブラリを使用したくない場合には、 --with-expat-dir=DIRを指定してPHP のconfigureを実行してください。ただし、DIRは、expatをインストールした ベースディレクトリです。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

実行時設定

設定ディレクティブは定義されていません。

リソース型

xml

xml_parser_create()および xml_parser_create_ns() により返された xmlリソースは、 このエクステンションにより提供された関数で使用される XMLパーサのインスタンスを参照します。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

XML_ERROR_NONE (integer)
XML_ERROR_NO_MEMORY (integer)
XML_ERROR_SYNTAX (integer)
XML_ERROR_NO_ELEMENTS (integer)
XML_ERROR_INVALID_TOKEN (integer)
XML_ERROR_UNCLOSED_TOKEN (integer)
XML_ERROR_PARTIAL_CHAR (integer)
XML_ERROR_TAG_MISMATCH (integer)
XML_ERROR_DUPLICATE_ATTRIBUTE (integer)
XML_ERROR_JUNK_AFTER_DOC_ELEMENT (integer)
XML_ERROR_PARAM_ENTITY_REF (integer)
XML_ERROR_UNDEFINED_ENTITY (integer)
XML_ERROR_RECURSIVE_ENTITY_REF (integer)
XML_ERROR_ASYNC_ENTITY (integer)
XML_ERROR_BAD_CHAR_REF (integer)
XML_ERROR_BINARY_ENTITY_REF (integer)
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF (integer)
XML_ERROR_MISPLACED_XML_PI (integer)
XML_ERROR_UNKNOWN_ENCODING (integer)
XML_ERROR_INCORRECT_ENCODING (integer)
XML_ERROR_UNCLOSED_CDATA_SECTION (integer)
XML_ERROR_EXTERNAL_ENTITY_HANDLING (integer)
XML_OPTION_CASE_FOLDING (integer)
XML_OPTION_TARGET_ENCODING (integer)
XML_OPTION_SKIP_TAGSTART (integer)
XML_OPTION_SKIP_WHITE (integer)

イベントハンドラ

XML イベントハンドラは次のように定義されます。

サポートされる XML ハンドラ

ハンドラ設定用の PHP 関数	イベントの説明
xml_set_element_handler()	要素イベントは、XML パーサーが開始または終了タグに出会うたび に発行されます。開始タグと終了タグについて別のハンドラがあり ます。
xml_set_character_data_handler()	文字データは、タグの間の空白を含めて XML ドキュメントにおけ るほぼ全ての非マークアップ部分の内容です。XML パーサーは、 空白を加えたり削除したりしないことに注意してください。空白が意味を有するかどうかを決めるのは、アプリケーション側の責任 です。
xml_set_processing_instruction_handler()	PHP プログラマは、既に処理用命令 (PI) に既に慣れているに違 いありません。<?php ?> は処理用命令であり、この場合、 php は "PI ターゲット"と呼ばれます。 これらの処理はアプリケーション依存ですが、全ての PI ターゲッ トが "XML" から始まることだけは、規定されています。
xml_set_default_handler()	別のハンドラでしないことをデフォルトのハンドラで行います。 XML およびドキュメント型の宣言のようなことをデフォルトハンドラで 行います。
xml_set_unparsed_entity_decl_handler()	このハンドラは、処理されない (NDATA) エンティティの宣言用に コールされます。
xml_set_notation_decl_handler()	このハンドラは、表記の宣言用にコールされます。
xml_set_external_entity_ref_handler()	このハンドラは、XML パーサーが外部処理された通常のエンティティ への参照を見つけた際にコールされます。これは、例えば、ファ イルまたは URL への参照とすることが可能です。例としては、 外部エンティティ の例 を参照ください。

大文字変換(Case Folding)

要素ハンドラ関数は、その要素に大文字小文字を変換する (case-folded)の名前をつけることができます。 大文字変換(case-folding) は、XML標準により "大文字でないものは等 価な大文字に置換される一連の文字に適用されるプロセス" として定義 されています。言い替えると、XML に関しては単に大文字変換は大文字 にすることを意味します。

デフォルトで、ハンドラ関数に渡される全ての要素名は、大文字変換さ れます。この動作は、xml_parser_get_option() および xml_parser_set_option() 関数でXMLパーサー 毎にそれぞれ問い合わせ、制御することが可能です。

エラーコード

(xml_parse() により返されるものとして) XMLエラーコードとして次のような定数が定義されています。:

- XML_ERROR_NONE
- XML_ERROR_NO_MEMORY
- XML_ERROR_SYNTAX

- XML_ERROR_NO_ELEMENTS
- XML_ERROR_INVALID_TOKEN
- XML_ERROR_UNCLOSED_TOKEN
- XML_ERROR_PARTIAL_CHAR
- XML_ERROR_TAG_MISMATCH
- XML_ERROR_DUPLICATE_ATTRIBUTE
- XML_ERROR_JUNK_AFTER_DOC_ELEMENT
- XML_ERROR_PARAM_ENTITY_REF
- XML_ERROR_UNDEFINED_ENTITY
- XML_ERROR_RECURSIVE_ENTITY_REF
- XML_ERROR_ASYNC_ENTITY
- XML_ERROR_BAD_CHAR_REF
- XML_ERROR_BINARY_ENTITY_REF
- XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
- XML_ERROR_MISPLACED_XML_PI
- XML_ERROR_UNKNOWN_ENCODING
- XML_ERROR_INCORRECT_ENCODING
- XML_ERROR_UNCLOSED_CDATA_SECTION
- XML_ERROR_EXTERNAL_ENTITY_HANDLING

文字エンコーディング

PHPのXML拡張機能は、異なった文字エンコーディング を通じて» Unicode 文字セットをサポートします。ソースエンコーディング およびターゲットエンコーディング という2種類の文字エンコーディングがあります。 PHP におけるドキュメントの内部表現は、常に UTF-8でエンコードされます。

ソースエンコーディングは、XMLドキュメントが 構文解析された際に行わ れます。XML パーサの 作成を行う際に、ソースエンコードを指定することができます。 (このエンコーディングは、その XML パーサーが存在する間、後で変更す ることはできません)サポートされるソースエンコーディングは、 ISO-8859-1, US-ASCII , UTF-8 です。前の二つは、シングルバイトエンコー ディングです。これは、各文字がシングルバイトで表現されることを意 味します。UTF-8 は、1から4バイトの可変ビット 数(最大21ビット)で構成された文字をエンコードすることが可能です。 PHP で用いられるデフォルトのソースエンコーディングは、 ISO-8859-1です。

ターゲットエンコーディングは、PHPがデータをXMLハンドラ関数に 渡す時に行われます。あるXMLパーサが作成された際、ターゲットエン コーディングは、ソースエンコーディングと同様に設定されます。 しかし、これは、いつでも変更可能です。ターゲットエンコーディング は、タグ名と同様に文字データに作用し、命令を処理します。

XML パーサがソースエンコーディングが表現できる範囲の外側の文字に 出会った場合、エラーが返されます。

解釈するXMLドキュメントにおいてPHPが文字に出会った際に、選択した ターゲットエンコーディングで表現できない文字に出会った場合、問題 の文字は "降格" されます。現在、このことはこのような文字が疑問符 で置換されることを意味します。

例

以下にXMLドキュメントを処理するPHPスクリプトの例をいくつか示します。

XML エレメント構造の例

この最初の例は、あるドキュメント中のstart エレメントの構造をイン デントを付けて表示します。

Example#1 XML エレメント構造を表示

```
<?php
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs)
{
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name¥n";
    $depth[$parser]++;
}

function endElement($parser, $name)
{
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("XML 入力をオープンできませんでした");
}
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML エラー: %s が %d 行目で発生しました",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
?>
```

XMLタグのマッピングの例

Example#2 XMLをHTMLにマップする

この例は、XMLドキュメントのタグを直接HTMLタグにマップします。 "map array" にないエレメントは無視されます。もちろん、この例は、 特定のXML ドキュメント型を有する場合のみ動作します。

```
<?php
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs)
{
    global $map_array;
    if (isset($map_array[$name])) {
        echo "<$map_array[$name]>";
    }
}

function endElement($parser, $name)
{
    global $map_array;
    if (isset($map_array[$name])) {
        echo "</$map_array[$name]>";
    }
}

function characterData($parser, $data)
{
    echo $data;
}

$xml_parser = xml_parser_create();
// case-folding を用いることで、$map_array から確実にタグを見つけられるようにします
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("XML 入力をオープンできませんでした");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML エラー: %s が %d 行目で発生しました",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
?>
```

XML 外部エンティティの例

この例は、XML コードに焦点を当てます。この例は、他のドキュメント をインクルードし処理するための外部エンティティリファレンスのハン ドラの使用法およびPIの処理方法、PIが含むコードに関する"信頼度" を定義する手段を説明します。

この例で使用される XML ドキュメントは、例題ファイル (xmltest.xml および xmltest2.xml) にあります。

Example#3 外部エンティティの例

```
<?php
$file = "xmltest.xml";

function trustedFile($file)
{
    // 信頼できるのは、自分自身が所有しているローカルファイルのみです
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs)
{
    echo "&lt;<font  color=¥"#0000cc¥">$name</font>";
    if (count($attribs)) {
        foreach ($attribs as $k => $v) {
            echo "  <font  color=¥"#009900¥">$k</font>=¥"<font
                      color=¥"#990000¥">$v</font>¥"";
        }
    }
    echo "&gt;";
}

function endElement($parser, $name)
{
    echo "&lt;/<font  color=¥"#0000cc¥">$name</font>&gt;";
}

function characterData($parser, $data)
{
    echo "<b>$data</b>";
}
```

```
function PIHandler($parser, $target, $data)
{
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
           // もし「信頼できる」ドキュメントだった場合、その中に書かれている
            // PHP コードを実行しても安全だと考えます。そうでない場合、
            // コードを実行するかわりにコードそのものを表示します。
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("信頼できない PHP コード: <i>%s</i>",
                       htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data)
{
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font  color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font  size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("エンティティ %s (%s にある) をオープンできませんでした\n", $openEntityNames,
                    $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML エラー: %s が、%d 行目でエンティティ %s のパース中に発生しました\n",
                        xml_error_string(xml_get_error_code($parser)),
                        xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}

function new_xml_parser($file)
{
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("XML 入力をオープンできませんでした");
}

echo "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML エラー: %s が %d 行目で発生しました\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
echo "</pre>";
echo "パースが完了しました\n";
xml_parser_free($xml_parser);

?>
```

Example#4 xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
  <TITLE>Title &plainEntity;</TITLE>
```

```
 <para>
   <informaltable>
    <tgroup cols="3">
     <tbody>
         <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
        <row><entry>a2</entry><entry>c2</entry></row>
         <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
     </tbody>
    </tgroup>
   </informaltable>
 </para>
 &systemEntity;
 <section id="about">
   <title>About this Document</title>
  <para>
    <!-- this is a comment -->
    <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </section>
</chapter>
```

このファイルは、xmltest.xml からインクルードされます。

Example#5 xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
     <element attrib="value"/>
    &testEnt;
    <?php print "This is some more PHP code being executed."; ?>
</foo>
```

---

utf8_decode

(PHP 4, PHP 5)

utf8_decode — UTF-8 エンコードされた ISO-8859-1 文字列をシングルバイトの ISO-8859-1 に変換する

説明

string **utf8_decode** (string $data)

この関数は、data が UTF-8 エンコードされているものとみなして ISO-8859-1 に変換します。

パラメータ

data

UTF-8 エンコードされた文字列。

返り値

data を ISO-8859-1 に変換した結果を返します。

参考

- UTF-8 エンコーディングについての説明は utf8_encode() を参照ください。

---

utf8_encode

(PHP 4, PHP 5)

utf8_encode — ISO-8859-1 文字列を UTF-8 にエンコードする

説明

string **utf8_encode** (string $data)

この関数は、文字列 data を UTF-8 にエンコードし、エンコードされた文字列を 返します。UTF-8 は、 wide character の値をバイトストリームに エンコードするために Unicode で使用される 標準的な仕組みです。UTF-8 は、プレーンな ASCII文字を通し、自己同期(これは、バイトストリームの 開始地点をプログラムが見積もることが可能であることを意味します) します。また、ソートのような標準的な文字列比較関数を使用可能です。 PHP は、UTF-8 文字を次のように最大 4 バイトで エンコードします。

UTF-8 エンコーディング

バイト	ビット	表現形式
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

各 b は、文字列データを保存可能なビットを示します。

パラメータ

data

ISO-8859-1 形式の文字列。

返り値

data を UTF-8 に変換した結果を返します。

---

xml_error_string

(PHP 4, PHP 5)

xml_error_string — XML パーサのエラー文字列を得る

説明

string ***xml_error_string*** *(int $code)*

エラーコード code の説明を有する文字列を 返します。

パラメータ

code

xml_get_error_code() からのエラーコード。

返り値

エラーコード code の説明を有する文字列を 返します。説明がない場合には ***FALSE*** *を返します。*

参考

- *xml_get_error_code()*

---

xml_get_current_byte_index

(PHP 4, PHP 5)

xml_get_current_byte_index — XML パーサのカレントのバイトインデックスを得る

説明

int ***xml_get_current_byte_index*** *(resource $parser)*

指定した XML パーサのカレントのバイトインデックスを取得します。

パラメータ

parser

バイトインデックスを得る XML パーサへのリファレンス。

返り値

この関数は、parser が有効なパーサを参照しない場合に ***FALSE****、そうでない場合に データバッファにおいてパーサが現在処理中のバイトインデックス（先頭が 0）を返します。*

注意

警告

この関数は、UTF-8 エンコードされたテキストに基づいたバイトインデックスを返します。 入力が他のエンコーディングであっても無視します。

参考

- ***xml_get_current_column_index()***
- ***xml_get_current_line_index()***

---

xml_get_current_column_number

(PHP 4, PHP 5)

xml_get_current_column_number — XML パーサのカレントのカラム番号を取得する

説明

int ***xml_get_current_column_number*** *(resource $parser)*

指定した XML パーサのカレントのカラム番号を取得します。

パラメータ

parser

カラム番号を得る XML パーサへのリファレンス。

返り値

この関数は、parser が有効なパーサでない場合に **FALSE** を返します。それ以外の場合は、現在のパーサの (xml_get_current_line_number() で取得した) 現在の行のカラムを返します。

参考

- xml_get_current_byte_index()
- **xml_get_current_line_index()**

---

xml_get_current_line_number

(PHP 4, PHP 5)

xml_get_current_line_number — XML パーサのカレントの行番号を得る

説明

int **xml_get_current_line_number** (resource $parser)

指定した XML パーサのカレントの行番号を取得します。

パラメータ

parser

行番号を得る XML パーサへのリファレンス。

返り値

この関数は、parser が有効なパーサでない場合に **FALSE**、それ以外の場合にデータバッファの中で 現在パーサが処理中の行番号を返します。

参考

- **xml_get_current_column_index()**
- xml_get_current_byte_index()

---

xml_get_error_code

(PHP 4, PHP 5)

xml_get_error_code — XML パーサのエラーコードを得る

説明

int **xml_get_error_code** (resource $parser)

XML パーサのエラーコードを取得します。

パラメータ

parser

エラーコードを得る XML パーサへのリファレンス。

返り値

この関数は、parser が有効なパーサを参照していない 場合に **FALSE** を返します。その他の場合、 エラーコードセクション に 一覧が示されたエラーコードを返します。

参考

- xml_error_string()

---

xml_parse_into_struct

(PHP 4, PHP 5)

xml_parse_into_struct — 配列構造体に XML データを処理する

説明

```
int xml_parse_into_struct ( resource $parser , string $data , array &$values [, array &$index ] )
```

この関数は、XMLファイルを処理し、2つの配列構造体に代入します。 ひとつめの配列 (index) は、配列 values にある適当な値の位置を指すポインタを保持しています。 これら最後の二つのパラメータは参照渡しとする必要があります。

パラメータ

parser

data

values

index

返り値

xml_parse_into_struct() は失敗した場合に 0、 成功した場合に 1 を返します。これは **FALSE** および **TRUE** とは 異なるものですので、=== のような演算子を使用する場合は注意しましょう。

例

以下の例は、この関数により生成された配列の内部構造を示すものです。 noteタグをparaタグの中に埋 め込んで使用した後、これをパースし、生成された構造体を出力します。

Example#1 xml_parse_into_struct() の例

```
<?php
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p, $simple, $vals, $index);
xml_parser_free($p);
echo "Index array¥n";
print_r($index);
echo "¥nVals array¥n";
print_r($vals);
?>
```

このコードを実行した場合、出力は次のようになります。

```
Index array
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )

)

Vals array
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )

    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )

)
```

(expatライブラリを使用した)イベント駆動型パーサによる処理は、XML ドキュメントが複雑な場合に複雑になる場合があります。この関数は、 DOM 形式のオブジェクトを生成しませんが、ツリー風に一連の処理を行い 得る構造体を生成します。つまり、XMLのファイルを表すオブジェクトを 容易に作成することが可能です。次のXMLファイルを見てみましょう。 このファイルでは、アミノ酸の情報に関する小さなデータベースを表します。

Example#2 moldb.xml - 分子情報の小さなデータベース

```
<?xml version="1.0"?>
<moldb>

   <molecule>
```

```
        <name>Alanine</name>
        <symbol>ala</symbol>
       <code>A</code>
        <type>hydrophobic</type>
   </molecule>

   <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
       <code>K</code>
        <type>charged</type>
   </molecule>

</moldb>
```

ドキュメントを処理し、適当なオブジェクトを生成するいくつかのコード

Example#3 parsemoldb.php - moldb.xml を処理し、分子オブジェクトの配列に代入

```
<?php

class AminoAcid {
     var $name;  // aa name
     var $symbol;    // three letter symbol
     var $code;  // one letter code
     var $type;  // hydrophobic, charged or neutral

     function AminoAcid ($aa)
     {
          foreach ($aa as $k=>$v)
              $this->$k = $aa[$k];
     }
}

function readDatabase($filename)
{
     // read the XML database of aminoacids
     $data = implode("", file($filename));
     $parser = xml_parser_create();
     xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING, 0);
     xml_parser_set_option($parser, XML_OPTION_SKIP_WHITE, 1);
     xml_parse_into_struct($parser, $data, $values, $tags);
     xml_parser_free($parser);

     // loop through the structures
     foreach ($tags as $key=>$val) {
          if ($key == "molecule") {
              $molranges = $val;
              // each contiguous pair of array entries are the
                 // lower and upper range for each molecule definition
              for ($i=0; $i < count($molranges); $i+=2) {
                  $offset = $molranges[$i] + 1;
                  $len = $molranges[$i + 1] - $offset;
                  $tdb[] = parseMol(array_slice($values, $offset, $len));
              }
          } else {
               continue;
          }
     }
     return $tdb;
}

function parseMol($mvalues)
{
     for ($i=0; $i < count($mvalues); $i++) {
         $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
     }
     return new AminoAcid($mol);
}

$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:¥n";
print_r($db);

?>
```

parsemoldb.phpを実行した後、変数 $db は、オブジェクト AminoAcidの配列を有しており、スクリプトの 出力は、次のようになります。

```
** Database of AminoAcid objects:
Array
(
      [0] => aminoacid Object
          (
               [name] => Alanine
               [symbol] => ala
              [code] => A
               [type] => hydrophobic
          )

      [1] => aminoacid Object
          (
               [name] => Lysine
               [symbol] => lys
              [code] => K
               [type] => charged
          )

)
```

---

xml_parse

(PHP 4, PHP 5)

xml_parse — XML ドキュメントの処理を開始する

説明

int **xml_parse** (resource $parser , string $data [, bool $is_final])

xml_parse() は XML ドキュメントを処理します。 設定されているイベントのハンドラが、必要に応じてコールされます。

パラメータ

parser

使用する XML パーサへのリファレンス。

data

処理するデータ。ドキュメントは、 is_final パラメータが設定され、 最後のデータが処理され **TRUE** になるまで、新規のデータに関して 複数回 **xml_parse()** をコールすることにより、 部分毎で処理することが可能です。

is_final

TRUE が設定された場合、data は この処理の間に送られた最後のデータということになります。

返り値

成功した場合に 1、失敗した場合に 0 を返します。

処理が成功しなかった場合、エラー情報を xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number() および xml_get_current_byte_index() により取得可能です。

> **注意**: エンティティのエラーが報告されるのは、ドキュメントの最後 つまり is_final が **TRUE** に設定されている場合です。

---

xml_parser_create_ns

(PHP 4 >= 4.0.5, PHP 5)

xml_parser_create_ns — 名前空間をサポートした XML パーサを生成する

説明

resource **xml_parser_create_ns** ([string $encoding [, string $separator]])

xml_parser_create_ns() は XML 名前空間をサポートした 新しい XML パーサを作成し、他の XML 関数が使用するハンドルを返します。

パラメータ

encoding

PHP 4 では、オプションの encoding で入出力の エンコーディングを指定します。PHP 5 以降では入力のエンコーディングは 自動判定されるので、encoding パラメータは 出力のエンコーディングのみを指定することになります。PHP 4 では、 デフォルトの出力エンコーディングは入力の文字セットと同じです。 もし空の文字列が渡された場合、先頭の 3 あるいは 4 バイトの内容をもとに パーサがエンコーディングの判別を試みます。PHP 5.0.0 および 5.0.1 ではデフォルトの出力文字セットは ISO-8859-1 で、PHP 5.0.2 以降では UTF-8 です。サポートされるエンコーディングは ISO-8859-1、UTF-8 および US-ASCII です。

separator

名前空間を含めたタグパラメータをハンドラ関数に渡す際には、 名前空間名とタグ名を文字列 seperator でつなげたものが使用されます。seperator のデフォルトは ':' です。

返り値

新しい XML パーサのリソースハンドルを返します。

参考

- xml_parser_create()
- xml_parser_free()

---

xml_parser_create

(PHP 4, PHP 5)

xml_parser_create — XML パーサを作成する

説明

resource **xml_parser_create** ([string $encoding])

xml_parser_create() は新しい XML パーサを作成し、 他の XML 関数が使用するハンドルを返します。

パラメータ

encoding

PHP 4 では、オプションの encoding で入出力の エンコーディングを指定します。PHP 5 以降では入力のエンコーディングは 自動判定されるので、encoding パラメータは 出力のエンコーディングのみを指定することになります。PHP 4 では、 デフォルトの出力エンコーディングは入力の文字セットと同じです。 もし空の文字列が渡された場合、先頭の 3 あるいは 4 バイトの内容をもとに パーサがエンコーディングの判別を試みます。PHP 5.0.0 および 5.0.1 ではデフォルトの出力文字セットは ISO-8859-1 で、PHP 5.0.2 以降では UTF-8 です。サポートされるエンコーディングは ISO-8859-1、UTF-8 および US-ASCII です。

返り値

新しい XML パーサのリソースハンドルを返します。

参考

- xml_parser_create_ns()
- xml_parser_free()

---

xml_parser_free

(PHP 4, PHP 5)

xml_parser_free — XML パーサを解放する

説明

bool **xml_parser_free** (resource $parser)

指定した XML パーサ parser を解放します。

パラメータ

parser
解放したい XML パーサへのリファレンス。

返り値

この関数は、parser が有効なパーサを 参照しない場合に **FALSE**、それ以外の場合にパーサを解放し、**TRUE** を返します。

---

xml_parser_get_option

(PHP 4, PHP 5)

xml_parser_get_option — XML パーサからオプションを得る

説明

mixed **xml_parser_get_option** (resource $parser , int $option)

XML パーサからオプションの値を取得します。

パラメータ

parser
オプションを取得する XML パーサへのリファレンス。
option
取得するオプション。**XML_OPTION_CASE_FOLDING** あるいは **XML_OPTION_TARGET_ENCODING** が使用可能です。 詳細は xml_parser_set_option() を参照ください。

返り値

この関数は、parser が有効なパーサを参照しないか、 option が不正な形式の場合に **FALSE** を返します (同時に **E_WARNING** を生成します)。 それ以外の場合、そのオプションの値が返されます。

---

xml_parser_set_option

(PHP 4, PHP 5)

xml_parser_set_option — XML パーサのオプションを設定する

説明

bool **xml_parser_set_option** (resource $parser , int $option , mixed $value)

XML パーサのオプションを設定します。

パラメータ

parser

オプションを設定する XML パーサへのリファレンス。

option

設定するオプション。以下を参照してください。

次のオプションが利用可能です。

XML パーサオプション

オプション定数	データ型	説明
XML_OPTION_CASE_FOLDING	integer	XMLパーサの大文字変換 を有効にするかどうかを制御する。デフォルトで有効。
XML_OPTION_SKIP_TAGSTART	integer	タグ名の最初の何文字を読み飛ばすかどうかを設定する。
XML_OPTION_SKIP_WHITE	integer	空白文字からなる値を読み飛ばすかどうかを設定する。
XML_OPTION_TARGET_ENCODING	string	XML パーサについてどの ターゲット エンコーディング を使用するかを設定する。デフォルトでは、xml_parser_create() により使用されたソース エンコーディングと同じエンコーディングが設定されます。 サポートされるターゲットエンコーディングは、 *ISO-8859-1*, *US-ASCII*, *UTF-8* です。

value

そのオプションの新しい設定値。

返り値

*この関数は、parser が有効なパーサを参照しないか、 オプションが設定出来なかった場合に **FALSE** を返します。 それ以外の場合、そのオプションが設定され、**TRUE** が返されます。*

---

xml_set_character_data_handler

(PHP 4, PHP 5)

xml_set_character_data_handler — 文字データハンドラを設定する

説明

*bool **xml_set_character_data_handler** (resource $parser , callback $handler)*

XML パーサ parser の文字データ用ハンドラ関数を設定します。

パラメータ

parser

handler

handler は、parser に関して xml_parse() がコールされた際に必要な関数の名前を有する文字列です。

handler という名前の関数は、次の 2 つのパラメータをとります。

***handler** (resource $parser , string $data)*

parser
最初のパラメータである parser は、 ハンドラをコールする XML パーサへのリファレンスです。

data
2番目のパラメータである data は、 文字データを文字列として有しています。

文字データハンドラは、XML ドキュメントのすべてのテキストに対してコールされます。(たとえば非 ASCII 文字列などで) ひとつのフラグメント内で複数回コールされることもあります。

*ハンドラ関数が空の文字列または、 **FALSE** に設定されている場合、そのハンドラは無効です。*

> ***注意**: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

---

xml_set_default_handler

(PHP 4, PHP 5)

xml_set_default_handler — デフォルトのハンドラを設定する

説明

*bool **xml_set_default_handler** (resource $parser , callback $handler)*

XMLパーサ parser のデフォルトのハンドラ関数を設定します。

パラメータ

parser

handler

handler は、 parser に関して xml_parse() がコールされた際に必要な関数の名前を有する文字列です。

handler という名前の関数は、 次の 2 つのパラメータをとる必要があります。

handler (resource $parser , string $data)
parser
最初のパラメータである parser は、 ハンドラをコールした XML パーサへのリファレンスです。
data
2 番目のパラメータである data は、 文字データを有しています。これは、XML 宣言またはドキュメント型宣言、エンティティ、 他にハンドラがない別のデータとすることが可能です。

ハンドラ関数が空の文字列または、**FALSE** に設定されている場合、そのハンドラは無効です。

注意: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

xml_set_element_handler

(PHP 4, PHP 5)

xml_set_element_handler — 開始要素および終了要素のハンドラを設定する

説明

bool **xml_set_element_handler** (resource $parser , callback $start_element_handler , callback $end_element_handler)

XML パーサ parser の要素ハンドラ関数を設定します。 start_element_handler および end_element_handler は、 xml_parse() が parser を コールした際に存在している必要がある関数の名前を有する文字列です。

パラメータ

parser

start_element_handler

start_element_handler という名前の関数は、 次の 3 つのパラメータをとる必要があります。

start_element_handler (resource $parser , string $name , array $attribs)
parser
最初のパラメータである parser は、 ハンドラをコールする XML パーサへのリファレンスです。
name
2 番目のパラメータである name は、 このハンドラがコールされた要素の名前を有しています。 大文字変換 がこのパーサに関して有効な場合、要素の名前は大文字になります。
attribs
第 3 のパラメータである attribs は、その要素の (全)属性に関する連想配列です。この配列のキーは属性の名前であり、 値は属性の値です。属性の名前は、要素名と同様に 大文字変換 となります。 属性の値は、大文字変換 されません。 この属性は、each() を使用して attribs を順次アクセスすることにより、 元の順序で取得することができます。 配列の最初のキーが最初の属性であり、後も同様です。

注意: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

end_element_handler

end_element_handler という名前の関数は、 2 つのパラメータをとる必要があります。

end_element_handler (resource $parser , string $name)
parser
最初のパラメータである parser は、 ハンドラをコールする XML パーサへのリファレンスです。
name
2 番目のパラメータである name は、 このハンドラがコールされた要素の名前を有しています。このパーサにおいて 大文字変換 が有効な場合、要素名は大文字になります。

ハンドラ関数が空の文字列または **FALSE** に設定されている場合、そのハンドラは無効です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

xml_set_end_namespace_decl_handler

(PHP 4 >= 4.0.5, PHP 5)

xml_set_end_namespace_decl_handler — 名前空間終了ハンドラを設定する

説明

bool **xml_set_end_namespace_decl_handler** (resource $parser , callback $handler)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

注意: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

xml_set_external_entity_ref_handler

(PHP 4, PHP 5)

xml_set_external_entity_ref_handler — 外部エンティティリファレンスハンドラを設定する

説明

bool **xml_set_external_entity_ref_handler** (resource $parser , callback $handler)

XML パーサ parser の外部エンティティ参照用ハンドラ関数を設定します。

パラメータ

parser

handler

handler は、 parser に関して xml_parse() がコールされた際に必要な関数の名前を有する文字列です。

handler という名前の関数は 5 つのパラメータをとり、 整数値を返す必要があります。ハンドラから返された値が **FALSE** の場合（これは値が返されない場合に相当します）、XML パーサは処理を中断し、 xml_get_error_code() は **XML_ERROR_EXTERNAL_ENTITY_HANDLING** を返します。

handler (resource $parser , string $open_entity_names , string $base , string $system_id , string $public_id)

parser
: 最初のパラメータ、parser は ハンドラをコールする XML パーサへのリファレンスです。

open_entity_names
: 2 番目のパラメータ、open_entity_names は、 エンティティの処理を行うためにオープンされたエンティティの 名前のスペース区切りのリストです（参照されるエンティティの名前を含みます）。

base
: これは外部エンティティのシステム ID（systemid ）を有しています。 現在このパラメータは常に空の文字列に設定されています。

system_id
: 4 番目のパラメータ、systemId は エンティティ宣言で指定されたシステム ID です。

public_id
: 5 番目のパラメータ publicId は、エンティティ宣言で 指定されたパブリック ID または指定されない場合は空の 文字列です。パブリック ID の中の空白文字は、XML 仕様で規定された 正規化を行っています。

ハンドラ関数が空の文字列あるいは **FALSE** に設定されている場合、 そのハンドラは無効となります。

> **注意**: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

xml_set_notation_decl_handler

(PHP 4, PHP 5)

xml_set_notation_decl_handler — 表記法宣言ハンドラを設定する

説明

bool **xml_set_notation_decl_handler** (resource $parser , callback $handler)

XML パーサ parser の表記法宣言用ハンドラ関数を設定します。

表記法の宣言は、ドキュメントの DTD の一部であり、 次のようなフォーマットとなります。

```
<!NOTATION  <parameter>name</parameter>
{  <parameter>systemId</parameter>  |  <parameter>publicId</parameter>?>
```

表記法宣言の定義に関しては、 » XML 1.0 仕様の 4.7 節 を参照ください。

パラメータ

parser

handler

handler は、 parser に関して xml_parse() がコールされた際に必要な関数の名前を有する文字列です。

handler という名前の関数は、5つのパラメータをとる必要があります。

handler (resource $parser , string $notation_name , string $base , string $system_id , string $public_id)

parser
: 最初のパラメータ、parser は ハンドラをコールする XML パーサへのリファレンスです。

notation_name
: これは、前記の表記用フォーマットに示すように表記法の 名前 です。

base
: 外部エンティティのシステムID(systemId)を 取得する際の基本となります。現在、このパラメータは、常に空の 文字列に設定されています。

system_id
: 外部表記用宣言のシステム ID

public_id
: 外部表記用宣言のパブリック ID

ハンドラ関数が空の文字列または **FALSE** に設定されていた場合、そのハンドラは無効となります。

> **注意**: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

xml_set_object

(PHP 4, PHP 5)

xml_set_object — オブジェクト内部で XML パーサを使用する

説明

bool **xml_set_object** (resource $parser , object &$object)

この関数は、object の内部から parser を使用可能にします。xml_set_element_handler() 等により設定される 全てのコールバック関数は、object のメソッドであると仮定されます。

パラメータ

parser

object

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 xml_set_object() の例

```
<?php
class xml  {
    var $parser;

    function xml()
    {
        $this->parser = xml_parser_create();

        xml_set_object($this->parser, $this);
        xml_set_element_handler($this->parser, "tag_open", "tag_close");
        xml_set_character_data_handler($this->parser, "cdata");
    }

    function parse($data)
    {
        xml_parse($this->parser, $data);
    }

    function tag_open($parser, $tag, $attributes)
    {
        var_dump($parser, $tag, $attributes);
    }

    function cdata($parser, $cdata)
    {
        var_dump($parser, $cdata);
    }

    function tag_close($parser, $tag)
    {
        var_dump($parser, $tag);
    }

} // class xml ここまで

$xml_parser = new xml();
$xml_parser->parse("<A ID='hallo'>PHP</A>");
?>
```

---

xml_set_processing_instruction_handler

(PHP 4, PHP 5)

xml_set_processing_instruction_handler — 処理命令 (PI) 用ハンドラを設定する

説明

bool **xml_set_processing_instruction_handler** (resource $parser , callback $handler)

XML パーサ parser の処理命令 (PI) 用ハンドラ関数を設定します。

処理命令は、次のフォーマットを有しています。

<?

target data

?>

このようなタグに PHP コードを入れることが可能ですが、一つの制約に 注意してください。XML PI において、PI 終了タグ (?>)は引用符で括ることができません。 このため、この文字の並びを XML ドキュメント中にPIにより埋め込んだ PHP コードの中で使用することはできません。これを使用した場合、残りの PHP コードは、"真の" PI 終了タグと同じく文字データとして処理 されます。

パラメータ

parser

handler

handler は、 parser に関して xml_parse() がコールされた際に必要な関数の名前を有する文字列です。

handler という名前の関数は、 次の 3 つのパラメータを引数とする必要があります。

handler (resource $parser , string $target , string $data)
parser
　　最初のパラメータ、parser は ハンドラをコールする XML パーサへのリファレンスです。
target
　　第 2 のパラメータ、target は PI のターゲットを有しています。
data
　　第 3 のパラメータ、data は、 PI のデータを有しています。

ハンドラ関数が空の文字列または、**FALSE** に設定されている場合、そのハンドラは無効となります。

　　注意: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

xml_set_start_namespace_decl_handler

(PHP 4 >= 4.0.5, PHP 5)

xml_set_start_namespace_decl_handler — 名前空間開始ハンドラを設定する

説明

bool **xml_set_start_namespace_decl_handler** (resource $parser , callback $handler)
警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

　　注意: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

xml_set_unparsed_entity_decl_handler

(PHP 4, PHP 5)

xml_set_unparsed_entity_decl_handler — 処理されないエンティティ宣言用ハンドラを設定する

説明

bool **xml_set_unparsed_entity_decl_handler** (resource $parser , callback $handler)

XML パーサ parser の、 処理されないエンティティ宣言用ハンドラ関数を設定します。

このハンドラは、XML パーサが次のような NDATA 宣言を有する 外部エンティティ宣言を処理する際にコールされます。

```
<!ENTITY  <parameter>name</parameter>  {<parameter>publicId</parameter>  |  <parameter>systemId</parameter>}
          NDATA  <parameter>notationName</parameter>
```

外部エンティティ宣言の記述用定義に関しては、 » XML 1.0 仕様の 4.2.2 節を参照ください。

パラメータ

parser

handler

handler は、 parser に関して xml_parse() がコールされた際に必要な関数の名前を有する文字列です。

handler という名前の関数は次のような 6 つのパラメータをとる必要があります。

handler (resource $parser , string $entity_name , string $base , string $system_id , string $public_id , string $notation_name)
parser
　　最初のパラメータ、parser は ハンドラをコールする XML パーサへのリファレンスです。
entity_name
　　宣言しようとするエンティティの名前。
base
　　外部エンティティのシステム ID(systemId) を取得する際の基本となります。現在、このパラメータは、常に 空の文字列に設定されていま

す。
system_id
外部エンティティのシステム ID。
public_id
外部エンティティのパブリック ID。
notation_name
このエンティティの表記法の名前 (xml_set_notation_decl_handler() を参照ください)。

ハンドラ関数が空の文字列または FALSE に設定されていた場合、そのハンドラは無効となります。

注意: 関数名の代わりに、オブジェクトへのリファレンスを格納した配列とメソッド名を指定することもできます。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

目次

XML-RPC 関数

導入

ここに示す関数は、XML-RPCサーバおよびクライアントを書くために使用 されます。XML-RPCに関するより詳細な情報については、 » http://www.xmlrpc.com/ を参照してください。 そして、この拡張文字モジュールと関数に関するより詳細なドキュメン トについては、 » http://xmlrpc-epi.sourceforge.net/を参照して ください。

警告

この拡張モジュールは、 実験的 なものです。この拡張モジュールの動作・関数名・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。このモジュールは自己責任で使用してください。

要件

外部ライブラリを必要としません。

インストール手順

PHPのXML-RPCサポートはデフォルトでは有効となっていません。 XML-RPCサポートを有効にするには、PHPをコンパイルする際に設定オプション --with-xmlrpc[=DIR] を使用する必要があります。この拡張モジュールは、4.1.0以降 PHP に付属 しています。

実行時設定

php.ini の設定により動作が変化します。

XML-RPC設定オプション

名前	デフォルト	変更可能	変更履歴
xmlrpc_errors	"0"	PHP_INI_SYSTEM	PHP 4.1.0 から利用可能

名前	デフォルト	変更可能	変更履歴
xmlrpc_error_number	"0"	PHP_INI_ALL	PHP 4.1.0 から利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

この拡張モジュールでは XML-RPC サーバリソースを定義しています。これは xmlrpc_server_create() が返すものです。

定義済み定数

定数は定義されていません。

---

xmlrpc_decode_request

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_decode_request — XML をネイティブなPHP 型にデコードする

説明

mixed ***xmlrpc_decode_request*** *(string $xml , string &$method [, string $encoding])*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_decode

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_decode — XML をネイティブな PHP 型にデコードする

説明

mixed ***xmlrpc_decode*** *(string $xml [, string $encoding])*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

xml

XMLRPC メソッドが返す XML レスポンス。

encoding

iconv がサポートする入力エンコーディング (デフォルトは "iso-8859-1")。

返り値

XMLRPC メソッドのレスポンスをもとに作成した、 配列あるいは整数、文字列、boolean 値を返します。

例

xmlrpc_encode_request() の例を参照ください。

参考

- *xmlrpc_encode_request()*
- *xmlrpc_is_fault()*

---

xmlrpc_encode_request

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_encode_request — メソッドリクエスト用の XML を生成する

説明

string ***xmlrpc_encode_request*** *(string $method , mixed $params [, array $output_options])*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

method

コールするメソッドの名前。

params

メソッドのシグネチャに対応したパラメータ。

output_options

出力オプションを指定する配列。以下の内容が指定できます（強調してあるものがデフォルトです）。

- *output_type: php, xml*
- *verbosity: no_white_space, newlines_only, pretty*
- *escaping: cdata, non-ascii, non-print, markup （単一の値を表す文字列、あるいは複数の値の配列となります）*
- *version: simple, xmlrpc, soap 1.1, auto*
- *encoding: iso-8859-1, その他 iconv がサポートする文字セット*

返り値

リクエストを表す XML 文字列を返します。

例

Example#1 XMLRPC クライアント関数の例

```
<?php
$request = xmlrpc_encode_request("method", array(1, 2, 3));
$context = stream_context_create(array('http' => array(
    'method' => "POST",
    'header' => "Content-Type: text/xml",
    'content' => $request
)));
$file = file_get_contents("http://www.example.com/xmlrpc", false, $context);
$response = xmlrpc_decode($file);
if (xmlrpc_is_fault($response)) {
    trigger_error("xmlrpc: $response[faultString] ($response[faultCode])");
} else {
    print_r($response);
}
?>
```

参考

- *stream_context_create()*
- *file_get_contents()*
- *xmlrpc_decode()*

---

xmlrpc_encode

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_encode — PHP の値に関する XML を生成する

説明

string **xmlrpc_encode** *(mixed $value)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_get_type

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_get_type — PHP の値に関する xmlrpc 型を取得する

説明

string **xmlrpc_get_type** *(mixed $value)*

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

この関数は特に base64 及び datetime 文字列で有用です。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_is_fault

(PHP 4 >= 4.3.0, PHP 5)

xmlrpc_is_fault — 配列の値が XMLRPC の失敗であるかどうかを調べる

説明

bool **xmlrpc_is_fault** (array $arg)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

arg

xmlrpc_decode() が返す配列。

返り値

引数が失敗を表す場合に **TRUE**、それ以外の場合に **FALSE** を返します。 失敗の内容は $arg["faultString"]、失敗のコードは $arg["faultCode"] に格納されます。

例

xmlrpc_encode_request() の例を参照ください。

参考

- xmlrpc_decode()

---

xmlrpc_parse_method_descriptions

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_parse_method_descriptions — XML を、メソッド説明のリストにデコードする

説明

array **xmlrpc_parse_method_descriptions** (string $xml)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_server_add_introspection_data

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_server_add_introspection_data — introspection ドキュメントを追加する

説明

int **xmlrpc_server_add_introspection_data** (resource $server , array $desc)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_server_call_method

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_server_call_method — XML リクエストをパースし、メソッドをコールする

説明

string **xmlrpc_server_call_method** (resource $server , string $xml , mixed $user_data [, array $output_options])

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_server_create

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_server_create — xmlrpc サーバを作成する

説明

resource **xmlrpc_server_create** (void)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_server_destroy

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_server_destroy — サーバリソースを破棄する

説明

int **xmlrpc_server_destroy** (resource $server)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_server_register_introspection_callback

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_server_register_introspection_callback — ドキュメントを生成する PHP 関数を登録する

説明

bool **xmlrpc_server_register_introspection_callback** (resource $server , string $function)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_server_register_method

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_server_register_method — メソッド名が一致するメソッドリソースに PHP 関数を登録する

説明

bool **xmlrpc_server_register_method** (resource $server , string $method_name , string $function)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

---

xmlrpc_set_type

(PHP 4 >= 4.0.7, PHP 5)

xmlrpc_set_type — PHP 文字列型用に xmlrpc 型、base64 または datetime を設定する

説明

bool **xmlrpc_set_type** (string &$value , string $type)

警告

この関数は、 実験的 なものです。この関数の動作・名前・その他ドキュメントに書かれている事項は、予告なく、将来的な PHP のリリースにおいて変更される可能性があります。この関数は自己責任で使用してください。

パラメータ

value

型を設定する値。

type

'base64' あるいは 'datetime'。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 成功した場合、value はオブジェクト型に変換されます。

エラー / 例外

XMLRPC がサポートしていない型を指定した場合は E_WARNING が発生します。

目次

XMLReader 関数

導入

XMLReader 拡張モジュールは、プル型の XML パーサです。ドキュメント ストリーム内をカーソル風に進んでいき、その途中の各ノードで立ち止まります。

エンコーディング

内部的には、libxml は UTF-8 エンコーディングを使用しています。 したがって、取得したデータはすべて UTF-8 エンコーディングに変換されます。

インストール手順

XMLReader 拡張モジュールは、PHP 5.0.0 では PECL から取得可能です。 また、PHP 5.1.0 ではデフォルトで組み込まれ、有効になっています。configure 時に、引数 --enable-xmlreader (5.1.0 より前では --with-xmlreader) を指定することでも有効にできます。libxml 拡張モジュールが必要です。

定義済みクラス

XMLReader

メソッド

- XMLReader::close - XMLReader の入力を閉じる
- XMLReader::expand - 現在のノードを DOM ノードにエクスポートする
- XMLReader::getAttribute - 名前をもとに、属性の値を取得する
- XMLReader::getAttributeNo - 位置をもとに、属性の値を取得する
- XMLReader::getAttributeNs - 名前および URI をもとに、属性の値を取得する
- XMLReader::getParserProperty - パーサのプロパティが設定されているかどうかを示す
- XMLReader::isValid - 妥当なドキュメントであるかどうかを示す
- XMLReader::lookupNamespace - ノードのスコープで、プレフィックスの URI を取得する
- XMLReader::moveToAttribute - リーダを、指定した名前の属性に移動する
- XMLReader::moveToAttributeNo - リーダを、指定したインデックスの属性に移動する
- XMLReader::moveToAttributeNs - リーダを、指定した名前および URI の属性に移動する
- XMLReader::moveToElement - 現在の属性ノードの親要素に移動する
- XMLReader::moveToFirstAttribute - ノードの最初の属性に移動する
- XMLReader::moveToNextAttribute - ノードの次の属性に移動する
- XMLReader::next - 子を飛ばして、次の要素に移動する
- XMLReader::open - パースする URI を設定する
- XMLReader::read - ストリーム内の次のノードに移動する
- XMLReader::setParserProperty - パーサのプロパティを設定する
- XMLReader::setRelaxNGSchema - 妥当性を検証するための RelaxNG スキーマの URI を設定する
- XMLReader::setRelaxNGSchemaSource - 妥当性を検証するための RelaxNG スキーマを含む文字列を設定する
- XMLReader::XML - パースする文字列データを設定する

プロパティ

名前	型	読み込み専用	説明
attributeCount	int	yes	ノード上の属性の数
baseURI	string	yes	ノードのベース URI
depth	int	yes	ツリー内でのノードの階層 (0 から数える)
hasAttributes	bool	yes	ノードが属性を保持しているかどうか
hasValue	bool	yes	ノードがテキストの値を保持しているかどうか
isDefault	bool	yes	属性が DTD のデフォルトかどうか
isEmptyElement	bool	yes	ノードが空要素のタグかどうか
localName	string	yes	ノードのローカル名
name	string	yes	ノードの限定名
namespaceURI	string	yes	ノードに関連付けられた名前空間の URI
nodeType	int	yes	ノードの型
prefix	string	yes	ノードに関連付けられた名前空間のプレフィックス
value	string	yes	ノードのテキスト値
xmlLang	string	yes	ノードが存在する xml:lang スコープ

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

警告

PHP 5.1 以降では、XMLReader はクラス定数を使用します。それより前の リリースでは、**XMLREADER_ELEMENT** のような形式の グローバル定数を使用します。

XMLReader ノード型

定数	値	説明
XMLReader::NONE (integer)	0	ノード型なし
XMLReader::ELEMENT (integer)	1	開始要素
XMLReader::ATTRIBUTE (integer)	2	属性ノード
XMLReader::TEXT (integer)	3	テキストノード
XMLReader::CDATA (integer)	4	CDATA ノード
XMLReader::ENTITY_REF (integer)	5	エンティティ参照ノード
XMLReader::ENTITY (integer)	6	エンティティ宣言ノード
XMLReader::PI (integer)	7	処理命令 (Processing Instruction) ノード
XMLReader::COMMENT (integer)	8	コメントノード
XMLReader::DOC (integer)	9	文書ノード
XMLReader::DOC_TYPE (integer)	10	文書型ノード
XMLReader::DOC_FRAGMENT (integer)	11	文書片ノード
XMLReader::NOTATION (integer)	12	記法ノード
XMLReader::WHITESPACE (integer)	13	Whitespace ノード
XMLReader::SIGNIFICANT_WHITESPACE (integer)	14	Significant Whitespace ノード
XMLReader::END_ELEMENT (integer)	15	終了要素
XMLReader::END_ENTITY (integer)	16	終了エンティティ
XMLReader::XML_DECLARATION (integer)	17	XML 宣言ノード

XMLReader パーサオプション

定数	値	説明
XMLReader::LOADDTD (integer)	1	DTD を読み込むが、妥当性は検証しない
XMLReader::DEFAULTATTRS (integer)	2	DTD およびデフォルト属性を読み込むが、妥当性は検証しない
XMLReader::VALIDATE (integer)	3	DTD を読み込み、パース時に妥当性を検証する
XMLReader::SUBST_ENTITIES (integer)	4	エンティティを参照で置き換える

XMLReader::close

(PHP 5 >= 5.1.2)

XMLReader::close — XMLReader の入力を閉じる

説明

XMLReader
bool ***close*** *(void)*

XMLReader オブジェクトが現在パースしている入力を閉じます。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLReader::open*
- *XMLReader::XML*

XMLReader::expand

(PHP 5 >= 5.1.2)

XMLReader::expand — 現在のノードのコピーを DOM オブジェクトとして返す

説明

XMLReader
DOMNode ***expand*** *(void)*

このメソッドは、現在のノードをコピーして適切な DOM オブジェクトを返します。

返り値

DOMNode か、エラー時には FALSE を返します。

XMLReader::getAttribute

(PHP 5 >= 5.1.2)

XMLReader::getAttribute — 名前をもとに、属性の値を取得する

説明

XMLReader
string **getAttribute** (string $name)

指定した名前の属性の値を返します。属性が存在しなかったり 現在位置が要素ノードでなかったりした場合には空の文字列を返します。

パラメータ

name

属性の名前。

返り値

属性の値を返します。指定した名前 name の 属性が見つからなかったり、現在位置が要素ではなかったりした場合には 空の文字列を返します。

参考

- XMLReader::getAttributeNo
- XMLReader::getAttributeNs

XMLReader::getAttributeNo

(PHP 5 >= 5.1.2)

XMLReader::getAttributeNo — インデックスをもとに、属性の値を取得する

説明

XMLReader
string **getAttributeNo** (int $index)

その位置をもとにして属性の値を返します。属性が存在しなかったり 現在位置が要素ノードでなかったりした場合には空の文字列を返します。

パラメータ

index

属性の位置。

返り値

属性の値を返します。指定した位置 index に 属性が見つからなかったり、現在位置が要素ではなかったりした場合には 空の文字列を返します。

参考

- XMLReader::getAttribute
- XMLReader::getAttributeNs

XMLReader::getAttributeNs

(PHP 5 >= 5.1.2)

XMLReader::getAttributeNs — 名前および URI をもとに、属性の値を取得する

説明

XMLReader
string **getAttributeNs** (string $localName , string $namespaceURI)

名前および名前空間 URI をもとにして属性の値を返します。 属性が存在しなかったり、現在位置が要素ノードでなかったりした場合には 空の文字列を返します。

パラメータ

localName

ローカルの名前。

namespaceURI

名前空間の URI。

返り値

属性の値を返します。指定した localName および namespaceURI に属性が見つからなかったり、 現在位置が要素ではなかったりした場合には空の文字列を返します。

参考

- XMLReader::getAttribute
- XMLReader::getAttributeNo

XMLReader::getParserProperty

(PHP 5 >= 5.1.2)

XMLReader::getParserProperty — 指定したプロパティが設定されているかどうかを示す

説明

XMLReader
bool **getParserProperty** (int $property)

指定したプロパティが設定されているかどうかを示します。

パラメータ

property

パーサオプション定数 のうちのひとつ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLReader::setParserProperty

XMLReader::isValid

(PHP 5 >= 5.1.2)

XMLReader::isValid — パースしているドキュメントの妥当性を示す

説明

XMLReader
bool **isValid** (void)

パースしているドキュメントが妥当なものであるかどうかを論理型で返します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLReader::setParserProperty
- XMLReader::setRelaxNGSchema
- XMLReader::setRelaxNGSchemaSource

XMLReader::lookupNamespace

(PHP 5 >= 5.1.2)

XMLReader::lookupNamespace — プレフィックスから、名前空間を検索する

説明

XMLReader
bool **lookupNamespace** (string $prefix)

指定したプレフィックスをもとに、スコープの名前空間を検索します。

パラメータ

prefix

プレフィックスを含む文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

XMLReader::moveToAttribute

(PHP 5 >= 5.1.2)

XMLReader::moveToAttribute — 指定した名前の属性にカーソルを移動する

説明

```
XMLReader
bool moveToAttribute ( string $name )
```

指定した名前の属性にカーソルを移動します。

パラメータ

name

属性の名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLReader::moveToElement
- XMLReader::moveToAttributeNo
- XMLReader::moveToAttributeNs
- XMLReader::moveToFirstAttribute

XMLReader::moveToAttributeNo

(PHP 5 >= 5.1.2)

XMLReader::moveToAttributeNo — 指定したインデックスの属性にカーソルを移動する

説明

```
XMLReader
bool moveToAttributeNo ( int $index )
```

指定した位置の属性にカーソルを移動します。

パラメータ

index

属性の位置。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNs
- XMLReader::moveToFirstAttribute

XMLReader::moveToAttributeNs

(PHP 5 >= 5.1.2)

XMLReader::moveToAttributeNs — 指定した名前の属性にカーソルを移動する

説明

```
XMLReader
bool moveToAttributeNs ( string $localName , string $namespaceURI )
```

指定した名前空間内で、指定した名前の属性にカーソルを移動します。

パラメータ

localName

ローカル名。

namespaceURI

名前空間の URI。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNo
- XMLReader::moveToFirstAttribute

XMLReader::moveToElement

(PHP 5 >= 5.1.2)

XMLReader::moveToElement — 現在の属性の親要素にカーソルを移動する

説明

XMLReader
bool **moveToElement** (void)

現在の属性の親要素にカーソルを移動します。

返り値

成功した場合に **TRUE**、失敗した場合、 あるいはメソッドがコールされた際の位置が属性ノードではなかった場合に **FALSE** を返します。

参考

- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNo
- XMLReader::moveToAttributeNs
- XMLReader::moveToFirstAttribute

XMLReader::moveToFirstAttribute

(PHP 5 >= 5.1.2)

XMLReader::moveToFirstAttribute — 最初の属性にカーソルを移動する

説明

XMLReader
bool **moveToFirstAttribute** (void)

最初の属性にカーソルを移動します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::moveToAttributeNo
- XMLReader::moveToAttributeNs
- XMLReader::moveToNextAttribute

XMLReader::moveToNextAttribute

(PHP 5 >= 5.1.2)

XMLReader::moveToNextAttribute — *次の属性にカーソルを移動する*

説明

XMLReader
bool ***moveToNextAttribute*** *(void)*

現在位置が属性ノードの場合は次の属性にカーソルを移動します。 現在位置が要素ノードの場合は最初の属性にカーソルを移動します。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *XMLReader::moveToElement*
- *XMLReader::moveToAttribute*
- *XMLReader::moveToAttributeNo*
- *XMLReader::moveToAttributeNs*
- *XMLReader::moveToFirstAttribute*

XMLReader::next

(PHP 5 >= 5.1.2)

XMLReader::next — *すべてのサブツリーを飛ばして、次のノードにカーソルを移動する*

説明

XMLReader
bool ***next*** *([string $localname])*

すべてのサブツリーを飛ばして、カーソルを次のノードに移動します。

パラメータ

localname

移動先のノードの名前。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *XMLReader::next*
- *XMLReader::moveToElement*
- *XMLReader::moveToAttribute*

XMLReader::open

(PHP 5 >= 5.1.2)

XMLReader::open — *パースする XML を含む URI を設定する*

説明

XMLReader
bool ***open*** *(string $URI [, string $encoding [, int $options]])*

パースされる XML ドキュメントを含む URI を設定します。

パラメータ

URI

ドキュメントを指す URI。

encoding

ドキュメントのエンコーディングあるいは ***NULL****。*

options

XMLReader のオプション定数のビットマスク。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

変更履歴

バージョン	説明
5.2.0	*encoding* および *options* が追加されました。

参考

- XMLReader::XML
- XMLReader::close

XMLReader::read

(PHP 5 >= 5.1.2)

XMLReader::read — ドキュメント内の次のノードに移動する

説明

```
XMLReader
bool read ( void )
```

ドキュメント内の次のノードにカーソルを移動します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLReader::moveToElement
- XMLReader::moveToAttribute
- XMLReader::next

XMLReader::setParserProperty

(PHP 5 >= 5.1.2)

XMLReader::setParserProperty — パーサのオプションを設定または設定解除する

説明

```
XMLReader
bool setParserProperty ( int $property , bool $value )
```

パーサのオプションを設定または設定解除します。オプションは、 xmlreader-open() あるいは xmlreader-xml() がコールされた後で、かつ xmlreader-read() が最初にコールされる前に 設定する必要があります。

パラメータ

property

パーサオプション定数 のひとつ。

value

TRUE を設定するとオプションが有効になり、それ以外を設定すると 無効になります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

XMLReader::setRelaxNGSchema

(PHP 5 >= 5.2.0)

XMLReader::setRelaxNGSchema — RelaxNG スキーマのファイル名あるいは URI を設定する

説明

```
XMLReader
bool setRelaxNGSchema ( string $filename )
```

検証の際に使用する RelaxNG スキーマのファイル名あるいは URI を 設定します。

パラメータ

filename

RelaxNG スキーマを指すファイル名あるいは URI。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLReader::setRelaxNGSchemaSource*

XMLReader::setRelaxNGSchemaSource

(PHP 5 >= 5.1.2)

XMLReader::setRelaxNGSchemaSource — RelaxNG スキーマを含むデータを設定する

説明

XMLReader
bool **setRelaxNGSchemaSource** *(string $source)*

検証の際に使用する RelaxNG スキーマを含むデータを設定します。

パラメータ

source

RelaxNG スキーマを含む文字列。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLReader::setRelaxNGSchema*

XMLReader::XML

(PHP 5 >= 5.1.2)

XMLReader::XML — パースする XML を含むデータを設定する

説明

XMLReader
bool **xml** *(string $source [, string $encoding [, int $options]])*

パースする XML を含むデータを設定します。

パラメータ

source

パースされる XML を含む文字列。

encoding

*ドキュメントのエンコーディングあるいは **NULL**。*

options

XMLReader のオプション定数のビットマスク。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

変更履歴

バージョン	説明
5.2.0	*encoding* および *options* が追加されました。

参考

- *XMLReader::open*
- *XMLReader::close*

目次

XMLWriter 関数

導入

これは、XMLWriter 拡張モジュールです。 libxml の xmlWriter API をラップしています。

この拡張モジュールは、キャッシュ処理をしない前進のみのライターで、 XML データを含むストリームあるいはファイルを作成します。

この拡張モジュールは、オブジェクト指向形式と手続き型のどちらでも使用可能です。 全メソッドについて、その両方のコール方法を説明しています。

定義済みクラス

XMLWriter

メソッド

- XMLWriter::startCData - CDATA タグを開始する
- XMLWriter::startComment - コメントを開始する
- XMLWriter::startDocument - ドキュメントタグを作成する
- XMLWriter::startDTDAttlist - DTD AttList を開始する
- XMLWriter::startDTDElement - DTD 要素を開始する
- XMLWriter::startDTDEntity - DTD エンティティを開始する
- XMLWriter::startDTD - DTD タグを開始する
- XMLWriter::startElementNS - 名前空間つき要素タグを開始する
- XMLWriter::startElement - 要素タグを開始する
- XMLWriter::startPI - 処理命令タグを開始する
- XMLWriter::text - テキストを書き込む
- XMLWriter::writeAttributeNS - 名前空間つき属性全体を書き込む
- XMLWriter::writeAttribute - 属性全体を書き込む
- XMLWriter::writeCData - CDATA タグ全体を書き込む
- XMLWriter::writeComment - コメント全体を書き込む
- XMLWriter::writeDTDAttlist - DTD AttList タグ全体を書き込む
- XMLWriter::writeDTDElement - DTD 要素タグ全体を書き込む
- XMLWriter::writeDTDEntity - DTD エンティティタグ全体を書き込む
- XMLWriter::writeDTD - DTD タグ全体を書き込む
- XMLWriter::writeElementNS - 名前空間つき要素タグ全体を書き込む
- XMLWriter::writeElement - 要素タグ全体を書き込む
- XMLWriter::writePI - 処理命令を書き込む
- XMLWriter::writeRaw - 生の XML テキストを書き込む

リソース型

手続き型の XMLWriter 拡張モジュールでは、一種類のリソースを使用します。 これは、xmlwriter_open_memory() あるいは xmlwriter_open_uri() が返すものです。

定義済み定数

定数は定義されていません。

XMLWriter::endAttribute

(No version information available, might be only in CVS)

XMLWriter::endAttribute — 属性を終了する

説明

オブジェクト指向型

XMLWriter
bool **endAttribute** (void)

手続き型

bool **xmlwriter_end_attribute** (resource $xmlwriter)

現在の属性を終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startAttribute
- XMLWriter::startAttributeNS

- XMLWriter::writeAttribute
- XMLWriter::writeAttributeNS

XMLWriter::endCData

(No version information available, might be only in CVS)

XMLWriter::endCData — 現在の CDATA を終了する

説明

オブジェクト指向型

XMLWriter
bool **endCData** (void)

手続き型

bool **xmlwriter_end_cdata** (resource $xmlwriter)

現在の CDATA セクションを終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startCData
- XMLWriter::writeCData

XMLWriter::endComment

(No version information available, might be only in CVS)

XMLWriter::endComment — コメントの終了部を作成する

説明

オブジェクト指向型

XMLWriter
bool **endComment** (void)

手続き型

bool **xmlwriter_end_comment** (resource $xmlwriter)

現在のコメントを終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startComment
- XMLWriter::writeComment

XMLWriter::endDocument

(No version information available, might be only in CVS)

XMLWriter::endDocument — 現在のドキュメントを終了する

説明

オブジェクト指向型

XMLWriter
bool **endDocument** (void)

手続き型

bool **xmlwriter_end_document** (resource $xmlwriter)

現在のドキュメントを終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startDocument

XMLWriter::endDTDAttlist

(No version information available, might be only in CVS)

XMLWriter::endDTDAttlist — 現在の DTD 属性リストを終了する

説明

オブジェクト指向型

XMLWriter
bool **endDTDAttlist** (void)

手続き型

bool **xmlwriter_end_dtd_attlist** (resource $xmlwriter)

現在の DTD 属性リストを終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startDTDAttlist
- XMLWriter::writeDTDAttlist

XMLWriter::endDTDElement

(No version information available, might be only in CVS)

XMLWriter::endDTDElement — 現在の DTD 要素を終了する

説明

オブジェクト指向型

XMLWriter
bool **endDTDElement** (void)

手続き型

bool **xmlwriter_end_dtd_element** (resource $xmlwriter)

現在の DTD 要素を終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLWriter::startDTDElement*
- *XMLWriter::writeDTDElement*

XMLWriter::endDTDEntity

(No version information available, might be only in CVS)

XMLWriter::endDTDEntity — 現在の DTD エンティティを終了する

説明

オブジェクト指向型

XMLWriter
bool ***endDTDEntity*** *(void)*

手続き型

bool ***xmlwriter_end_dtd_entity*** *(resource $xmlwriter)*

現在の DTD エンティティを終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLWriter::startDTDEntity*
- *XMLWriter::writeDTDEntity*

XMLWriter::endDTD

(No version information available, might be only in CVS)

XMLWriter::endDTD — 現在の DTD を終了する

説明

オブジェクト指向型

XMLWriter
bool ***endDTD*** *(void)*

手続き型

bool ***xmlwriter_end_dtd*** *(resource $xmlwriter)*

ドキュメントの DTD を終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLWriter::startDTD*

- XMLWriter::writeDTD

XMLWriter::endElement

(No version information available, might be only in CVS)

XMLWriter::endElement — 現在の要素を終了する

説明

オブジェクト指向型

XMLWriter
bool **endElement** (void)

手続き型

bool **xmlwriter_end_element** (resource $xmlwriter)

現在の要素を終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startElement
- XMLWriter::writeElement

XMLWriter::endPI

(No version information available, might be only in CVS)

XMLWriter::endPI — 現在の PI (処理命令) を終了する

説明

オブジェクト指向型

XMLWriter
bool **endPI** (void)

手続き型

bool **xmlwriter_end_pi** (resource $xmlwriter)

現在の処理命令を終了します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startPI
- XMLWriter::writePI

XMLWriter::flush

(PHP 5 >= 5.1.2, PECL xmlwriter:1.0-2.0.4)

XMLWriter::flush — 現在のバッファをフラッシュする

説明

オブジェクト指向型

XMLWriter
mixed ***flush*** *([bool $empty])*

手続き型

mixed ***xmlwriter_flush*** *(resource $xmlwriter [, bool $empty])*

現在のバッファをフラッシュします。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

empty

*バッファを空にするかどうか。デフォルトは **TRUE** です。*

返り値

ライターをメモリにオープンした場合は、この関数は出来上がった XML バッファを返します。 そうではなく URI を使用している場合は、この関数はバッファを書き込み、 書き込んだバイト数を返します。

XMLWriter::fullEndElement

(No version information available, might be only in CVS)

XMLWriter::fullEndElement — 現在の要素を終了する

説明

オブジェクト指向型

XMLWriter
bool ***fullEndElement*** *(void)*

手続き型

bool ***xmlwriter_full_end_element*** *(resource $xmlwriter)*

現在の xml 要素を終了します。要素が空であっても終了タグを書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLWriter::endElement*

XMLWriter::openMemory

(No version information available, might be only in CVS)

XMLWriter::openMemory — 文字列の出力にメモリを使用する新しい xmlwriter を作成する

説明

オブジェクト指向型

XMLWriter
bool ***openMemory*** *(void)*

手続き型

resource ***xmlwriter_open_memory*** *(void)*

文字列の出力にメモリを使用する新しい XMLWriter を作成します。

パラメータ

返り値

*オブジェクト指向型: 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。。*

*手続き型: 成功した場合に、その後の xmlwriter 関数で使用するための新しい xmlwriter リソース、 エラーの場合に **FALSE** を返します。*

参考

- XMLWriter::openURI

XMLWriter::openURI

(No version information available, might be only in CVS)

XMLWriter::openURI — ソース URI を指定して新しい xmlwriter を作成する

説明

オブジェクト指向型

XMLWriter
bool ***openURI*** *(string $uri)*

手続き型

resource ***xmlwriter_open_uri*** *(string $uri)*

uri を指定した新しい XMLWriter を作成します。

パラメータ

uri

出力するリソースの URI。

返り値

*オブジェクト指向型: 成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。。*

*手続き型: 成功した場合に、その後の xmlwriter 関数で使用するための新しい xmlwriter リソース、 エラーの場合に **FALSE** を返します。*

参考

- XMLWriter::openMemory

XMLWriter::outputMemory

(No version information available, might be only in CVS)

XMLWriter::outputMemory — 現在のバッファを返す

説明

オブジェクト指向型

XMLWriter
bool ***outputMemory*** *([bool $flush])*

手続き型

string ***xmlwriter_output_memory*** *(resource $xmlwriter [, bool $flush])*

現在のバッファを返します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

flush

*出力バッファをフラッシュするかどうか。デフォルトは **TRUE**。*

返り値

現在のバッファを文字列で返します。

参考

- XMLWriter::flush

XMLWriter::setIndentString

(No version information available, might be only in CVS)

XMLWriter::setIndentString — 字下げに使用する文字列を設定する

説明

オブジェクト指向型

XMLWriter
bool **setIndentString** (string $indentString)

手続き型

bool **xmlwriter_set_indent_string** (resource $xmlwriter , string $indentString)

文字列を設定します。この文字を使用して、結果の xml の要素/属性 を字下げします。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

indentString

字下げ用の文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::setIndent

XMLWriter::setIndent

(No version information available, might be only in CVS)

XMLWriter::setIndent — 字下げの on/off を切り替える

説明

オブジェクト指向型

XMLWriter
bool **setIndent** (bool $indent)

手続き型

bool **xmlwriter_set_indent** (resource $xmlwriter , bool $indent)

字下げの on/off を切り替えます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

indent

字下げを有効にするかどうか。デフォルトは **FALSE** です。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::setIndentString

XMLWriter::startAttributeNS

(No version information available, might be only in CVS)

XMLWriter::startAttributeNS — 名前空間つきの属性を開始する

説明

オブジェクト指向型

XMLWriter

bool ***startAttributeNS*** *(string $prefix , string $name , string $uri)*

手続き型

bool ***xmlwriter_start_attribute_ns*** *(resource $xmlwriter , string $prefix , string $name , string $uri)*

名前空間つきの属性を開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

prefix

名前空間プレフィックス。

name

属性の名前。

uri

名前空間 URI。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *XMLWriter::startAttribute*
- *XMLWriter::endAttribute*
- *XMLWriter::writeAttribute*
- *XMLWriter::writeAttributeNS*

XMLWriter::startAttribute

(No version information available, might be only in CVS)

XMLWriter::startAttribute — 属性を開始する

説明

オブジェクト指向型

XMLWriter
bool ***startAttribute*** *(string $name)*

手続き型

bool ***xmlwriter_start_attribute*** *(resource $xmlwriter , string $name)*

属性を開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

属性の名前。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *XMLWriter::startAttributeNS*
- *XMLWriter::endAttribute*
- *XMLWriter::writeAttribute*
- *XMLWriter::writeAttributeNS*

XMLWriter::startCData

(No version information available, might be only in CVS)

XMLWriter::startCData — CDATA の開始タグを作成する

説明

オブジェクト指向型

XMLWriter
bool **startCData** (void)

手続き型

bool **xmlwriter_start_cdata** (resource $xmlwriter)

CDATA を開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::endCData
- XMLWriter::writeCData

XMLWriter::startComment

(No version information available, might be only in CVS)

XMLWriter::startComment — コメントを開始する

説明

オブジェクト指向型

XMLWriter
bool **startComment** (void)

手続き型

bool **xmlwriter_start_comment** (resource $xmlwriter)

コメントを開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::endComment
- XMLWriter::writeComment

XMLWriter::startDocument

(No version information available, might be only in CVS)

XMLWriter::startDocument — ドキュメントタグを作成する

説明

オブジェクト指向型

XMLWriter
bool **startDocument** ([string $version [, string $encoding [, string $standalone]]])

手続き型

bool **xmlwriter_start_document** (resource $xmlwriter [, string $version [, string $encoding [, string $standalone]]])

ドキュメントを開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

version

XML 宣言で使用するドキュメントのバージョン番号。 デフォルトは 1.0。

encoding

XML 宣言で使用するドキュメントのエンコーディング。 デフォルトは **NULL**。

standalone

yes あるいは no。 デフォルトは **NULL**。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::endDocument

XMLWriter::startDTDAttlist

(No version information available, might be only in CVS)

XMLWriter::startDTDAttlist — DTD 属性リストを開始する

説明

オブジェクト指向型

XMLWriter
bool **startDTDAttlist** (string $name)

手続き型

bool **xmlwriter_start_dtd_attlist** (resource $xmlwriter , string $name)

DTD 属性リストを開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

属性リストの名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::endDTDAttlist
- XMLWriter::writeDTDAttlist

XMLWriter::startDTDElement

(No version information available, might be only in CVS)

XMLWriter::startDTDElement — DTD 要素を開始する

説明

オブジェクト指向型

XMLWriter
bool **startDTDElement** (string $qualifiedName)

手続き型

bool **xmlwriter_start_dtd_element** (resource $xmlwriter , string $qualifiedName)

DTD 要素を開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

qualifiedName

作成するドキュメント型の修飾名。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::endDTDElement
- XMLWriter::writeDTDElement

XMLWriter::startDTDEntity

(No version information available, might be only in CVS)

XMLWriter::startDTDEntity — DTD エンティティを開始する

説明

オブジェクト指向型

```
XMLWriter
bool startDTDEntity ( string $name , bool $isparam )
```

手続き型

```
bool xmlwriter_start_dtd_entity ( resource $xmlwriter , string $name , bool $isparam )
```

DTD エンティティを開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

エンティティの名前。

isparam

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::endDTDEntity
- XMLWriter::writeDTDEntity

XMLWriter::startDTD

(No version information available, might be only in CVS)

XMLWriter::startDTD — DTD の開始タグを作成する

説明

オブジェクト指向型

```
XMLWriter
bool startDTD ( string $qualifiedName [, string $publicId [, string $systemId ]] )
```

手続き型

```
bool xmlwriter_start_dtd ( resource $xmlwriter , string $qualifiedName [, string $publicId [, string $systemId ]] )
```

DTD を開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

qualifiedName

作成するドキュメント型の修飾名。

publicId

外部サブセットの公開識別子。

systemId

外部サブセットのシステム識別子。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- XMLWriter::endDTD
- XMLWriter::writeDTD

XMLWriter::startElementNS

(No version information available, might be only in CVS)

XMLWriter::startElementNS — 名前空間つき要素の開始タグを作成する

説明

オブジェクト指向型

```
XMLWriter
bool startElementNS ( string $prefix , string $name , string $uri )
```

手続き型

```
bool xmlwriter_start_element_ns ( resource $xmlwriter , string $prefix , string $name , string $uri )
```

名前空間つき要素を開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

prefix

名前空間プレフィックス。

name

要素名。

uri

名前空間 URI。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

参考

- XMLWriter::endElement
- XMLWriter::writeElementNS

XMLWriter::startElement

(No version information available, might be only in CVS)

XMLWriter::startElement — 要素の開始タグを作成する

説明

オブジェクト指向型

```
XMLWriter
bool startElement ( string $name )
```

手続き型

bool ***xmlwriter_start_element*** *(resource $xmlwriter , string $name)*

要素を開始します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

要素名。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *XMLWriter::endElement*
- *XMLWriter::writeElement*

XMLWriter::startPI

(No version information available, might be only in CVS)

XMLWriter::startPI — PI (処理命令) の開始タグを作成する

説明

オブジェクト指向型

XMLWriter
bool ***startPI*** *(string $target)*

手続き型

bool ***xmlwriter_start_pi*** *(resource $xmlwriter , string $target)*

処理命令の開始タグを作成します。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

target

処理命令の対象。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

参考

- *XMLWriter::endPI*
- *XMLWriter::writePI*

XMLWriter::text

(PHP 5 >= 5.1.2, PECL xmlwriter:0.1-2.0.4)

XMLWriter::text — テキストを書き込む

説明

オブジェクト指向型

XMLWriter
bool ***text*** *(string $content)*

手続き型

bool ***xmlwriter_text*** *(resource $xmlwriter , string $content)*

テキストを書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

content

テキストの内容。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

XMLWriter::writeAttributeNS

(No version information available, might be only in CVS)

XMLWriter::writeAttributeNS — 名前空間つき属性全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool ***writeAttributeNS*** *(string $prefix , string $name , string $uri , string $content)*

手続き型

bool ***xmlwriter_write_attribute_ns*** *(resource $xmlwriter , string $prefix , string $name , string $uri , string $content)*

名前空間つき属性全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

prefix

名前空間プレフィックス。

name

属性名。

uri

名前空間 URI。

content

属性の値。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- *XMLWriter::writeAttribute*
- *XMLWriter::startAttribute*
- *XMLWriter::startAttributeNS*
- *XMLWriter::endAttribute*

XMLWriter::writeAttribute

(No version information available, might be only in CVS)

XMLWriter::writeAttribute — 属性全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool ***writeAttribute*** *(string $name , string $value)*

手続き型

bool ***xmlwriter_write_attribute*** *(resource $xmlwriter , string $name , string $value)*

属性全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

属性の名前。

value

属性の値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::writeAttributeNS
- XMLWriter::startAttribute
- XMLWriter::startAttributeNS
- XMLWriter::endAttribute

XMLWriter::writeCData

(No version information available, might be only in CVS)

XMLWriter::writeCData — CDATA タグ全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool **writeCData** (string $content)

手続き型

bool **xmlwriter_write_cdata** (resource $xmlwriter , string $content)

CDATA 全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

content

CDATA の内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startCData
- XMLWriter::endCData

XMLWriter::writeComment

(No version information available, might be only in CVS)

XMLWriter::writeComment — コメントタグ全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool **writeComment** (string $content)

手続き型

bool **xmlwriter_write_comment** (resource $xmlwriter , string $content)

コメント全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

content

コメントの内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startComment
- XMLWriter::endComment

XMLWriter::writeDTDAttlist

(No version information available, might be only in CVS)

XMLWriter::writeDTDAttlist — DTD 属性リストタグ全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool **writeDTDAttlist** (string $name , string $content)

手続き型

bool **xmlwriter_write_dtd_attlist** (resource $xmlwriter , string $name , string $content)

DTD 属性リストを書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

DTD 属性リストの名前。

content

DTD 属性リストの内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startDTDAttlist
- XMLWriter::endDTDAttlist

XMLWriter::writeDTDElement

(No version information available, might be only in CVS)

XMLWriter::writeDTDElement — DTD 要素タグ全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool **writeDTDElement** (string $name , string $content)

手続き型

bool **xmlwriter_write_dtd_element** (resource $xmlwriter , string $name , string $content)

DTD 要素全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

DTD 要素の名前。

content

要素の内容。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- XMLWriter::startDTDElement
- XMLWriter::endDTDElement

XMLWriter::writeDTDEntity

(No version information available, might be only in CVS)

XMLWriter::writeDTDEntity — DTD エンティティタグ全体を書き込む

説明

オブジェクト指向型

```
XMLWriter
bool writeDTDEntity ( string $name , string $content )
```

手続き型

```
bool xmlwriter_write_dtd_entity ( resource $xmlwriter , string $name , string $content )
```

DTD エンティティ全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

エンティティの名前。

content

エンティティの内容。

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

参考

- XMLWriter::startDTDEntity
- XMLWriter::endDTDEntity

XMLWriter::writeDTD

(No version information available, might be only in CVS)

XMLWriter::writeDTD — DTD タグ全体を書き込む

説明

オブジェクト指向型

```
XMLWriter
bool writeDTD ( string $name [, string $publicId [, string $systemId [, string $subset ]]] )
```

手続き型

```
bool xmlwriter_write_dtd ( resource $xmlwriter , string $name [, string $publicId [, string $systemId [, string $subset
]]] )
```

DTD 全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

DTD 名。

publicId

外部サブセットの公開識別子。

systemId

外部サブセットのシステム識別子。

subset

DTD の内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startDTD
- XMLWriter::endDTD

XMLWriter::writeElementNS

(No version information available, might be only in CVS)

XMLWriter::writeElementNS — 名前空間つき要素タグ全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool **writeElementNS** (string $prefix , string $name , string $uri [, string $content])

手続き型

bool **xmlwriter_write_element_ns** (resource $xmlwriter , string $prefix , string $name , string $uri [, string $content])

名前空間つき要素タグ全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

prefix

名前空間プレフィックス。

name

要素名。

uri

名前空間 URI。

content

要素の内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
PHP 5.2.3	*content* パラメータがオプションとなりました。

参考

- XMLWriter::startElementNS

- XMLWriter::endElement
- XMLWriter::writeElement

XMLWriter::writeElement

(No version information available, might be only in CVS)

XMLWriter::writeElement — 要素タグ全体を書き込む

説明

オブジェクト指向型

XMLWriter
bool **writeElement** (string $name [, string $content])

手続き型

bool **xmlwriter_write_element** (resource $xmlwriter , string $name [, string $content])

要素タグ全体を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

name

要素名。

content

要素の内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

変更履歴

バージョン	説明
PHP 5.2.3	content パラメータがオプションとなりました。

参考

- XMLWriter::startElement
- XMLWriter::endElement
- XMLWriter::writeElementNS

XMLWriter::writePI

(No version information available, might be only in CVS)

XMLWriter::writePI — PI (処理命令) 書き込む

説明

オブジェクト指向型

XMLWriter
bool **writePI** (string $target , string $content)

手続き型

bool **xmlwriter_write_pi** (resource $xmlwriter , string $target , string $content)

処理命令を書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

target

処理命令の対象。

content

処理命令の内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::startPI
- XMLWriter::endPI

XMLWriter::writeRaw

(No version information available, might be only in CVS)

XMLWriter::writeRaw — 生の XML テキストを書き込む

説明

オブジェクト指向型

XMLWriter
bool **writeRaw** (string $content)

手続き型

bool **xmlwriter_write_raw** (resource $xmlwriter , string $content)

生の xml テキストを書き込みます。

パラメータ

xmlwriter

手続き型のコールでのみ使用します。 変更される XMLWriter resource です。 このリソースは、xmlwriter_open_uri() あるいは xmlwriter_open_memory() のコールによって取得したものです。

content

書き込むテキスト文字列。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XMLWriter::text

目次

- XMLWriter::startDTDEntity — DTD エンティティを開始する
- XMLWriter::startDTD — DTD の開始タグを作成する
- XMLWriter::startElementNS — 名前空間つき要素の開始タグを作成する
- XMLWriter::startElement — 要素の開始タグを作成する
- XMLWriter::startPI — PI（処理命令）の開始タグを作成する
- XMLWriter::text — テキストを書き込む
- XMLWriter::writeAttributeNS — 名前空間つき属性全体を書き込む
- XMLWriter::writeAttribute — 属性全体を書き込む
- XMLWriter::writeCData — CDATA タグ全体を書き込む
- XMLWriter::writeComment — コメントタグ全体を書き込む
- XMLWriter::writeDTDAttlist — DTD 属性リストタグ全体を書き込む
- XMLWriter::writeDTDElement — DTD 要素タグ全体を書き込む
- XMLWriter::writeDTDEntity — DTD エンティティタグ全体を書き込む
- XMLWriter::writeDTD — DTD タグ全体を書き込む
- XMLWriter::writeElementNS — 名前空間つき要素タグ全体を書き込む
- XMLWriter::writeElement — 要素タグ全体を書き込む
- XMLWriter::writePI — PI（処理命令）書き込む
- XMLWriter::writeRaw — 生の XML テキストを書き込む

XSL 関数

導入

XSL エクステンションは、XSL の標準規格を実装したもので、 libxslt ライブラリを用いて » XSLT 変換 を行います。

要件

このテクステンションは » http://xmlsoft.org/XSLT/ にある libxslt を使用します。 libxslt バージョン 1.1.0 以降が必要です。

インストール手順

PHP 5には、デフォルトでXSLエクステンションが含まれており、 configureに引数--with-xsl[=DIR] を追加することにより有効にすることができます。 DIR は libxslt をインストールしたディレクトリです。

定義済みクラス

XSLTProcessor

コンストラクタ

- XSLTProcessor::__construct - 新規 XSLTProcessor オブジェクトを生成する

メソッド

- XSLTProcessor::getParameter - パラメータの値を取得する
- XSLTProcessor::hasExsltSupport - PHP が EXSLT をサポートしているかどうかを判定する
- XSLTProcessor::importStylesheet - スタイルシートを取り込む
- XSLTProcessor::registerPHPFunctions - XSLT 関数として PHP 関数を使用できるようにする
- XSLTProcessor::removeParameter - パラメータを取り除く
- XSLTProcessor::setParameter - パラメータの値を設定する
- XSLTProcessor::transformToDoc - DOMDocument に変換する
- XSLTProcessor::transformToURI - URI に変換する
- XSLTProcessor::transformToXML - XML に変換する

例

このリファレンスにある多くの例は、XML ファイルと XSL ファイルの両方を必要とします。 例では、以下の内容を含む collection.xml と collection.xsl を使用します。

Example#1 collection.xml

```
<collection>
  <cd>
    <title>Fight for your mind</title>
    <artist>Ben Harper</artist>
    <year>1995</year>
  </cd>
  <cd>
    <title>Electric Ladyland</title>
    <artist>Jimi Hendrix</artist>
```

```
   <year>1997</year>
  </cd>
</collection>
```

Example#2 collection.xsl

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="owner" select="'Nicolas Eliaszewicz'"/>
  <xsl:output method="html" encoding="iso-8859-1" indent="no"/>
  <xsl:template match="collection">
   Hey! Welcome to <xsl:value-of select="$owner"/>'s sweet CD collection!
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="cd">
    <h1><xsl:value-of select="title"/></h1>
    <h2>by <xsl:value-of select="artist"/> - <xsl:value-of select="year"/></h2>
   <hr />
  </xsl:template>
</xsl:stylesheet>
```

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

XSL_CLONE_AUTO (integer)
XSL_CLONE_NEVER (integer)
XSL_CLONE_ALWAYS (integer)
LIBXSLT_VERSION (integer)
libxslt のバージョンを 10117 のように表します。PHP 5.1.2 以降で使用可能です。
LIBXSLT_DOTTED_VERSION (string)
libxslt のバージョンを 1.1.17 のように表します。PHP 5.1.2 以降で使用可能です。
LIBEXSLT_VERSION (integer)
libexslt のバージョンを 813 のように表します。PHP 5.1.2 以降で使用可能です。
LIBEXSLT_DOTTED_VERSION (string)
libexslt のバージョンを 1.1.17 のように表します。PHP 5.1.2 以降で使用可能です。

XSLTProcessor::__construct

(No version information available, might be only in CVS)

XSLTProcessor::__construct — 新規 XSLTProcessor オブジェクトを生成する

説明

XSLTProcessor
__construct (void)

新規 XSLTProcessor オブジェクトを生成します。

例

Example#1 XSLTProcessor の生成

```
<?php

$doc = new DOMDocument();
$xsl = new XSLTProcessor();

$doc->load($xsl_filename);
$xsl->importStyleSheet($doc);

$doc->load($xml_filename);
echo $xsl->transformToXML($doc);

?>
```

XSLTProcessor::getParameter

(No version information available, might be only in CVS)

XSLTProcessor::getParameter — パラメータの値を取得する

説明

XSLTProcessor
string **getParameter** (string $namespaceURI , string $localName)

パラメータがあらかじめ XSLTProcessor::setParameter で設定されている場合、 それを取得します。

パラメータ

namespaceURI

XSLT パラメータの名前空間 URI を指定します。

localName

XSLT パラメータのローカル名を指定します。

返り値

パラメータの値、あるいは設定されていない場合は **NULL** 。

参考

- XSLTProcessor::setParameter
- XSLTProcessor::removeParameter

XSLTProcessor::hasExsltSupport

(No version information available, might be only in CVS)

XSLTProcessor::hasExsltSupport — PHP が EXSLT をサポートしているかどうかを判定する

説明

XSLTProcessor
bool **hasExsltSupport** (void)

このメソッドは PHP が » EXSLT library と共にビルドされたかどうかを判定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 EXSLT サポートの判定

```
<?php

$proc = new XSLTProcessor;
if (!$proc->hasExsltSupport()) {
    die('EXSLT support not available');
}

// EXSLT 処理がここに ..

?>
```

XSLTProcessor::importStylesheet

(No version information available, might be only in CVS)

XSLTProcessor::importStylesheet — スタイルシートを取り込む

説明

XSLTProcessor
void **importStylesheet** (DOMDocument $stylesheet)

このメソッドは XSLTProcessor に変換のためのスタイルシートを取り込みます。

パラメータ

stylesheet

取り込まれるスタイルシートを DOMDocument オブジェクトとして指定します。

返り値

値を返しません。

XSLTProcessor::registerPHPFunctions

(No version information available, might be only in CVS)

XSLTProcessor::registerPHPFunctions — PHP 関数を XSLT 関数として利用できるようにする

説明

XSLTProcessor
void **registerPHPFunctions** ([mixed $restrict])

このメソッドは、PHP 関数を XSL スタイルシートでの XSLT 関数として利用できるようにします。

パラメータ

restrict

このパラメータは、XSLT からコールされる信頼できる関数のみを許可します。

このパラメータには文字列（関数名）あるいは関数の配列のいずれかを指定します。

返り値

値を返しません。

例

Example#1 スタイルシートからの単純な PHP 関数コール

```
<?php
$xml = <<<EOB
<allusers>
  <user>
    <uid>bob</uid>
  </user>
  <user>
    <uid>joe</uid>
  </user>
</allusers>
EOB;
$xsl = <<<EOB
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:php="http://php.net/xsl">
<xsl:output method="html" encoding="utf-8" indent="yes"/>
  <xsl:template match="allusers">
    <html><body>
      <h2>Users</h2>
      <table>
       <xsl:for-each select="user">
        <tr><td>
          <xsl:value-of
                select="php:function('ucfirst',string(uid))"/>
        </td></tr>
       </xsl:for-each>
      </table>
    </body></html>
  </xsl:template>
</xsl:stylesheet>
EOB;
$xmldoc = DOMDocument::loadXML($xml);
$xsldoc = DOMDocument::loadXML($xsl);

$proc = new XSLTProcessor();
$proc->registerPHPFunctions();
$proc->importStyleSheet($xsldoc);
echo $proc->transformToXML($xmldoc);
?>
```

変更履歴

バージョン	説明
5.1.0	restrict パラメータが追加されました。

XSLTProcessor::removeParameter

(No version information available, might be only in CVS)

XSLTProcessor::removeParameter — パラメータを削除する

説明

XSLTProcessor
bool **removeParameter** (string $namespaceURI , string $localName)

もしパラメータが設定されていれば削除します。これにより、 プロセッサはスタイルシートで指定されるようなパラメータに デフォルト値を使用します。

パラメータ

namespaceURI

XSLT パラメータの名前空間 URI を指定します。

localName

XSLT パラメータのローカル名を指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- XSLTProcessor::setParameter

- XSLTProcessor::getParameter

XSLTProcessor::setParameter

(No version information available, might be only in CVS)

XSLTProcessor::setParameter — パラメータの値を設定する

説明

```
XSLTProcessor
bool setParameter ( string $namespace , string $name , string $value )
XSLTProcessor
bool setParameter ( string $namespace , array $options )
```

XSLTProcessor を使った変換のための 1 つあるいは多くのパラメータの値を設定します。 もしパラメータがスタイルシートに存在しない場合、無視されます。

パラメータ

namespace

XSLT パラメータの名前空間 URI を指定します。

name

XSLT パラメータのローカル名を指定します。パラメータ名に相当する文字列、 あるいは name => value の組の配列を指定可能です。

value

XSLT パラメータの新しい値を指定します。

options

名前 => 値 の組の配列を指定します。 この書式は PHP5.1.0 から利用可能です。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

例

Example#1 返還前に所有者を変更する

```php
<?php

$collections = array(
    'Marc Rutkowski' => 'marc',
    'Olivier Parmentier' => 'olivier'
);

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// 変換の設定を行う
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // attach the xsl rules

foreach ($collections as $name => $file) {
    // XML ソースをロードするload the XML source
    $xml = new DOMDocument;
    $xml->load('collection_' . $file . '.xml');

    $proc->setParameter('', 'owner', $name);
    $proc->transformToURI($xml, 'file:///tmp/' . $file . '.html');
}

?>
```

参考

- XSLTProcessor::getParameter
- XSLTProcessor::removeParameter

XSLTProcessor::transformToDoc

(No version information available, might be only in CVS)

XSLTProcessor::transformToDoc — DOMDocument に変換する

説明

```
XSLTProcessor
DOMDocument transformToDoc ( DOMNode $doc )
```

XSLTProcessor::importStylesheet メソッドで与えられたスタイルシートを適用し、 ソースノードを DOMDocument に変換します。

パラメータ

doc

変換されるノードを指定します。

返り値

結果の DOMDocument 。 エラーが発生した場合は **FALSE** 。

例

Example#1 DOMDocument への変換

```
<?php

// XML ソースをロードする
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// 変換の設定を行う
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // XSL ルールを適用する

echo trim($proc->transformToDoc($xml)->firstChild->wholeText);

?>
```

上の例の出力は以下となります。

```
Hey! Welcome to Nicolas Eliaszewicz's sweet CD collection!
```

参考

- XSLTProcessor::transformToURI
- XSLTProcessor::transformToXML

XSLTProcessor::transformToURI

(No version information available, might be only in CVS)

XSLTProcessor::transformToURI — URI に変換する

説明

XSLTProcessor
int **transformToURI** (DOMDocument $doc , string $uri)

XSLTProcessor::importStylesheet メソッドで与えられたスタイルシートを適用し、 ソースノードを URI に変換します。

パラメータ

doc

変換される文章を指定します。 The transformed document.

uri

返り値

書き込まれたバイト数。エラーが発生した場合は **FALSE** 。

例

Example#1 HTML ファイルへの変換

```
<?php

// XML ソースをロードする
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// 変換の設定を行う
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // XSL ルールを適用する

$proc->transformToURI($xml, 'file:///tmp/out.html');

?>
```

参考

- XSLTProcessor::transformToDoc
- XSLTProcessor::transformToXML

XSLTProcessor::transformToXML

(No version information available, might be only in CVS)

XSLTProcessor::transformToXML — XML に変換する

説明

XSLTProcessor
string **transformToXML** (DOMDocument $doc)

XSLTProcessor::importStylesheet メソッドで与えられたスタイルシートを適用し、 ソースノードを文字列に変換します。

パラメータ

doc

変換されるノードを指定します。

返り値

変換結果の文字列。エラーが発生した場合は **FALSE** 。

例

Example#1 文字列への変換

```
<?php

// XML ソースをロードする
$xml = new DOMDocument;
$xml->load('collection.xml');

$xsl = new DOMDocument;
$xsl->load('collection.xsl');

// 変換の設定を行う
$proc = new XSLTProcessor;
$proc->importStyleSheet($xsl); // XSL ルールを適用する

echo $proc->transformToXML($xml);

?>
```

上の例の出力は以下となります。

```
Hey! Welcome to Nicolas Eliaszewicz's sweet CD collection!

<h1>Fight for your mind</h1><h2>by Ben Harper - 1995</h2><hr>
<h1>Electric Ladyland</h1><h2>by Jimi Hendrix - 1997</h2><hr>
```

参考

- XSLTProcessor::transformToDoc
- XSLTProcessor::transformToURI

目次

XSLT 関数

導入

このPHP拡張モジュールは、APIに独立なXSLT変換を提供します。現在、 この拡張モジュールは、Ginger AllianceによるSablotronライブラリのみをサポートします。Xalanライブラリまたはlibxsltライブラリのよう な他のライブラリのサポートも計画されています。

XSLT (Extensible Stylesheet Language (XSL) Transformations) は、XMLドキュメントを他のXMLドキュメントに変換 する言語です。XSLTは、World Wide Web コンソーシアム (W3C) により 標準化されています。XSLTに関する情報と関連技術については、 » http://www.w3.org/TR/xslt から得ることができま す。

> **注意**: この拡張は、PHP 4.1.0 より前のバージョンPHPで配布されていた sablotron拡張モジュールとは異なっており、現在、PHP 4.1.0 の新しい XSLT拡張モジュールのみがサポートされています。古い拡張モジュー ルのサポートが必要な場合は、PHPのメーリングリストに質問してください。
>
> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。PHP 5.0.0.
>
> **注意**: PHP5でxsltサポートを必要とする場合、 XSLエクステンションを使用することができます。

要件

このエクステンションは、Sablotonおよび expat を使用します。これらは、共に » http://www.gingerall.org/sablotron.htmlから得ることが可能です。 実行バイナリがソースコードと同様に配布されています。

インストール手順

Unix では、オプション --enable-xslt --with-xslt-sablotを指定 して**configure** を実行します。ライブラリ Sablotron をコンパイラが見付けられる 場所にインストールする必要があります。

Sablotronにリンクされているのと 同じライブラリとPHPがリンクされていることを確認してください。 設定オプション --with-expat-dir=DIR --with-iconv-dir=DIR は、これらの指定を行う際に有用です。 サポートに質問する際に、常にこれらのディレクティブや、他のバージョン のライブラリが使用するシステムのどこかにインストールされて いるかどうかについて述べるようにしてください。 普通は、全てのバージョン番号を知らせてください。

警告

Sablot ライブラリは -lstdc++ にリンクしていることに 注意してください。そうでないとコンフィギュアに失敗したり、 PHP が起動しない、あるいはロードされません。

> **注意**: **JavaScript E-XSLT サポート** SablotronをJavaScriptサポートを付けてコ ンパイルした場合、オプション --with-sablot-js=DIRを指定する必 要があります。
>
> **注意**: **Win32 ユーザへの注意** この拡張モジュールを動作させるには、 Windows システムの PATH が通った場所に DLL ファイルが存在する必要があります。 FAQ の "Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?" で、その方法を説明しています。 DLL ファイルを PHP のフォルダから Windows のシステムディレクトリにコピーしても動作します (システムディレクトリは、デフォルトで PATH に含まれるからです) が、これは推奨しません。 この拡張モジュールを使用するには、以下のファイルが PATH の通った場所にある必要があります。 sablot.dll、 expat.dll および iconv.dll
> PHP <= 4.2.0 では、iconv.dll は不要です。

実行時設定

設定ディレクティブは定義されていません。

リソース型

この拡張モジュールでは XSLT プロセッサリソースを定義しています。これは xslt_create() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

XSLT_OPT_SILENT (integer)
ログ出力とエラー報告を全てドロップします。 将来追加される全てのバックエンドについて共通のオプションです。
XSLT_SABOPT_PARSE_PUBLIC_ENTITIES (integer)

Sablotronがpublicエンティティを パースするようにします。デフォルトではオフになっています。

XSLT_SABOPT_DISABLE_ADDING_META (integer)

HTML出力のときに "Content-Type" メタタグを追加しません。 Sablotronのコンパイル時にデフォルト値が セットされます。

XSLT_SABOPT_DISABLE_STRIPPING (integer)
空白の除去を抑制します。(データファイルに対してのみ)
XSLT_SABOPT_IGNORE_DOC_NOT_FOUND (integer)
解決できない文章 (document() 関数) をリテラルでないものと見なします。
XSLT_SABOPT_FILES_TO_HANDLER (integer)
XSLT_ERR_UNSUPPORTED_SCHEME (integer)
スキームハンドラ のエラーをコードで返します。

---

xslt_backend_info

(PHP 4 >= 4.3.0)

xslt_backend_info — バックエンドのコンパイル設定についての情報を返す

説明

string **xslt_backend_info** (void)

xslt_backend_info() は、バックエンドのコンパイル設定についての情報を取得します。

返り値

バックエンドのコンパイル設定についての情報を文字列で返すか、 情報が利用できないときエラー文字列を返します。

参考

- xslt_backend_name()
- xslt_backend_version()

---

xslt_backend_name

(PHP 4 >= 4.3.0)

xslt_backend_name — バックエンドの名前を返す

説明

string **xslt_backend_name** (void)

xslt_backend_name() は、バックエンドの名前を取得します。

返り値

Sablotron を返します。

例

Example#1 xslt_backend_name() の例

```
<?php
echo xslt_backend_name(); // Sablotron
?>
```

参考

- xslt_backend_info()
- xslt_backend_version()

---

xslt_backend_version

(PHP 4 >= 4.3.0)

xslt_backend_version — Sablotron のバージョン番号を返す

説明

string **xslt_backend_version** (void)

xslt_backend_version() は、 Sablotron のバージョン番号を取得します。

返り値

バージョン番号を返します。失敗した場合は **FALSE** を返します。

例

Example#1 xslt_backend_version() の例

```
<?php
echo xslt_backend_version(); // たとえば 0.98
?>
```

参考

- xslt_backend_info()
- xslt_backend_name()

---

xslt_create

(PHP 4 >= 4.0.3)

xslt_create — 新規の XSLT プロセッサを作成する

説明

resource ***xslt_create*** *(void)*

新規に XSLT プロセッサリソースを作成し、返します。このリソースは、 他の XSLT 関数による処理で使用されます。

返り値

成功した場合に XSLT プロセッサリンク ID、 エラー時に **FALSE** を返します。

例

Example#1 xslt_create() の例

```
<?php
function xml2html($xmldata, $xsl)
{
    /* $xmldata -> あなたの XML */
     /* $xsl -> XSLT ファイル */

    $path = 'include';
    $arguments = array('/_xml' => $xmldata);
    $xsltproc = xslt_create();
    xslt_set_encoding($xsltproc, 'ISO-8859-1');
    $html =
        xslt_process($xsltproc, 'arg:/_xml', "$path/$xsl", NULL, $arguments);

     if (empty($html)) {
       die('XSLT processing error: '. xslt_error($xsltproc));
    }
    xslt_free($xsltproc);
     return $html;
}
?>
```

参考

- xslt_free()

---

xslt_errno

(PHP 4 >= 4.0.3)

xslt_errno — エラー番号を返す

説明

int ***xslt_errno*** *(resource $xh)*

XSLT プロセッサで発生した直近のエラーを記述するエラーコードを返します。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

返り値

エラーコードを表す整数値を返します。

参考

- xslt_error()

---

xslt_error

(PHP 4 >= 4.0.3)

xslt_error — エラー文字列を返す

説明

string ***xslt_error*** *(resource $xh)*

指定した XSLT プロセッサで発生した直近のエラーを説明する文字列を返します。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

返り値

エラーメッセージを表す文字列を返します。

例

Example#1 関数 xslt_error() および xslt_errno() によるエラー処理

```
<?php

$xh = xslt_create();
$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
if (!$result) {
    die(sprintf("Cannot process XSLT document [%d]: %s",
                xslt_errno($xh), xslt_error($xh)));
}

echo $result;

xslt_free($xh);
?>
```

参考

- xslt_errno()

---

xslt_free

(PHP 4 >= 4.0.3)

xslt_free — XSLT プロセッサを開放する

説明

void **xslt_free** (resource $xh)

ハンドルで指定した XSLT プロセッサを開放します。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

返り値

値を返しません。

参考

- xslt_create()

---

xslt_getopt

(PHP 4 >= 4.3.0)

xslt_getopt — xsl プロセッサのオプションを取得する

説明

int **xslt_getopt** (resource $processor)

xslt_getopt() は、与えられた processor のオプションを返します。

パラメータ

processor

xslt_create() で作成した XSLT プロセッサリンク ID。

返り値

オプションを返します。これは、定数 XSLT_SABOPT_XXX のビットマスクとなります。

参考

- xslt_setopt()

---

xslt_process

(PHP 4 >= 4.0.3)

xslt_process — XSLT による変換を行う

説明

mixed **xslt_process** (resource $xh , string $xmlcontainer , string $xslcontainer [, string $resultcontainer [, array $arguments [, array $parameters]]])

xslt_process()関数は、XSLT 拡張モジュールの中心となる関数です。 ほとんど全ての型の入力ソース（コンテナ）を用いて XSLT 変換を実行可能です。 これを実現しているのが、引数バッファです。 引数バッファとは、Sablotron XSLTプロセッサ（現在、この拡張モジュールがサポートする唯一の XSLT プロセッサ）から得た概念です。 入力コンテナは処理する文章を '含んでいる' ファイル名がデフォルトです。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

xmlcontainer

XML ファイルへのパス、あるいは XML 引数用のプレースホルダ。

xslcontainer

XSL ファイルへのパス、あるいは XML 引数用のプレースホルダ。

resultcontainer

結果コンテナは、変換された文章のためのファイル名が デフォルトです。もし結果コンテナが指定されていない場合 - 例えば **NULL** - 、結果が返されます。

arguments

XML や XSLT のファイル名を **xslt_process()** 関数に指定するかわりに、"引数プレースホルダ" を使用することもできます。これは、配列 arguments に指定した内容で置き換えられます。

parameters

任意のトップレベルパラメータの配列。これが XSLT ドキュメントに渡されます。XSL ファイル内でこれらのパラメータにアクセスするには <xsl:param name="parameter_name"> を使用します。 パラメータは UTF-8 でエンコードする必要があります。 その値は文字列として Sablotron プロセッサが処理します。つまり、XSLT ドキュメントのパラメータには ノードセットを渡すことはできないということです。

また、arguments 配列を通じてコンテナを設定することができます（以下参照）。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 結果コンテナを指定していない場合は結果を返します。

変更履歴

バージョン	説明
4.0.6	この関数は xmlcontainer もしくは xslcontainer で XML 文字列を受け付けなくなりました。 XML を含む文字列を渡すと、0.95とそれ以降の Sablotron バージョンでセグメンテーションフォルトを引き起こします。

例

xslt_process()関数で変換する最も簡単な方法は、 XMLファイルをXSLTファイルで変換し、結果を新しいXMLドキュメント(ま たはHTMLドキュメント)を含む3番目のファイルに出力することです。 これをsablotronを行うのは、かなり簡単です。

Example#1 XMLファイルとXSLファイルを新規XMLファイルに変換するために xslt_process()を使用する

```
<?php

// 新規 XSLT プロセッサを割り当てる
$xh = xslt_create();

// 文章を処理する
if (xslt_process($xh, 'sample.xml', 'sample.xsl', 'result.xml')) {
    echo "SUCCESS, sample.xml was transformed by sample.xsl into result.xml";
    echo ", result.xml has the following contents¥n<br />¥n";
    echo "<pre>¥n";
    readfile('result.xml');
    echo "</pre>¥n";
} else {
    echo "Sorry, sample.xml could not be transformed by sample.xsl into";
    echo "  result.xml the reason is that " . xslt_error($xh) . " and the ";
    echo "error code is " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

この機能は優れていますが、特にWeb環境では、結果を直接出力したい場 合があります。そこで、**xslt_process()**の3番目の 引数を省略した場合(またはその引数にNULL値を指定した場合)、ファイ ルに書き込む替わりに自動的にXSLT変換後の出力を返します。

Example#2 XMLファイルとXSLファイルを結果XMLデータを含む変数に変換するため にxslt_process()を使用する

```
<?php

// 新規 XSLT プロセッサを割り当てる
$xh = xslt_create();
```

```
// 文章を処理し、変数 $result に結果を返す
$result = xslt_process($xh, 'sample.xml', 'sample.xsl');
if ($result) {
    echo "SUCCESS, sample.xml was transformed by sample.xsl into the ¥$result";
    echo " variable, the ¥$result variable has the following contents¥n<br />¥n";
    echo "<pre>¥n";
    echo $result;
    echo "</pre>¥n";
} else {
    echo "Sorry, sample.xml could not be transformed by sample.xsl into";
    echo "  the ¥$result variable the reason is that " . xslt_error($xh);
    echo " and the error code is " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

上の二つのケースは、XSLT変換の最も簡単な場合です。これは、多くの 場合には通用しますが、時々、データベースまたはソケットのような外 部ソースからXMLとXSLTコードを取得する場合があります。このような場 合、XMLまたはXSLTデータを変数に有することになります。 実用アプリケーションでは、これらをファイルにダンプする際のオーバー ヘッドは大きいと言えます。このような場合こそ、XSLT "argument" 構文が役に立ちます。 **xslt_process()**関数のXMLおよびXSLT引数としてファ イルの替わりに引数配列(**xslt_process()**関数の5番 目のパラメータ)で指定した値に置換される"argument place holders"を指定することが可能です。 以下にファイルを全く使用せずにXMLおよびXSLTを結果変数に処理する例 を示します。

Example#3 XMLデータを含む変数とXSLTデータを含む変数をXMLデータ出力結果を 含む変数に変換するためにxslt_process()を使用 する

```
<?php
// $xml と $xsl は XML データと XSL データを含む

$arguments = array(
    '/_xml' => $xml,
    '/_xsl' => $xsl
);

// 新規 XSLT プロセッサを割り当てる
$xh = xslt_create();

// 文章を処理する
$result = xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, $arguments);
if ($result) {
    echo "SUCCESS, sample.xml was transformed by sample.xsl into the ¥$result";
    echo " variable, the ¥$result variable has the following contents¥n<br />¥n";
    echo "<pre>¥n";
    echo $result;
    echo "</pre>¥n";
} else {
    echo "Sorry, sample.xml could not be transformed by sample.xsl into";
    echo "  the ¥$result variable the reason is that " . xslt_error($xh);
    echo " and the error code is " . xslt_errno($xh);
}
xslt_free($xh);
?>
```

Example#4 PHP 変数を XSL ファイルに渡す

```
<?php

// XML 文字列
$xml = '<?xml version="1.0"?>
<para>
 change me
</para>';

// XSL 文字列
$xsl = '
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1" indent="no"
  omit-xml-declaration="yes"   media-type="text/html"/>
  <xsl:param name="myvar"/>
  <xsl:param name="mynode"/>
  <xsl:template match="/">
My PHP variable : <xsl:value-of select="$myvar"/><br />
My node set : <xsl:value-of select="$mynode"/>
  </xsl:template>
</xsl:stylesheet>';


$xh = xslt_create();

// 2番目のパラメータは文字列として解釈される
$parameters = array (
  'myvar' => 'test',
  'mynode' => '<foo>bar</foo>'
);

$arguments = array (
  '/_xml' => $xml,
  '/_xsl' => $xsl
);

echo xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, $arguments, $parameters);

?>
```

上の例の出力は以下となります。

```
My PHP variable : test<br>
My node set : &lt;foo&gt;bar&lt;/foo&gt;
```

注意

注意: Windows を使用している場合、 file:// がパスの前に必要であることに注意してください。

---

xslt_set_base

(PHP 4 >= 4.0.5)

xslt_set_base — 全ての XSLT 変換用の基準 URI を設定する

説明

void **xslt_set_base** (resource $xh , string $uri)

全てのXSLT変換の基準URIを設定します。基準URIは、外部リソースにア クセスするdocument()及び他のコマンドを解決する際にXpath命令に関して使用されます。 また、<xsl:include> と <xsl:import> 要素のための URI を解決するために使用されます。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

uri

使用するベース URI。

返り値

値を返しません。

変更履歴

バージョン	説明
4.3.0	4.3 以降、デフォルトのベース URI は実行しているスクリプトのディレクトリです。 実際には **__FILE__** 定数の値のディレクトリ名です。 4.3 以前では、デフォルトのベース URI はほとんど予想できません。

注意

注意: Windows を使用している場合、 file:// がパスの前に必要であることに注意してください。

---

xslt_set_encoding

(PHP 4 >= 4.0.5)

xslt_set_encoding — XML ドキュメントをパースするエンコーディングを設定する

説明

void **xslt_set_encoding** (resource $xh , string $encoding)

XSLT 変換の出力エンコーディングを設定します。 Sablotron バックエンドを使用する際、 このオプションは、Sablotron にエンコーディングサポートを付けてコンパイルした場合のみ利用可能です。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

encoding

出力エンコーディング。たとえば iso-8859-1 など。

返り値

値を返しません。

---

xslt_set_error_handler

(PHP 4 >= 4.0.4)

xslt_set_error_handler — XSLT プロセッサ用のエラーハンドラを設定する

説明

```
void xslt_set_error_handler ( resource $xh , mixed $handler )
```

xh で指定した XSLT プロセッサ用にエラーハンドラ関数を設定します。 この関数は、XSLT 変換においてエラーが発生する度にコールされます（この関数は通知のためにもコールされます）。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

handler

ユーザ関数は 4 つのパラメータを受け取る必要があります: XSLT プロセッサ、エラーレベル、エラーコードとメッセージの配列です。 その関数は次のように表すことができます。

```
error_handler ( resource $xh , int $error_level , int $error_code , array $messages )
```

返り値

値を返しません。

例

Example#1 xslt_set_error_handler() の例

```
<?php

// 独自の XSLT エラーハンドラ
function xslt_error_handler($handler, $errno, $level, $info)
{
  // 差し当たり、引数を見るだけにする
  var_dump(func_get_args());
}

// XML の内容 :
$xml='<?xml version="1.0"?>
<para>
 oops, I misspelled the closing tag
</pata>';

// XSL の内容 :
$xsl='<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
     <strong><xsl:value-of select="para"/></strong>
</xsl:template>
</xsl:stylesheet>';

$xh = xslt_create();

xslt_set_error_handler($xh, "xslt_error_handler");

echo xslt_process($xh, 'arg:/_xml', 'arg:/_xsl',
                  NULL, array("/_xml" => $xml, "/_xsl" => $xsl));

?>
```

上の例の出力は、たとえば以下のようになります。

```
array(4) {
   [0]=>
   resource(1) of type (XSLT Processor)
   [1]=>
   int(3)
   [2]=>
   int(0)
   [3]=>
   array(6) {
      ["msgtype"]=>
      string(5) "error"
      ["code"]=>
      string(1) "2"
      ["module"]=>
      string(9) "Sablotron"
      ["URI"]=>
      string(9) "arg:/_xml"
      ["line"]=>
      string(1) "4"
      ["msg"]=>
      string(34) "XML parser error 7: mismatched tag"
   }
}
```

参考

- オブジェクトメソッドをハンドラとして使用したい場合は xslt_set_object()

---

xslt_set_log

(PHP 4 >= 4.0.6)

xslt_set_log — ログメッセージを書き込むためのログファイルを設定する

説明

void **xslt_set_log** (resource $xh [, mixed $log])

この関数により、XSLT ログメッセージを書き込むためのファイルを設定することができます。 XSLT ログメッセージはエラーメッセージとは異なります。 実際のエラーメッセージではなく、XSLT プロセッサの状態に関連するメッセージです。 何かかおかしいとき XSLT のデバッグを行う助けになります。

デフォルトではロギングは無効です。有効にするには、 まずロギングを有効にするためのブール型のパラメータを用いて **xslt_set_log()** をコールする必要があります。 もしデバッグのためにログファイルを設定したい場合、 ファイル名を含む文字列を渡す必要があります。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

log

このパラメータには、ロギングをオン・オフするトグルとしてのブール型の値、 あるいはエラーもロギングするためのログファイルを含む文字列を指定します。

返り値

値を返しません。

注意

注意: Windows を使用している場合、 file:// がパスの前に必要であることに注意してください。

例

Example#1 XSLT のロギング機能の使用法

```
<?php

$xh = xslt_create();
xslt_set_log($xh, true);
xslt_set_log($xh, getcwd() . '/myfile.log');

$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
echo $result;

xslt_free($xh);
?>
```

---

xslt_set_object

(PHP 4 >= 4.3.0)

xslt_set_object — コールバック関数を解決するためのオブジェクトを設定する

説明

bool **xslt_set_object** (resource $processor , object &$obj)

この関数は、全てのコールバック関数を解決するため、 processor に object の内部情報を利用できるようにします。

コールバック関数の宣言は **xml_set_sax_handlers()**, xslt_set_scheme_handlers(), xslt_set_error_handler() で行い、それが object のメソッドとみなされます。

パラメータ

processor

xslt_create() で作成した XSLT プロセッサリンク ID。

obj

オブジェクト。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 独自のエラーハンドラとしてメソッドを使用する

```
<?php

class my_xslt_processor {

    var $_xh; // XSLT プロセッサ

    function my_xslt_processor()
```

```
    {
        $this->_xh = xslt_create();

       // $this オブジェクトをコールバックリゾルバにする
        xslt_set_object($this->_xh, $this);

       // エラーを処理させる
        xslt_set_error_handler($this->_xh, "my_xslt_error_handler");
    }

    function my_xslt_error_handler($handler, $errno, $level, $info)
    {
       // 差し当たり、引数を見るだけ
        var_dump(func_get_args());
    }
}

?>
```

---

xslt_set_sax_handler

(PHP 4 >= 4.0.3)

xslt_set_sax_handler — XSLT プロセッサに SAX ハンドラを設定する

説明

void **xslt_set_sax_handler** (resource $xh , array $handlers)

xh で指定したリソースハンドルに SAX ハンドラを設定します。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

handlers

SAXハンドラは、フォーマットの2次元配列とする必要があります（全てのトップレベル要素はオプションです）。

```
array(
[document] =>
     array(
          start document handler,
          end document handler
     ),
[element] =>
     array(
          start element handler,
          end element handler
     ),
[namespace] =>
     array(
          start namespace handler,
          end namespace handler
     ),
[comment] => comment handler,
[pi] => processing instruction handler,
[character] => character data handler
)
```

返り値

値を返しません。

---

xslt_set_sax_handlers

(PHP 4 >= 4.0.6)

xslt_set_sax_handlers — XML ドキュメントを処理する際にコールされる SAX ハンドラを設定する

説明

void **xslt_set_sax_handlers** (resource $processor , array $handlers)

xslt_set_sax_handlers() は、与えられた XSLT processor リソースに文章のための SAX ハンドラ handlers を設定します。

xslt_set_sax_handlers() を使用することは、 xslt_process() を使用した変換の結果を xml_parse() のような SAX パーサで実行することと比べて あまり難しく見えません。

パラメータ

processor

xslt_create() で作成した XSLT プロセッサリンク ID。

handlers

handlers は、 次のフォーマット形式の配列でなければなりません。

```
<?php

$handlers = array(

  "document" => array(
    "start_doc",
    "end_doc"),

  "element"  => array(
    "start_element",
    "end_element"),

  "namespace" => array(
    "start_namespace",
    "end_namespace"),

  "comment"   => "comment",

  "pi"        => "pi",

  "character" => "characters"

);
?>
```

ここでの関数は、スキーマハンドラ関数のシンタックスに従います。

> **注意**: 与えられた配列は全ての異なる SAX ハンドラ要素を含める必要はありません（ただし、それも可能です）。 しかし、上記にある "ハンドラ" => "関数" というフォーマットだけは 従う必要があります。

それぞれの独立した SAX ハンドラ関数は次のフォーマットです。

- **start_doc** (resource $processor)
- **end_doc** (resource $processor)
- **start_element** (resource $processor , string $name , array $attributes)
- **end_element** (resource $processor , string $name)
- **start_namespace** (resource $processor , string $prefix , string $uri)
- **end_namespace** (resource $processor , string $prefix)
- **comment** (resource $processor , string $contents)
- **pi** (resource $processor , string $target , string $contents)
- **characters** (resource $processor , string $contents)

返り値

値を返しません。

例

Example#1 xslt_set_sax_handlers() の例

```
<?php
// ohlesbeauxjours at yahoo dot fr より。
// これは、全ての <auteur> タグの内容に strtoupper() を適用し、
// 結果の XML ツリーを表示する、というシンプルな例です。

$xml='<?xml version="1.0"?>
<books>
  <book>
    <title>Mme Bovary</title>
    <author>Gustave Flaubert</author>
  </book>
  <book>
    <title>Mrs Dalloway</title>
    <author>Virginia Woolf</author>
  </book>
</books>';

$xsl='<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1" indent="no" omit-xml-declaration="yes"/>
<xsl:template match="/">
  <xsl:for-each select="books/book">
   <livre>
      <auteur><xsl:value-of select="author/text()"/></auteur>
   </livre>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>';

// ハンドラ :
function start_document()
{
  // 文章の読み込みを開始する
}

function end_document()
{
  // 文章の読み込みを終了する
}

function start_element($parser, $name, $attributes)
{
```

```
  global $result,$tag;
  $result .= "<". $name . ">";
  $tag = $name;
}

function end_element($parser, $name)
{
  global $result;
  $result .= "</" . $name . ">";
}

function characters($parser, $data)
{
  global $result,$tag;
  if ($tag == "auteur" ) {
    $data = strtoupper($data);
  }
  $result .= $data;
}

// 変換する :
$xh = xslt_create();
$handlers = array("document" => array("start_document","end_document"),
   "element" => array("start_element","end_element"),
   "character" => "characters");

xslt_set_sax_handlers($xh, $handlers);
xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, array("/_xml"=>$xml, "/_xsl"=>$xsl));
xslt_free($xh);
?>
```

オブジェクト内にハンドラを実装したい場合、 xslt_set_object() を使用することもできます。

Example#2 オブジェクト指向ハンドラ

```
<?php
// これは、前の例のオブジェクト指向バージョン
class data_sax_handler  {

  var $buffer, $tag, $attrs;

  var $_xh;

  function data_sax_handler($xml, $xsl)
  {
      // XSLT リソース
      $this->_xh = xslt_create();

      xslt_set_object($this->_xs, $this);

      // SAX ハンドラを設定する
      $handlers = array(
        "document" => array('start_document', 'end_document'),
        "element" => array('start_element', 'end_element'),
        "character" => 'characters'
      );

      xslt_set_sax_handlers($this->_xh, $handlers);

      xslt_process($this->_xh, 'arg:/_xml', 'arg:/_xsl', NULL, array("/_xml"=>$xml, "/_xsl"=>$xsl));
      xslt_free($this->_xh);


  }

  function start_document()
  {
       // 文章の読み込みを開始する
  }

  function end_document() {
       // 文章の読み込みを完了する
  }

  function start_element($parser, $name, $attributes) {
        $this->tag = $name;
        $this->buffer .= "<" . $name . ">";
        $this->attrs = $attributes;
  }

  function end_element($parser, $name)
  {
        $this->tag = '';
        $this->buffer .= "</" . $name . ">";
  }

  function characters($parser, $data)
  {
    if ($this->tag == 'auteur') {
         $data = strtoupper($data);
    }
    $this->buffer .= $data;
  }

  function get_buffer() {
    return $this->buffer;
  }

}

$exec = new data_sax_handler($xml, $xsl);
```

```
?>
```

両方のサンプルとも以下を出力します。

```
<livre>
     <auteur>GUSTAVE  FLAUBERT</auteur>
</livre>
<livre>
     <auteur>VIRGINIA  WOOLF</auteur>
</livre>
```

---

xslt_set_scheme_handler

(PHP 4 >= 4.0.5)

xslt_set_scheme_handler — XSLT プロセッサ用にスキーマハンドラを設定する

説明

void **xslt_set_scheme_handler** (resource $xh , array $handlers)

xh で指定したリソースハンドルにスキーマハンドラを設定します。

パラメータ

xh

xslt_create() で作成した XSLT プロセッサリンク ID。

handlers

スキーマハンドラは、フォーマットの配列とする必要があります (全ての要素はオプションです)。

```
array(
[get_all] => get all handler,
[open] => open handler,
[get] => get handler,
[put] => put handler,
[close] => close handler
)
```

返り値

値を返しません。

---

xslt_set_scheme_handlers

(PHP 4 >= 4.0.6)

xslt_set_scheme_handlers — XSLT プロセッサに関するスキーマハンドラを設定する

説明

void **xslt_set_scheme_handlers** (resource $processor , array $handlers)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

返り値

値を返しません。

---

xslt_setopt

(PHP 4 >= 4.3.0)

xslt_setopt — 与えられた xsl プロセッサにオプションを設定する

説明

mixed **xslt_setopt** (resource $processor , int $newmask)

xslt_setopt() は、与えられた processor に newmask で指定されるオプションを設定します。

パラメータ

processor

xslt_create() で作成した XSLT プロセッサリンク ID。

newmask

newmask は次の定数で定義されたビットマスクです。

- ***XSLT_SABOPT_PARSE_PUBLIC_ENTITIES*** - プロセッサにパブリックなエンティティをパースするよう伝える。 デフォルトはオフ。
- ***XSLT_SABOPT_DISABLE_ADDING_META*** - HTML 出力にメタタグ "Content-Type" を出力しない。 デフォルトはプロセッサのコンパイル時に設定される。
- ***XSLT_SABOPT_DISABLE_STRIPPING*** - Suppress the whitespace stripping (on data files only).
- ***XSLT_SABOPT_IGNORE_DOC_NOT_FOUND*** - 空白の除去を抑制する （データファイルの場合のみ） 。

返り値

以前の設定値が取得できた場合はその値、それ以外の場合は ***TRUE*** を返します。エラーが発生した場合は ***FALSE*** を返します。

例

Example#1 xslt_setopt() の例

```
<?php

$xh = xslt_create();

// Sablotron にパブリックなエンティティを処理するよう伝える
xslt_setopt($xh, XSLT_SABOPT_PARSE_PUBLIC_ENTITIES);

// 空白の除去もしてくれるよう問い合わせてみる
xslt_setopt($xh, xslt_getopt($xh) | XSLT_SABOPT_DISABLE_STRIPPING);

?>
```

参考

- xslt_getopt()

目次

YAZ 関数

導入

この拡張モジュールは、情報取得用の Z39.50 プロトコルを実装する YAZ ツールキットへの PHP インターフェースを 提供するものです。この拡張モジュールにより、Z39.50 ターゲット（サーバ） を並列に検索またはスキャンする Z39.50 オリジン（クライアント）を 容易に実装することが可能になります。

このモジュールは Z39.50 の複雑さを隠蔽するため、使用法がかなり容易に なります。PHP で利用可能な様々な RDB の API により提供されているものに 非常によく似た、ステートレスな持続的接続がサポートされています。 これは、セッションはステートレスですが、ユーザ間で共有されるということを 意味します。これにより多くの場合に接続および初期化手順が保存されます。

YAZ は、» http://www.indexdata.dk/yaz/ で取得可能です。この拡張モジュールに 関する新しい情報、スクリプトの例等を » http://www.indexdata.dk/phpyaz/ にて 参照可能です。

> **注意**: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.0.0.

インストール手順

YAZ (ANSI/NISO Z39.50 サポート) を取得し、インストールしてください。 » YAZ archive から、ソース形式あるいは さまざまな形式のビルド済みパッケージで YAZ が取得可能です。 Debian GNU/Linux、Suse Linux、FreeBSD のようなシステムでは、 ディストリビューションの一部として YAZ が組み込まれています。

PHP 4 シリーズでは、YAZ 拡張モジュールがバンドルされています(しかし、 YAZ 自体はバンドルされていません)。 オプション --with-yaz[=DIR] を 指定し、他の任意のモジュールとあわせて PHP をコンパイルしてください。

Example#1 Unix 上の PHP 4 での YAZ のコンパイル

```
gunzip -c php-4.4.X.tar.gz|tar xf -
gunzip -c yaz-2.1.8.tar.gz|tar xf -
cd yaz-2.1.8
./configure --prefix=/usr
make
sudo make install
cd ../php-4.4.X.
./configure --with-yaz=/usr/bin
make
sudo make install
```

PHP 5 では YAZ 拡張モジュールは » PECL にあり、共有オブジェクト/dll としてインストールされます。 もし pear がインストールされている場合、YAZ 拡張モジュールを ダウンロードして設定・インストールをする最も簡単な方法は **pecl** コマンドを使用することです。

Linux 上での YAZ のインストール

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/yaz

Windows システム上でのインストール

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。php_yaz.dll は yaz.dll に依存しています。 yaz.dll は、PHP サイトにある Win32 zip アーカイブに 含まれています。また、» YAZ WIN32 area にある Windows 版の YAZ の中にも含まれています。

警告

PHP 5.0.5 の Win32 zip アーカイブに含まれている yaz.dll はバージョンが古すぎます (バージョン 1.9.1 < 要求されるバージョン 2.0.13)。 そのため、» YAZ WIN32 インストール から得られる新しいバージョンの yaz.dll を使用してください。

Windows では、システムが yaz.dll ファイルを 見つけられるよう、PHP のディレクトリを PATH に追加することを 忘れないでください。

YAZ を共有モジュールとして使用する場合、Unix では php.ini の以下の行を 追加(あるいはコメント解除)してください。

```
extension=yaz.so
```

また、Windows では以下の行になります。

```
extension=php_yaz.dll
```

警告

IMAP、recode、 YAZ および Cyrus 拡張モジュールは、組み合わせて使用することはできません。これは、これらすべてが同一の内部シンボルを使用しているためです。

注意: 上で挙げた問題については、YAZ のバージョン 2.0 で解消されています。

実行時設定

php.ini の設定により動作が変化します。

YAZ 設定オプション

名前	デフォルト	変更の可否	変更履歴
yaz.max_links	"100"	PHP_INI_ALL	PHP 4.3.0 以降で使用可能です。PHP 5.0.0 で削除されました。
yaz.log_file	NULL	PHP_INI_ALL	PHP 4.3.0 以降で使用可能です。PHP 5.0.0 で削除されました。

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

リソース型

リソース型は定義されていません。

定義済み定数

定数は定義されていません。

例

PHP/YAZ はターゲット(Z-Associations)との接続を保持し続けます。 正の整数で特定の接続のIDを表します。

以下のスクリプトは、API の並列検索機能のデモです。引数を指定せずに コールした場合、この関数はクエリフォームを出力します。そうでない場合（引数を指定した場合）は、配列 host にあるターゲットを 検索します。

Example#2 YAZ による並列検索

```
<?php
$host=$_REQUEST[host];
$query=$_REQUEST[query];
$num_hosts = count($host);
```

```
if (empty($query) || count($host) == 0) {
    echo '<form method="get">
     <input type="checkbox"
      name="host[]" value="bagel.indexdata.dk/gils" />
         GILS test
      <input type="checkbox"
      name="host[]" value="localhost:9999/Default" />
         local test
      <input type="checkbox" checked="checked"
      name="host[]" value="z3950.loc.gov:7090/voyager" />
          Library of Congress
    <br />
    RPN Query:
     <input type="text" size="30" name="query" />
     <input type="submit" name="action" value="Search" />
    </form>
    ';
} else {
    echo 'You searched for ' . htmlspecialchars($query) . '<br />';
    for ($i = 0; $i < $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i], "usmarc");
        yaz_range($id[$i], 1, 10);
        yaz_search($id[$i], "rpn", $query);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr />' . $host[$i] . ':';
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i], $p, "string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo nl2br($rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
?>
```

---

yaz_addinfo

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_addinfo — 詳細なエラー情報を返す

説明

*string **yaz_addinfo** (resource $id)*

サーバ上での直近のリクエストに関する詳細なエラーメッセージを返します。

サーバによっては、この関数は yaz_error() と同じ 文字列を返すことがあります。

パラメータ

id

yaz_connect() が返す接続リソース。

返り値

エラーの詳細情報を文字列で返します。直近の処理が成功したか、あるいは サーバから詳細な情報が提供されていない場合には、空の文字列を返します。

参考

- *yaz_error()*
- *yaz_errno()*

---

yaz_ccl_conf

(PHP 4 >= 4.0.5, PECL yaz:0.9-1.0.9)

yaz_ccl_conf — CCL パーサを設定する

説明

*void **yaz_ccl_conf** (resource $id , array $config)*

この関数は、アクセスポイント(CCL限定辞)が定義するサーバに関して CCL クエリパーサと RPN へのマッピングを設定します。

特定の CCL クエリを後で RPN にマップするには、 yaz_ccl_parse() 関数をコールしてください。

パラメータ

id

yaz_connect() が返す接続リソース。

config

設定の配列。配列の各キーが CCL フィールドの名前で、対応する値は RPN へのマッピングを指定する文字列です。

マッピングは、属性型と属性値の組が並んだものです。 属性型と属性値は、等号（=）で区切られ、 組と組の間は空白で区切られます。

詳細な情報は » CCL のページを 参照ください。

返り値

値を返しません。

例

以下の例では、CCL パーサは ti、 au、isbn という 3 つの CCL フィールドをサポートするように設定されます。各フィールドは、等価な BIB-1 へマップされます。 この例では、$id が接続 ID であることを仮定しています。

Example#1 CCL 設定

```
<?php
$fields = array(
  "ti" => "1=4",
  "au"    => "1=1",
  "isbn" => "1=7"
);
yaz_ccl_conf($id, $fields);
?>
```

参考

- yaz_ccl_parse()

---

yaz_ccl_parse

(PHP 4 >= 4.0.5, PECL yaz:0.9-1.0.9)

yaz_ccl_parse — CCL パーサを起動する

説明

bool **yaz_ccl_parse** (resource $id , string $query , array &$result)

この関数は、CCL パーサを起動します。パーサは、指定された CCL FIND クエリを RPN クエリに変換します。これは検索を実行する yaz_search() 関数に渡すためのものです。

有効な CCL フィールドの組を定義するには、この関数の前に yaz_ccl_conf() をコールします。

パラメータ

id

yaz_connect() が返す接続リソース。

query

CCL FIND クエリ。

result

関数が成功した場合、キー rpn に有効な RPN クエリを含む配列となります。

失敗した場合は、以下の 3 つの添え字を含む配列となり、失敗の原因を示します。

- errorcode - CCL エラーコード（整数）
- errorstring - CCL エラー文字列
- errorpos - クエリが失敗したおおよその位置 （文字の位置を整数で表す）

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 CCL のパース

CCL を利用して検索します。以下の例で、$ccl は CCL クエリです。

```
<?php

yaz_ccl_conf($id, $fields);  // yaz_ccl_conf のサンプルを参照ください
```

```
if (!yaz_ccl_parse($id, $ccl, &$cclresult)) {
    echo 'Error: ' . $cclresult["errorstring"];
} else {
    $rpn = $cclresult["rpn"];
    yaz_search($id, "rpn", $rpn);
}
?>
```

---

yaz_close

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_close — YAZ 接続をクローズする

説明

bool **yaz_close** (resource $id)

id で指定した接続をクローズします。

> **注意**: この関数は、persistent オプションを **FALSE** にして yaz_connect() でオープンした 持続的でない接続のみをクローズします。

パラメータ

id

yaz_connect() が返す接続リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- yaz_connect()

---

yaz_connect

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_connect — Z39.50 サーバへの接続を準備する

説明

mixed **yaz_connect** (string $zurl [, mixed $options])

この関数は、成功した場合に接続リソース、失敗した場合にゼロを返します。

yaz_connect() は Z39.50 サーバへの接続を準備します。 この関数は非ブロックモードで動作し、接続を確立しようとしません - 単にあとで yaz_wait() がコールされた際に 接続を実行する準備を行うだけです。

> **注意**: » YAZ プロキシ は、フリーで使用可能な Z39.50 のプロキシです。

パラメータ

zurl

host[:port][/database] という形式の文字列です。 port が省略された場合、210 が使用されます。database が省略された 場合、Default が使用されます。

options

文字列が指定された場合、 Z39.50 V2 認証文字列(OpenAuth)として処理されます。

配列が指定された場合、その配列の内容はオプションとして処理されます。

user

認証用のユーザ名

group

認証用のグループ

password

認証用のパスワード

cookie

セッションクッキー (YAZ プロキシ)

proxy

接続用プロキシ (YAZ プロキシ)

persistent

論理値。**TRUE** の場合、接続は持続的となります。**FALSE** の 場合、接続は持続的ではありません。デフォルトでは 接続は持続的です。

注意: 持続的な接続をオープンした場合、後でそれを yaz_close() によってクローズすることは できません。

piggyback

論理値。**TRUE** の場合、検索時の piggyback が有効になります。 **FALSE** の場合、piggyback が無効になります。デフォルトでは piggyback は有効です。

piggyback を有効にするとより効率的となり、 通常、レコードを最初に取得する際のネットワーク往復時間を 節約することが可能です。しかし、少数の Z39.50 ターゲットは piggyback をサポートしていないか、エレメント集合名を無視します。 この場合、piggyback を無効とする必要があります。

charset

Z39.50 の言語および文字セットネゴシエーションに使用する 文字セットを指定する文字列。以下のような文字列を使用します。 ISO-8859-1 、UTF-8、 UTF-16 。

ほとんどの Z39.50 サーバはこの機能をサポートしません（そのため、 無視されます）。ほとんどのサーバは、クエリやメッセージに ISO-8859-1 エンコーディングを使用します。MARC21/USMARC レコードは この設定の影響を受けません。

preferredMessageSize

ターゲットから取得される全レコードの最大バイト数を指定する整数値。 詳細な情報は » Z39.50 standard を参照ください。

注意: このオプションは PECL YAZ 1.0.5 以降でサポートされます。

maximumRecordSize

ターゲットから取得される単一のレコードの最大バイト数を指定する 整数値。この項目は、 » Z39.50 standard の Exceptional-record-size として参照されます。

注意: このオプションは PECL YAZ 1.0.5 以降でサポートされます。

返り値

成功した場合に接続リソース、エラー時に **FALSE** を返します。

変更履歴

バージョン	説明
4.1.0	*options* が追加されました。

参考

- yaz_close()

---

yaz_database

(PHP 4 >= 4.0.6, PECL yaz:0.9-1.0.9)

yaz_database — セッション内のデータベースを指定する

説明

int **yaz_database** (int $id , string $databases)

この関数は、検索、取得等で使用する 1 つまたは複数のデータベースを 指定します。yaz_connect()をコールする際に指定 したデータベースを上書きします。複数のデータベースは、可算記号 +で区切ります。

この関数は、あるセッションの中で異なるデータベースの集合を使用す ることを可能にします。

成功時に**TRUE**、エラー時に**FALSE**を返します。

---

yaz_element

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_element — 取得時の要素集合の名前を指定する

説明

bool **yaz_element** (resource $id , string $elementset)

この関数は、取得時の要素集合の名前を指定します。

この関数は、yaz_search() あるいは yaz_present() をコールする前に、 取得するレコードの要素集合の名前を指定するために使用します。

注意: この関数が動作していないように見える場合は、 yaz_connect() の piggybacking オプションについての説明を 参照ください。

パラメータ

id

yaz_connect() が返す接続リソース。

elementset

ほとんどのサーバは、 F（完全版）および B（短縮版） をサポートします。

返り値

成功した場合に TRUE を、失敗した場合に FALSE を返します。

---

yaz_errno

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_errno — エラー番号を返す

説明

int **yaz_errno** (resource $id)

id で表される サーバの（直近のリクエストについての）エラー番号を返します。

yaz_errno() は、各ターゲットへのネットワーク接続が 確立された後(つまり、yaz_wait()の後)に コールされる必要があり、直近の操作(例えば検索)が成功したか 失敗したかを判定します。

パラメータ

id

yaz_connect() が返す接続リソース。

返り値

エラーコードを返します。エラーコードは Z39.50 診断コード （通常は Bib-1 診断）あるいは PHP/YAZ が生成するクライアント側の エラーコード（たとえば "Connect failed"、"Init Rejected" など） のいずれかです。

参考

- yaz_error()
- yaz_addinfo()

---

yaz_error

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_error — エラーの内容を返す

説明

string **yaz_error** (resource $id)

yaz_error() は、yaz_errno() で返された直近のエラー番号に対応する英語のメッセージを返します。

パラメータ

id

yaz_connect() が返す接続リソース。

返り値

id で表される(直近のリクエストの）サーバに関するエラーメッセージを返します。 直近の操作が成功した場合には空の文字列が返されます。

参考

- yaz_errno()
- yaz_addinfo()

---

yaz_es_result

(PHP 4 >= 4.2.0, PECL yaz:0.9-1.0.9)

yaz_es_result — 拡張サービスの結果を調査する

説明

array **yaz_es_result** (resource $id)

この関数は、サーバから返された直近の拡張サービスの結果を調査します。 拡張サービスは、**yaz_item_order()** あるいは yaz_es() で起動しま

す。

パラメータ

id

yaz_connect() が返す接続リソース。

返り値

要素 *targetReference* を持つ配列を返します。 この要素には（サーバ側で生成して返された）拡張サービス操作への参照が格納されます。

参考

- *yaz_es()*

---

yaz_es

(PECL yaz:0.9-1.0.9)

yaz_es — 拡張サービスのリクエストを準備する

説明

void ***yaz_es*** *(resource $id , string $type , array $args)*

この関数は、拡張サービスのリクエストを準備します。 拡張サービスとは、レコードの更新や蔵書の注文、データベースの管理などの *Z39.50* のさまざまな機能のことです。

注意: *Z39.50* サーバの多くは、拡張サービスをサポートしていません。

yaz_es() は、拡張サービスリクエストのパッケージを作成し、 それを操作キューに配置します。リクエストをサーバに送信するには *yaz_wait()* を使用します。 *yaz_wait()* の処理が完了すると、拡張サービス操作の結果が *yaz_es_result()* で取得できるようになります。

パラメータ

id

yaz_connect() が返す接続リソース。

type

拡張サービスの形式を表す文字列。 *itemorder*（蔵書の注文）、 *create*（データベースの作成）、 *drop*（データベースの削除）、 *commit*（操作のコミット）、 *update*（レコードの更新）、 *xmlupdate*（XMLの更新）のいずれかです。 個々の形式については、以下の節で指定されます。

args

拡張サービスのオプションに加えてパッケージ固有のオプションを指定する配列。 オプションの形式は、*ZOOM C の C API* で提供されているものと同じです。 *ZOOM » Extended Services* を参照ください。

返り値

値を返しません。

例

Example#1 レコードの更新

```
<?php
$con = yaz_connect("myhost/database");
$args = array (
     "record" => "<gils><title>some  title</title></gils>",
     "syntax" => "xml",
     "action" => "specialUpdate"
);
yaz_es($con, "update", $args);
yaz_wait();
$result = yaz_es_result($id);
?>
```

参考

- *yaz_es_result()*

---

yaz_get_option

(PECL yaz:0.9-1.0.9)

yaz_get_option — 接続に関するオプションの値を返す

説明

string ***yaz_get_option*** *(resource $id , string $name)*

name で指定したオプションの値を返します。

パラメータ

id

yaz_connect() が返す接続リソース。

name

オプションの名前。

返り値

指定したオプションの値を返します。オプションが設定されていない場合は 空文字列を返します。

参考

- *使用可能なオプションについては、yaz_set_option() を参照ください。*

---

yaz_hits

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_hits — 直近の検索に関するヒット数を返す

説明

*int **yaz_hits** (resource $id [, array &$searchresult])*

***yaz_hits()** は、直近の検索に関するヒット数を返します。*

パラメータ

id

yaz_connect() が返す接続リソース。

searchresult

詳細な検索結果情報の配列。

返り値

直近の検索に関するヒット数を返します。検索が行われていない場合は 0 を返します。

検索結果配列がもし指定されていた場合、そこには Z39.50 サーバが返す情報が 含まれます。この情報の形式は、検索結果の応答の一部である SearchResult-1 フォーマットとなります。 SearchResult-1 フォーマットは、さまざまなクエリ(サブクエリ)のヒット数に 関する情報を取得するために使用可能です。 特に、クエリ内の個々の検索語句に関するヒット数を得ることが可能です。 最初のサブクエリについての情報が $array[0] に、2 番目のサブクエリについての 情報が $array[1] に、というような形式となります。

検索結果の項目

要素	説明
id	サブクエリの ID2 (string)
count	結果のカウント / ヒット数 (integer)
subquery.term	サブクエリの語句 (string)
interpretation.term	解釈されたサブクエリの語句 (string)
recommendation.term	推奨されるサブクエリの語句 (string)

***注意**: SearchResult 機能を使用するには、PECL YAZ 1.0.5 以降 および YAZ 2.1.9 以降が必要です。*

***注意**: SearchResult 機能をサポートしている Z39.50 実装は ほとんどありません。*

---

yaz_itemorder

(PHP 4 >= 4.0.5, PECL yaz:0.9-1.0.9)

yaz_itemorder — ILLリクエストパッケージを関してZ39.50 Item Orderを準備する

説明

int **yaz_itemorder** (array $args)

この関数は、Transport ILL (Profile/1)へのZ39.50 Item Order Extended Service用のプロファイルを使用するExtended Servicesリクエストを準備します。 » このページ および » 仕様を参照ください。パラメータargsは、送信するItem Orderリ クエストに関する情報を有するハッシュ配列とする必要があります。こ のハッシュ配列のキーは、対応するASN.1タグのパス名です。例えば、 Item-ID以下にISBNがある場合のキーはitem-id,ISBNとなります。

ILL-Request パラメータは次のようになります。

```
protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date
item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
```

forward-note

Extended Services Requestパッケージおよびitemorderパッケージにも 次のような小数のパラメータがあります。

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

---

yaz_present

(PHP 4 >= 4.0.5, PECL yaz:0.9-1.0.9)

yaz_present — (Z39.50による)取得の準備を行う

説明

bool **yaz_present** (resource $id)

この関数は、検索に成功した後でレコードを取得するための準備を行います。

この関数をコールする前に、取得するレコードの範囲を指定するために まず yaz_range() をコールしておく必要があります。

パラメータ

id

yaz_connect() が返す接続リソース。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

---

yaz_range

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_range — 取得するレコードの範囲を指定する

説明

void **yaz_range** (resource $id , int $start , int $number)

取得するレコードの範囲を指定します。

この関数は、yaz_search() あるいは yaz_present() の前にコールしなければなりません。

パラメータ

id

yaz_connect() が返す接続リソース。

start

取得する最初のレコードの位置を指定します。レコード番号は 1 から yaz_hits() までです。

number

取得するレコードの数を指定します。

返り値

値を返しません。

---

yaz_record

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_record — レコードを返す

説明

string **yaz_record** (resource $id , int $pos , string $type)

yaz_record() 関数は、現在の結果セットの pos で指定した位置にあるレコードを調べます。

パラメータ

id

yaz_connect() が返す接続リソース。

pos

レコードの位置。結果セット内の位置は 1, 2, ... $hits で表されます。 ここで、$hits は yaz_hits() が返す値です。

type

type は、返されるレコードの型を指定します。

> ***注意**: Z39.50/SRW サーバから適切な形式でレコードが返されたことを 確認するのは、アプリケーション側の役目です。このパラメータは 単にクライアント側（PHP/YAZ 内）での型変換についてのみ設定します。*

レコードを文字列/配列に変換することに加え、PHP/YAZ はレコードの 文字セットを変換することも可能です。 特に USMARC/MARC21 の場合に文字セット変換が推奨されます。なぜなら これらは一般的に結果を MARC-8 文字セットで返しますが、ブラウザでは この文字セットがサポートされていないからです。変換を指定するには、 ; charset=from, to を追加します。 from はレコードの元の文字セットで、 to は結果の文字セット（PHP によって 表示されるもの）です。

string

> *レコードは、単純な形式の文字列で返されます。 このモードではすべての MARC レコードが ISO2709 の行単位 フォーマットに変換されますが、これは非常に読みにくいものです。 XML レコードおよび SUTRS は、本来のフォーマットのまま返されます。 GRS-1 は（醜い）行単位フォーマットで返されます。*
>
> *このフォーマットが役に立つのは、結果を手っ取り早く表示したい場合 - デバッグ中など - の場合です。適切な形式で表示するのには適していません。*

xml

> *レコードは、可能ならば XML 文字列として返されます。 このモードでは、すべての MARC レコードは » MARCXML に変換されます。 XML レコードおよび SUTRS は、本来のフォーマットのままで返されます。 GRS-1 はサポートしていません。*
>
> *このフォーマットは string と似ていますが、 MARC レコードが MARCXML に変換される点が違います。*
>
> *このフォーマットは、レコードが XML パーサや XSLT プロセッサに 渡される場合に役立ちます。*

raw

> *レコードは、その本来のフォーマットのままで返されます。 MARC、XML および SUTRS で使用可能です。GRS-1 では動作しません。*
>
> *MARC レコードは ISO2709 文字列として返されます。XML および SUTRS は文字列として返されます。*

syntax

> *レコードの構文が文字列で返されます。たとえば USmarc、GRS-1、 XML などです。*

database

> *レコードの該当位置に関連するデータベースの名前を文字列で返します。*

array

> *レコードは、GRS-1 の構造を反映した配列で返されます。 この型は MARC および GRS-1 で使用可能です。XML、SUTRS はサポートされておらず、実際のレコードが XML あるいは SUTRS であった場合は空文字列が返されます。*
>
> *返される配列には、GRS-1 の各リーフ/内部ノードに対応するリストが 含まれます。個々のリスト内にもリストがあり、 最初の要素が path でその次が data です（もし data が存在する場合）。*
>
> *path は、ルートから各リーフまでの（構造化された GRS-1 レコードの） ツリーコンポーネントの一覧を保持します。各コンポーネントはタグ型で、 タグの値は（type, value のような形式のペアとなります。*
>
> *tags 文字列は、一般に type 3 の tag に対応します。 MARC は配列で返すことも可能です（内部で GRS-1 に変換されます）。*

返り値

位置 pos のレコードを返します。もし指定した位置に レコードが存在しない場合は空文字列を返します。

もし指定した位置にデータベースのレコードが存在しない場合、 空文字列が返されます。

例

Example#1 GRS-1 レコードの配列

このような GRS-1 レコードを想定します。

```
(4,52)Robert  M.  Pirsig
(4,70)
       (4,90)
              (2,7)Transworld  Publishers,  ltd.
```

このレコードは、ルートに 2 つのノードを持っています。ルートレベルの 最初の要素は (4,52) [tag type 4, tag value 52] で、そのデータは Robert M. Pirsig です。ルートレベルの 2 番目の 要素 (4,70) はサブツリーを持っており、そこにはひとつの要素 (4,90) があります。(4,90) はまた別のサブツリー (2,7) を持っており、 そのデータは Transworld Publishers, ltd. です。

このレコードが $p の位置にあるとすると、

```php
<?php

$ar = yaz_record($id, $p, "array");
print_r($ar);

?>
```

この出力は以下のようになります。

```
Array
(
    [0] => Array
        (
            [0] => (4,52)
            [1] => Robert M. Pirsig
        )
    [1] => Array
        (
            [0] => (4,70)
        )
    [2] => Array
        (
            [0] => (4,70)(4,90)
        )
    [3] => Array
        (
            [0] => (4,70)(4,90)(2,7)
            [1] => Transworld Publishers, ltd.
        )
)
```

Example#2 MARCXML の使用

この PHP コードは、MARC21/USMARC レコードを MARCXML として返します。 元のレコードは marc-8（ほとんどの XML パーサはこれを理解できません）なので、 これを UTF-8（すべての XML パーサはこれを理解できなければなりません） に変換します。

```
<?php
$rec = yaz_record($id, $p, "xml;  charset=marc-8,utf-8");
?>
```

レコード $rec は Sablotron XSLT を使用して 以下のように処理可能です。

```
<?php

$xslfile = 'display.xsl';
$processor = xslt_create();
$parms = array('/_xml' => $rec);
$res = xslt_process($processor, 'arg:/_xml', $xslfile, NULL, $parms);
xslt_free($processor);
$res = preg_replace("'</?html[^>]*>'", '', $res);
echo $res;

?>
```

PHP 5 では、Sablotron XSLT ではなく XSL 拡張モジュールが使用される必要があります。

---

yaz_scan_result

(PHP 4 >= 4.0.5, PECL yaz:0.9-1.0.9)

yaz_scan_result — スキャンリクエストの結果を返す

説明

array **yaz_scan_result** (resource $id [, array &$result])

yaz_scan_result() は、直近の yaz_scan() の実行時にサーバから受信した 項目と関連情報を返します。

パラメータ

id

yaz_connect() が返す接続リソース。

result

指定された場合、この配列にスキャン時の応答から得られた追加情報が 代入されます。

- number - 返されたエントリの数。
- stepsize - ステップサイズ。
- position - 項目の位置。
- status - スキャンステータス。

返り値

配列 (0..n-1) を返します。n は返された項目の数です。個々の値は ペアになっており、最初の値が項目名、2 つめの値が結果の数となります。

---

yaz_scan

(PHP 4 >= 4.0.5, PECL yaz:0.9-1.0.9)

yaz_scan — スキャンの準備をする

説明

void ***yaz_scan*** *(resource $id , string $type , string $startterm [, array $flags])*

この関数は、指定した接続上で Z39.50 スキャンリクエストの 準備をします。

実際にスキャンリクエストをサーバへ送信して応答を受信するには、 yaz_wait() をコールする必要があります。 yaz_wait() のコールが完了した後、応答を処理するために yaz_error() および yaz_scan_result() をコールします。

パラメータ

id

yaz_connect() が返す接続リソース。

type

現時点では rpn 型のみがサポートされています。

startterm

スキャンを開始する項目の位置。

開始項目の形式は、パラメータ type で与えられます。

このパラメータの構文は、 yaz_search() で説明した RPN クエリに似ています。 startterm は、ゼロ以上の @attr 演算子の後に トークンがひとつだけ続く形式となります。

flags

オプションのパラメータで、スキャンリクエストの動作を制御する 追加情報を指定します。現在、次の3つの添字が flags 配列から 読み込まれます。 number（リクエストされた語の数）、 position（要求された語の位置）、 stepSize（要求されたステップ数）

返り値

値を返しません。

例

Example#1 タイトルをスキャンする PHP 関数

```
<?php
function scan_titles($id, $startterm)
{
  yaz_scan($id, "rpn", "@attr 1=4 " . $startterm);
  yaz_wait();
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id, &$options);
    echo 'Scan ok; ';
    foreach ($options as $key => $val) {
      echo "$key = $val  ";
    }
    echo '<br /><table>';
    while (list($key, list($k, $term, $tcount)) = each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>$tcount</td></tr>";
    }
    echo '</table>';
  } else {
    echo "Scan failed. Error: " . yaz_error($id) . "<br />";
  }
}
?>
```

---

yaz_schema

(PHP 4 >= 4.2.0, PECL yaz:0.9-1.0.9)

yaz_schema — 取得するスキーマを指定する

説明

void ***yaz_schema*** *(resource $id , string $schema)*

yaz_schema() は取得するスキーマを指定します。

この関数は、 yaz_search() あるいは yaz_present() より前にコールしなければなりません。

パラメータ

id

yaz_connect() が返す接続リソース。

schema

ドット記法の（1.2.840.10003.13.4 のような） OID あるいは登録されているスキーマのひとつ、たとえば GILS-schema、Holdings、 Zthes などのいずれかで指定する必要があります。

返り値

値を返しません。

---

yaz_search

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_search — 検索を準備する

説明

bool **yaz_search** (resource $id , string $type , string $query)

yaz_search() は、指定した接続で検索を行う準備をします。

yaz_connect() と同様にこの関数は非ブロックモードで 動作し、後で yaz_wait() がコールされたときのために 検索の準備をするだけです。

パラメータ

id

yaz_connect() が返す接続リソース。

type

クエリの型を表します。現在は "rpn" のみがサポート されており、この場合 3 番目の引数には Type-1 クエリを前置表記で 指定します。

query

RPN クエリは、Z39.50 規格により定義された Type-1 クエリのテキスト 表現です。しかし、YAZ により使用されるテキスト表現では、演算子が オペランドの前にある前置表記が使用されます。クエリ文字列はトークンの 並びであり、このトークンでは 2 重引用符で括られない限り空白文字は 無視されます。アットマーク(@)で始まるトークンは 演算子とみなされ、そうでない場合は検索項目として処理されます。

RPN 演算子

構文	説明
@and query1 query2	query1 および query2 の積集合
@or query1 query2	query1 および query2 の和集合
@not query1 query2	query1 であり、query2 でない
@set name	結果セットへのリファレンス
@attrset set query	クエリの属性セットを指定します。この構造が使用できるのは全てのクエリの始めに一回だけです。
@attr [set] type=value query	クエリに属性を適用します。型と値は属性型と属性値をそれぞれ指定する整数です。この組みが指定された場合、属性の組を指定します。

属性についての情報は » Z39.50 Maintenance Agency サイトにあります。

> ***注意***: *もう少しわかりやすい記法を利用したい場合は、CCL パーサ関数 - yaz_ccl_conf() および yaz_ccl_parse() を使用してください。*

返り値

*成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。*

例

Example#1 クエリの例

単純な検索は以下のように行います。

```
computer
```

は、"computer" を含む文書にマッチします。属性は指定されていません。

クエリ

```
"knuth  donald"
```

は、"knuth donald" を含む文書にマッチします（サーバが複数の単語による 検索をサポートしている場合）。

このクエリは、同じフレーズに対して 2 つの属性を適用しています。

```
@attr 1=1003 @attr 4=1 "knuth donald"
```

最初の属性は type 1 (Bib-1 use) で、その値は 1003 (Author) です。 2 番目の属性は type 4 (structure) で、その値は 1 (phrase) です。つまりこれは、Donald Knuth が著者である文書にマッチする ことになります。

クエリ

```
@and @or a b @not @or c d e
```

は、次のような意味になります。(a or b) and ((c or d) not e)

さらに複雑な例です。

```
@attrset gils @and @attr 1=4 art @attr 1=2000 company
```

このクエリは GILS 属性セットをすべて使用しています。このクエリは 表題 (Title) に art が含まれ (GILS,BIB-1) 、 配布者 (Distributor) に company が含まれる (GILS) 文書にマッチします。

---

yaz_set_option

(PECL yaz:0.9-1.0.9)

yaz_set_option — 接続に関するひとつあるいは複数のオプションを設定する

説明

```
void yaz_set_option ( resource $id , string $name , string $value )
void yaz_set_option ( resource $id , array $options )
```

指定した接続について、ひとつあるいは複数のオプションを設定します。

パラメータ

id

yaz_connect() が返す接続リソース。

name あるいは options

文字列あるいは配列です。

文字列を指定すると、指定するオプションの名前として扱われます。 その value を指定する必要があります。

配列を指定すると、連想配列 (オプション名 -> オプションの値) として 扱われます。

PHP/YAZ 接続オプション

名前	説明
implementationName	サーバの実装名。
implementationVersion	サーバの実装バージョン。
implementationId	サーバの実装 ID 。
schema	取得するスキーマ。デフォルトではスキーマを使用しません。このオプションを設定することは、yaz_schema() を使用することと等価です。
preferredRecordSyntax	取得するレコードの構文。 デフォルトでは構文を 使用しません。このオプションを設定することは、yaz_syntax() を使用することと等価です。
start	yaz_search() あるいは yaz_present() で取得する最初のレコードのオフセット。最初のレコードの番号は 0 で、2 番目のレコードは 1 です。 このオプションを count とともに設定する ことは、yaz_range() をコールすることと 同じですが、yaz_range() ではレコードの番号が 1 から始まる点が違います。
count	yaz_search() あるいは yaz_present() で取得するレコードの最大数。
elementSetName	取得するエレメントセットの名前。このオプションを設定することは、yaz_element() を使用することと等価です。

value

オプションの新しい値。変更前の値が文字列である場合にのみ これを使用します。

返り値

値を返しません。

---

yaz_sort

(PHP 4 >= 4.0.7, PECL yaz:0.9-1.0.9)

yaz_sort — ソート条件を設定する

説明

void ***yaz_sort*** *(resource $id , string $criteria)*

この関数は、ソート条件を設定して Z39.50 Sort を有効にします。

この関数は、yaz_search() の前に コールします。この関数を単独で使用しても何も意味はありません。 yaz_search() と組み合わせて使用した場合、 検索応答が受信されてから全ての Z39.50 Present で取得される前に Z39.50 Sort が送信されます。

パラメータ

id

yaz_connect() が返す接続リソース。

criteria

field1 flags1 field2 flags2 のような 形式の文字列。*field1* は最初にソートする属性を、そして *field2* は *2* 番目の属性を… のように指定します。

フィールドは、カンマで区切られた *型 = 値* の組からなる数値属性の 組み合わせ（例 *1=4,2=1*）、または 文字列の方法を指定することも可能です（例 *title*）。 フラグは次の文字の並びからなり、空白により区切ることはありません。

ソートフラグ

a

昇順ソート

d

降順ソート

i

大文字小文字を区別しないソート

s

大文字小文字を区別するソート

返り値

値を返しません。

例

Example#1 ソート条件

Bib1 属性のタイトル（*title*）で大文字小文字を区別しない昇順のソートを 行うには、以下のソート方法を使用してください。

```
1=4 ia
```

2 番目のソート条件として著者（*author*）を指定し、大文字小文字を区別する 昇順のソートを行うには以下のようにします。

```
1=4 ia 1=1003 sa
```

---

yaz_syntax

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_syntax — 取得用に適当なレコード構文を指定する

説明

void **yaz_syntax** (*resource $id* , *string $syntax*)

yaz_syntax() は、取得するレコードの構文を指定します。

この関数は *yaz_search()* あるいは *yaz_present()* より前にコールしなければなりません。

パラメータ

id

yaz_connect() が返す接続リソース。

syntax

ドット記法の（*1.2.840.10003.5.10* のような） *OID* あるいは登録されている構文のひとつ（たとえば *sutrs*、 *usmarc*、*grs1*、*xml* など）のいずれかで指定する必要があります。

返り値

値を返しません。

---

yaz_wait

(PHP 4 >= 4.0.1, PECL yaz:0.9-1.0.9)

yaz_wait — *Z39.50* リクエストが完了するまで待つ

説明

mixed **yaz_wait** ([array &$options])

この関数は、関数 yaz_connect()、 yaz_search()、yaz_present()、 yaz_scan()、yaz_itemorder() により準備された発行済のリクエストを、ネットワーク経由で（ブロック モードで）伝送します。

yaz_wait()は、全てのターゲットが全てのリクエストを 完了したか、（エラーの場合には）中断された場合に処理を戻します。

パラメータ

options

オプションの連想配列。

timeout

タイムアウトの秒数を設定します。ターゲットが時間内に応答しなかった 場合は動作していないとみなされ、**yaz_wait()** は終了します。タイムアウトのデフォルト値は 15 秒です。

event

論理型の値。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。 イベントモードでは、成功した場合にリソース、エラー時に **FALSE** を返します。

目次

YP/NIS 関数

導入

(以前はイエローページと呼ばれていた)NISは、(パスワードファイルの ような)重要な管理ファイルのネットワーク管理を可能にします。 詳細は、NIS の man ページおよび » The Linux NIS(YP)/NYS/NIS+ HOWTO を参照ください。 Hal Stern による » Managing NFS and NIS という本もあります。

注意: この拡張モジュールは » PECL レポジトリに移動されており、以下のバージョン以降 PHP にバンドルされなくなっています。 PHP 5.1.0.

注意: この拡張モジュールは Windows 環境では利用できません。

要件

常に利用可能な標準 Unix ライブラリ (libc または libnsl。configure がどちらを使用するかを検出します) の種類に依存する関数はありません。

インストール手順

これらの関数が動作するためには、configure を実行する際に --enable-yp を付ける必要があります。

実行時設定

設定ディレクティブは定義されていません。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

YPERR_ACCESS (integer)
アクセス違反 (これはつい最近追加されたもので、現在は PECL の CVS から取得したバージョンでしか使用できません)。
YPERR_BADARGS (integer)
関数の引数が間違っています。
YPERR_BADDB (integer)
YP データベースに問題があります。
YPERR_BUSY (integer)
データベースの反応がありません。
YPERR_DOMAIN (integer)
そのようなマップはサーバのドメインにありません。
YPERR_KEY (integer)
そのようなキーはマップにありません。
YPERR_MAP (integer)
そのようなマップはサーバのドメインにありません。
YPERR_NODOM (integer)
ローカルドメイン名が設定されていません。
YPERR_NOMORE (integer)
マップデータベースにもうレコードがありません。
YPERR_PMAP (integer)
ポートマッパーと通信できません。
YPERR_RESRC (integer)
リソースの割り当てに失敗しました。
YPERR_RPC (integer)
RPC の失敗 - ドメインがバインドされていません。
YPERR_YPBIND (integer)
ypbind と通信できません。
YPERR_YPERR (integer)
yp サーバあるいはクライアントの内部エラー。
YPERR_YPSERV (integer)
ypserv と通信できません。
YPERR_VERS (integer)
YP バージョンが一致しません。

---

yp_all

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

yp_all — マップを走査し、各エントリ上で関数をコールする

説明

void **yp_all** (string $domain , string $map , string $callback)

警告

この関数は、現在のところ詳細な情報はありません。引数のリストのみが記述されています。

パラメータ

domain

NIS ドメイン名。

map

NIS マップ。

callback

返り値

値を返しません。

---

yp_cat

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

yp_cat — マップ全体を含む配列を返す

説明

array **yp_cat** (string $domain , string $map)

すべてのマップエントリを返します。

パラメータ

domain

NIS ドメイン名。

map

NIS マップ。

返り値

すべてのマップエントリを配列で返します。 マップのキーが配列のインデックスに、 マップのエントリが配列のデータとなります。

---

yp_err_string

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

yp_err_string — 指定したエラーコードに対応するエラー文字列を返す

説明

string ***yp_err_string*** *(int $errorcode)*

指定されたエラーコードに対応する エラーメッセージを返します。何が悪かったのかを正確に調べる際に便利です。

パラメータ

errorcode

エラーコード。

返り値

エラーメッセージを表す文字列を返します。

例

Example#1 NIS エラーの例

```php
<?php
echo "Error: " . yp_err_string(yp_errno());
?>
```

参考

- *yp_errno()*

---

yp_errno

(PHP 4 >= 4.0.6, PHP 5 <= 5.0.5)

yp_errno — 前の操作のエラーコードを返す

説明

int ***yp_errno*** *(void)*

前に行った操作のエラーコードを返します。

返り値

エラー定数 YPERR_XXX のいずれかを返します。

参考

- *yp_err_string()*

---

yp_first

(PHP 4, PHP 5 <= 5.0.5)

yp_first — 指定したマップから最初のキー/値の組を返す

説明

array ***yp_first*** *(string $domain , string $map)*

ドメイン domain のマップ map から、最初のキー/値の組を取得します。

パラメータ

domain

NIS ドメイン名。

map

NIS マップ。

返り値

*最初のキー/値 の組を表す配列、あるいはエラー時に **FALSE** を返します。*

例

Example#1 NIS の最初のエントリを取得する例

```
<?php
$entry = yp_first($domain, "passwd.byname");

$key = key($entry);
$value = $entry[$key];

echo "このマップの最初のエントリのキーは " . $key . "、そして値は " . $value;
?>
```

参考

- yp_next()
- yp_get_default_domain()

---

yp_get_default_domain

(PHP 4, PHP 5 <= 5.0.5)

yp_get_default_domain — マシンのデフォルト NIS ドメインを取得する

説明

*string **yp_get_default_domain** (void)*

ノードのデフォルトドメインを返します。 この後の NIS コールで、これをドメインパラメータとして使用可能です。

NIS ドメインは、NIS マップの集合として説明できます。情報を検索する 必要がある各ホストは、そのホスト自体あるドメインに属しています。 詳細な情報については、冒頭に示したドキュメントを参照ください。

返り値

*ノードのデフォルトドメイン または **FALSE** を返します。 返り値は、この後の NIS コールでドメインパラメータとして使用可能です。*

例

Example#1 デフォルトドメインの例

```
<?php
$domain = yp_get_default_domain();
echo "デフォルトの NIS ドメインは、" . $domain;
?>
```

---

yp_master

(PHP 4, PHP 5 <= 5.0.5)

yp_master — 指定したマップのマスタ NIS サーバのマシン名を返す

説明

*string **yp_master** (string $domain , string $map)*

***yp_master()** は、指定したマップ map のマスタ NIS サーバのマシン名を返します。*

パラメータ

domain

NIS ドメイン名。

map

NIS マップ。

返り値

例

Example#1 NIS マスタの例

```php
<?php
$number  = yp_master($domain, $mapname);
echo "このマップのマスタは、" . $master;
?>
```

参考

- yp_get_default_domain()

---

yp_match

(PHP 4, PHP 5 <= 5.0.5)

yp_match — 検索した行を返す

説明

string **yp_match** (string $domain , string $map , string $key)

指定した map から、 key に対応する値を返します。

パラメータ

domain

NIS ドメイン名。

map

NIS マップ。

key

キーは正確でなければなりません。

返り値

値、あるいはエラー時に **FALSE** を返します。

例

Example#1 NIS 検索の例

```php
<?php
$entry  = yp_match($domain, "passwd.byname", "joe");
echo "マッチしたエントリは、" . $entry;
?>
```

上の例の出力は、たとえば以下のようになります。

```
joe:##joe:11111:100:Joe  User:/home/j/joe:/usr/local/bin/bash
```

参考

- yp_get_default_domain()

---

yp_next

(PHP 4, PHP 5 <= 5.0.5)

yp_next — マップから、次のキー/値の組を返す

説明

array **yp_next** (string $domain , string $map , string $key)

map という名前のマップの中で、指定したキー key の次にある キー/値 の組を返します。

パラメータ

domain

map

key

返り値

次のキー/値 の組を表す配列、あるいはエラー時に **FALSE** を返します。

例

Example#1 NIS next の例

```
<?php
$entry = yp_next($domain, "passwd.byname", "joe");

if (!$entry) {
    echo "エントリがありません\n";
    echo "<!--" . yp_errno() . ": " . yp_err_string() . "-->";
}

$key = key($entry);

echo "joe の次のエントリのキーは " . $key
     . "、値は " . $entry[$key];
?>
```

参考

- yp_first()
- yp_get_default_domain()

---

yp_order

(PHP 4, PHP 5 <= 5.0.5)

yp_order — マップの呼出番号を返す

説明

int **yp_order** (string $domain , string $map)

マップの呼出番号を返します。

パラメータ

domain

map

返り値

マップの呼出番号、あるいはエラー時に **FALSE** を返します。

例

Example#1 NIS 呼出番号の例

```
<?php
    $number = yp_order($domain, $mapname);
    echo "このマップの呼出番号は、" . $number;
?>
```

参考

- yp_get_default_domain()

目次

Zip ファイル関数

導入

この拡張モジュールにより、ZIP 圧縮されたアーカイブとその内部のファイルに対する透過的な読み書きが可能となります。

要件

PHP 4

PHP 4 に同梱されているバージョンは、Guido Draheim 氏による » ZZIPlib バージョン 0.10.6 以降を必要とします。

PHP 5.2.0 以降

この拡張モジュールは、Jean-loup Gailly 氏と Mark Adler 氏による » zlib の機能を利用します。

インストール手順

PHP 4

注意: PHP 4.1.0 以前の Zip サポートは実験的なものです。

警告

PHP 4 の zip 拡張モジュールはメンテナンスされていませんので、 私たちはバンドルされているものよりも PECL 拡張モジュールの利用を推奨します。

Linux システム

これらの関数を使用するには、zip サポートを有効にして PHP をコンパイルしなければなりません。そのためには、設定オプション --with-zip[=DIR] を使用します。 [DIR] は、» ZZIPlib ライブラリのインストールされている場所です。

Windows

Windows ユーザは、これらの関数を使用するために php.ini 内の php_zip.dll を有効にする必要があります。

PHP 5.2.0 以降

Linux systems

これらの関数を利用するには、設定オプション --enable-zip を使用して zip サポートを有効にして PHP をコンパイルしなければなりません。

Windows

Windows ユーザは、これらの関数を使用するために php.ini 内の php_zip.dll を有効にする必要があります。

PECL 経由でのインストール

この PECL 拡張モジュールをインストールする方法は、マニュアルの PECL 拡張モジュールのインストール という章にあります。新規リリース・ダウンロード・ソースファイル・管理者情報・CHANGELOG といった関連する情報については、次の場所にあります。 » http://pecl.php.net/package/zip.

この PECL 拡張モジュール用の DLL は、» PHP のダウンロード ページあるいは » http://pecl4win.php.net/ からダウンロードできます。

PHP 4 の場合、この DLL は PHP の Windows ダウンロードバイナリの extensions/ ディレクトリにあります。

実行時設定

設定ディレクティブは定義されていません。

リソース型

Zip モジュールでは二種類のリソース型が使用されます。 まず最初が Zip アーカイブを表す Zip directory で、 もうひとつはアーカイブのエントリを表す Zip Entry です。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

ZipArchive はクラス定数を使用します。定数には フラグ (FL_)、エラー (ER_) あるいはモード (接頭辞なし) の三種類があります。

ZIPARCHIVE::CREATE (integer)
アーカイブが存在しない場合に、作成します。
ZIPARCHIVE::OVERWRITE (integer)
常に新しいアーカイブを開始します。このモードは、 ファイルが既に存在する場合にはそれを上書きします。
ZIPARCHIVE::EXCL (integer)
アーカイブが既に存在する場合はエラーとします。
ZIPARCHIVE::CHECKCONS (integer)
アーカイブの一貫性チェックを別途行い、 失敗した場合はエラーとします。
ZIPARCHIVE::FL_NOCASE (integer)
名前で検索する際に大文字小文字を区別しません。
ZIPARCHIVE::FL_NODIR (integer)
ディレクトリ要素を無視します。
ZIPARCHIVE::FL_COMPRESSED (integer)
圧縮されたデータを読み込みます。
ZIPARCHIVE::FL_UNCHANGED (integer)
元のデータを使用し、変更内容を無視します。
ZIPARCHIVE::CM_DEFAULT (integer)
圧縮あるいは保存のどちらか有効なほうを実行します。
ZIPARCHIVE::CM_STORE (integer)
保存します (圧縮しません)。
ZIPARCHIVE::CM_SHRINK (integer)
圧縮します。

ZIPARCHIVE::CM_REDUCE_1 (integer)
reduced with factor 1
ZIPARCHIVE::CM_REDUCE_2 (integer)
reduced with factor 2
ZIPARCHIVE::CM_REDUCE_3 (integer)
reduced with factor 3
ZIPARCHIVE::CM_REDUCE_4 (integer)
reduced with factor 4
ZIPARCHIVE::CM_IMPLODE (integer)
imploded
ZIPARCHIVE::CM_DEFLATE (integer)
deflated
ZIPARCHIVE::CM_DEFLATE64 (integer)
deflate64
ZIPARCHIVE::CM_PKWARE_IMPLODE (integer)
PKWARE 方式。
ZIPARCHIVE::CM_BZIP2 (integer)
BZIP2 アルゴリズム。
ZIPARCHIVE::ER_OK (integer)
エラーはありません。
ZIPARCHIVE::ER_MULTIDISK (integer)
複数ディスクの zip アーカイブはサポートされません。
ZIPARCHIVE::ER_RENAME (integer)
一時ファイルの名前変更に失敗しました。
ZIPARCHIVE::ER_CLOSE (integer)
zip アーカイブのクローズに失敗しました。
ZIPARCHIVE::ER_SEEK (integer)
シークエラー。
ZIPARCHIVE::ER_READ (integer)
読み込みエラー。
ZIPARCHIVE::ER_WRITE (integer)
書き込みエラー。
ZIPARCHIVE::ER_CRC (integer)
CRC エラー。
ZIPARCHIVE::ER_ZIPCLOSED (integer)
zip アーカイブはクローズされました。
ZIPARCHIVE::ER_NOENT (integer)
そのファイルはありません。
ZIPARCHIVE::ER_EXISTS (integer)
ファイルが既に存在します。
ZIPARCHIVE::ER_OPEN (integer)
ファイルをオープンできません。
ZIPARCHIVE::ER_TMPOPEN (integer)
一時ファイルの作成に失敗しました。
ZIPARCHIVE::ER_ZLIB (integer)
Zlib エラー。
ZIPARCHIVE::ER_MEMORY (integer)
メモリの確保に失敗しました。
ZIPARCHIVE::ER_CHANGED (string)
エントリが変更されました。
ZIPARCHIVE::ER_COMPNOTSUPP (integer)
圧縮方式がサポートされていません。
ZIPARCHIVE::ER_EOF (integer)
予期せぬ EOF です。
ZIPARCHIVE::ER_INVAL (integer)
無効な引数です。
ZIPARCHIVE::ER_NOZIP (integer)
zip アーカイブではありません。
ZIPARCHIVE::ER_INTERNAL (integer)
内部エラー。
ZIPARCHIVE::ER_INCONS (integer)
矛盾した Zip アーカイブです。
ZIPARCHIVE::ER_REMOVE (integer)
ファイルを削除できません。
ZIPARCHIVE::ER_DELETED (integer)
エントリが削除されました。

例

Example#1 Zip アーカイブの作成

```
<?php

$zip = new ZipArchive();
$filename = "./test112.zip";

if ($zip->open($filename, ZIPARCHIVE::CREATE)!==TRUE) {
    exit("cannot open <$filename>¥n");
}

$zip->addFromString("testfilephp.txt" . time(), "#1 This is a test string added as testfilephp.txt.¥n");
$zip->addFromString("testfilephp2.txt" . time(), "#2 This is a test string added as testfilephp2.txt.¥n");
$zip->addFile($thisdir . "/too.php","/testfromfile.php");
echo "numfiles: " . $zip->numFiles . "¥n";
echo "status:" . $zip->status . "¥n";
$zip->close();
?>
```

Example#2 アーカイブの詳細の出力および一覧表示

```
<?php
$za = new ZipArchive();

$za->open('test_with_comment.zip');
print_r($za);
var_dump($za);
echo "numFiles: " . $za->numFiles . "¥n";
echo "status: " . $za->status . "¥n";
echo "statusSys: " . $za->statusSys . "¥n";
echo "filename: " . $za->filename . "¥n";
```

```
echo "comment: " . $za->comment . "¥n";

for ($i=0; $i<$za->numFiles;$i++) {
    echo "index: $i¥n";
    print_r($za->statIndex($i));
}
echo "numFile:" . $za->numFiles . "¥n";
?>
```

Example#3 Zip ストリームラッパーによる OpenOffice メタ情報の読み込み

```
<?php
$reader = new XMLReader();

$reader->open('zip://' . dirname(__FILE__) . '/test.odt#meta.xml');
$odt_meta = array();
while ($reader->read()) {
    if ($reader->nodeType == XMLREADER::ELEMENT) {
        $elm = $reader->name;
    } else {
        if ($reader->nodeType == XMLREADER::END_ELEMENT && $reader->name == 'office:meta') {
            break;
        }
        if (!trim($reader->value)) {
            continue;
        }
        $odt_meta[$elm] = $reader->value;
    }
}
print_r($odt_meta);
?>
```

この例は旧 API (PHP 4 用) を使用します。まず ZIP ファイルアーカイブをオープンし、アーカイブ内の各ファイルを読み込み、 その内容を出力します。この例で使用するアーカイブ test2.zip は、ZZIPlib のソース配布物に含まれているテスト用アーカイブのひとつです。

Example#4 Zip の使用例

```
<?php

$zip = zip_open("/tmp/test2.zip");

if ($zip) {
    while ($zip_entry = zip_read($zip)) {
        echo "Name:               " . zip_entry_name($zip_entry) . "¥n";
        echo "Actual Filesize:    " . zip_entry_filesize($zip_entry) . "¥n";
        echo "Compressed Size:    " . zip_entry_compressedsize($zip_entry) . "¥n";
        echo "Compression Method: " . zip_entry_compressionmethod($zip_entry) . "¥n";

        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "File Contents:¥n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf¥n";

            zip_entry_close($zip_entry);
        }
        echo "¥n";

    }

    zip_close($zip);

}
?>
```

---

zip_close

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_close — ZIP ファイルアーカイブを閉じる

説明

void **zip_close** (resource $zip)

ZIP ファイルアーカイブを閉じます。

パラメータ

zip

事前に zip_open() でオープンされた ZIP ファイル。

返り値

値を返しません。

参考

- zip_open()
- zip_read()

---

zip_entry_close

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_entry_close — ディレクトリエントリを閉じる

説明

bool **zip_entry_close** (resource $zip_entry)

指定されたディレクトリエントリを閉じます。

パラメータ

zip_entry

zip_entry_open()によりオープンされたディレクトリの エントリ

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- zip_entry_open()
- zip_entry_read()

---

zip_entry_compressedsize

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_entry_compressedsize — ディレクトリエントリの圧縮時のサイズを取得する

説明

int **zip_entry_compressedsize** (resource $zip_entry)

ディレクトリエントリの圧縮時のサイズを取得します。

パラメータ

zip_entry

zip_read()により返されたディレクトリのエントリ

返り値

圧縮後のサイズ

参考

- zip_open()
- zip_read()

---

zip_entry_compressionmethod

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_entry_compressionmethod — ディレクトリエントリの圧縮方法を取得する

説明

string **zip_entry_compressionmethod** (resource $zip_entry)

zip_entry により指定されたディレクトリエントリの 圧縮方法を返します。

パラメータ

zip_entry

zip_read()により返されたディレクトリのエントリ

返り値

圧縮方法

参考

- zip_open()

- zip_read()

---

zip_entry_filesize

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_entry_filesize — ディレクトリエントリの実際のファイルサイズを取得する

説明

int **zip_entry_filesize** (resource $zip_entry)

ディレクトリエントリの実際のファイルサイズを返します。

パラメータ

zip_entry

zip_read()により返されたディレクトリのエントリ

返り値

ディレクトリエントリのサイズ

参考

- zip_open()
- zip_read()

---

zip_entry_name

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_entry_name — ディレクトリエントリの名前を取得する

説明

string **zip_entry_name** (resource $zip_entry)

指定したディレクトリエントリの名前を返します。

パラメータ

zip_entry

zip_read()により返されたディレクトリエントリ

返り値

ディレクトリエントリの名前

参考

- zip_open()
- zip_read()

---

zip_entry_open

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_entry_open — 読込み用にディレクトリエントリをオープンする

説明

bool **zip_entry_open** (resource $zip , resource $zip_entry [, string $mode])

読込み用にzipファイルの中にディレクトリエントリをオープンします。

パラメータ

zip

zip_open()により返された有効なリソースハンドル

zip_entry

zip_read()により返されたディレクトリエントリ

mode

fopen()のマニュアルで指定されたモードのどれか

注意: 現在、mode は無視され、 常に"rb"となります。 これは、PHPのzipサポートが読み込みのみのアクセスであるためです。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

注意: fopen()や他の同種の関数と異なり、 **zip_entry_open()**の戻り値は、 処理の結果のみを示し、ディレクトリエントリの読み込みやクローズの際には 必要とされません。

参考

- zip_entry_close()
- zip_entry_read()

---

zip_entry_read

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_entry_read — オープンされたディレクトリエントリから読み込む

説明

string **zip_entry_read** (resource $zip_entry [, int $length])

オープンされたディレクトリエントリから読み込みます。

パラメータ

zip_entry

zip_read()により返されたディレクトリエントリ

length

返すバイト数。指定されない場合、この関数は1024バイトを読み込みます。

注意: これは、読み込むデータの非圧縮時の長さとなります。

返り値

読み込んだデータ、または、ファイルの終端に達した時に **FALSE** を返します。 reached.

参考

- zip_entry_open()
- zip_entry_close()
- zip_entry_filesize()

---

zip_open

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_open — Zip ファイルアーカイブをオープンする

説明

mixed **zip_open** (string $filename)

読み込み用に新規に Zip アーカイブをオープンします。

パラメータ

filename

オープンする ZIP アーカイブのファイル名。

返り値

後で zip_read() および zip_close() で使用されるリソースハンドル、または、 filename が存在しない場合やその他のエラーが発生した場合はエラーの番号を返します。

参考

- zip_read()
- zip_close()

---

zip_read

(PHP 4 >= 4.0.7, PHP 5 >= 5.2.0, PECL zip:1.0-1.9.0)

zip_read — Zip ファイルアーカイブの中の次のエントリを読み込む

説明

mixed **zip_read** (resource $zip)

Zip ファイルアーカイブの中の次のエントリを読み込みます。

パラメータ

zip

zip_open() でオープン済みの ZIP ファイル。

返り値

後で zip_entry_... 関数で使用されるディレクトリエントリリソース、または、 読み込むエントリがもうない場合に **FALSE**、 その他のエラーが発生した場合はエラー番号を返します。

参考

- zip_open()
- zip_close()
- zip_entry_open()
- zip_entry_read()

ZipArchive::addEmptyDir

(No version information available, might be only in CVS)

ZipArchive::addEmptyDir — 新しいディレクトリを追加する

説明

bool **ZipArchive::addEmptyDir** (string $dirname)

空のディレクトリをアーカイブに追加します。

パラメータ

dirname

追加するディレクトリ。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 アーカイブ内での新規ディレクトリの作成

```
<?php
$zip = new ZipArchive;
if ($zip->open('test.zip') === TRUE) {
    if($zip->addEmptyDir('newDirectory')) {
        echo '新しいディレクトリを作成しました';
    } else {
        echo 'ディレクトリが作成できませんでした';
    }
    $zip->close();
} else {
    echo '失敗しました';
}
?>
```

ZipArchive::addFile

(No version information available, might be only in CVS)

ZipArchive::addFile — 指定したパスからファイルを ZIP アーカイブに追加する

説明

bool **ZipArchive::addFile** (string $filename [, string $localname])

指定したパスから、ファイルを ZIP アーカイブに追加します。

パラメータ

filename

追加するファイルへのパス。

localname

ZIP アーカイブ内部での名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

この例は、ZIP ファイルアーカイブ test.zip をオープンし、ファイル /path/to/index.txt を newname.txt という名前で追加します。

Example#1 オープンおよび抽出

```
<?php
$zip = new ZipArchive;
if ($zip->open('test.zip') === TRUE) {
    $zip->addFile('/path/to/index.txt', 'newname.txt');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

ZipArchive::addFromString

(No version information available, might be only in CVS)

ZipArchive::addFromString — その内容を指定して、ファイルを ZIP アーカイブに追加する

説明

bool **ZipArchive::addFromString** (string $localname , string $contents)

その内容を指定して、ファイルを ZIP アーカイブに追加します。

パラメータ

localname

作成するエントリの名前。

contents

エントリを作成するために使用するデータの内容。 バイナリセーフな形式で使用されます。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 新しいアーカイブへのエントリの追加

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip', ZipArchive::CREATE);
if ($res === TRUE) {
    $zip->addFromString('test.txt', 'ここにファイルの内容を書きます');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

Example#2 アーカイブ内のディレクトリへのファイルの追加

```
<?php
$zip = new ZipArchive;
if ($zip->open('test.zip') === TRUE) {
    $zip->addFromString('dir/test.txt', 'ここにファイルの内容を書きます');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

ZipArchive::close

(No version information available, might be only in CVS)

ZipArchive::close — *アクティブな（オープンされた、あるいは新しく作成された）アーカイブを閉じる*

説明

bool ***ZipArchive::close*** *(void)*

オープンされた、あるいは作成されたアーカイブを閉じ、 変更内容を保存します。このメソッドは、 スクリプトの最後で自動的にコールされます。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

ZipArchive::deleteIndex

(No version information available, might be only in CVS)

ZipArchive::deleteIndex — *インデックスを使用して、アーカイブ内のエントリを削除する*

説明

bool ***ZipArchive::deleteIndex*** *(int $index)*

そのインデックスをもとにして、アーカイブ内のエントリを削除します。

パラメータ

index

削除するエントリのインデックス。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

例

Example#1 インデックスを使用した、アーカイブからのファイルの削除

```
<?php
$zip = new ZipArchive;
if ($zip->open('test.zip') === TRUE) {
    $zip->deleteIndex(2);
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

ZipArchive::deleteName

(No version information available, might be only in CVS)

ZipArchive::deleteName — *名前を使用して、アーカイブからエントリを削除する*

説明

bool ***ZipArchive::deleteName*** *(string $name)*

その名前をもとにして、アーカイブ内のエントリを削除します。

パラメータ

name

削除するエントリの名前。

返り値

成功した場合に ***TRUE*** *を、失敗した場合に* ***FALSE*** *を返します。*

例

Example#1 名前を使用した、アーカイブからのファイルの削除

```
<?php
$zip = new ZipArchive;
if ($zip->open('test1.zip') === TRUE) {
    $zip->deleteName('testfromfile.php');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

ZipArchive::extractTo

(No version information available, might be only in CVS)

ZipArchive::extractTo — アーカイブの内容を展開する

説明

mixed **ZipArchive::extractTo** (string $destination [, mixed $entries])

アーカイブの全体あるいは指定したファイルを、 指定した場所に展開します。

パラメータ

destination

ファイルを展開する場所。

entries

展開するエントリ。単一のエントリ名、 あるいはエントリ名の配列を指定します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

この例は、ZIP ファイルアーカイブをオープンして内部の各ファイルを読み込み、 その内容を表示します。この例で使用するアーカイブ test2.zip は、ZZIPlib のソース配布物に含まれているテスト用アーカイブのひとつです。

Example#1 全エントリの展開

```
<?php
$zip = new ZipArchive;
if ($zip->open('test.zip') === TRUE) {
    $zip->extractTo('/my/destination/dir/');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

Example#2 二つのエントリのみの展開

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test_im.zip');
if ($res === TRUE) {
    $zip->extractTo('/my/destination/dir/', array('pear_item.gif', 'testfromfile.php'));
    $zip->close();
    echo 'ok';
} else {
    echo 'failed';
}
?>
```

ZipArchive::getArchiveComment

(No version information available, might be only in CVS)

ZipArchive::getArchiveComment — ZIP アーカイブのコメントを返す

説明

string **ZipArchive::getArchiveComment** (void)

Zip アーカイブのコメントを返します。

返り値

Zip アーカイブのコメント、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 アーカイブのコメントの出力

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test_with_comment.zip')
if ($res === TRUE) {
    var_dump($zip->getArchiveComment());
    /* あるいはアーカイブのプロパティを使用します */
    var_dump($zip->comment);
} else {
    echo '失敗、コード:' . $res;
}
?>
```

ZipArchive::getCommentIndex

(No version information available, might be only in CVS)

ZipArchive::getCommentIndex — エントリのインデックスを使用して、エントリのコメントを返す

説明

string **ZipArchive::getCommentIndex** (int $index [, int $flags])

エントリのインデックスを使用して、エントリのコメントを返します。

パラメータ

index

エントリのインデックス。

flags

ZIPARCHIVE::FL_UNCHANGED を指定すると、 元の変化していないコメントが返されます。

返り値

成功した場合にコメント、失敗した場合に **FALSE** を返します。

例

Example#1 エントリのコメントの出力

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test1.zip')
if ($res === TRUE) {
    var_dump($zip->getCommentIndex(1));
} else {
    echo '失敗、コード:' . $res;
}
?>
```

ZipArchive::getCommentName

(No version information available, might be only in CVS)

ZipArchive::getCommentName — エントリ名を使用して、エントリのコメントを返す

説明

string **ZipArchive::getCommentName** (string $name [, int $flags])

エントリ名を使用して、エントリのコメントを返します。

パラメータ

name

エントリの名前。

flags

ZIPARCHIVE::FL_UNCHANGED を指定すると、 元の変化していないコメントが返されます。

返り値

成功した場合にコメント、失敗した場合に **FALSE** を返します。

例

Example#1 エントリのコメントの出力

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test1.zip')
if ($res === TRUE) {
    var_dump($zip->getCommentName('test/entry1.txt'));
} else {
    echo '失敗、コード:' . $res;
}
?>
```

ZipArchive::getFromIndex

(No version information available, might be only in CVS)

ZipArchive::getFromIndex — インデックスを使用して、エントリの内容を返す

説明

mixed **ZipArchive::getFromIndex** *(int $index [, int $flags])*

インデックスを使用して、エントリの内容を返します。

パラメータ

index

エントリのインデックス。

flags

アーカイブのオープン時に使用するフラグ。以下の値を *OR* で連結して使用します。

- **ZIPARCHIVE::FL_UNCHANGED**
- **ZIPARCHIVE::FL_COMPRESSED**

返り値

成功した場合にエントリの内容、失敗した場合に **FALSE** を返します。

例

Example#1 ファイルの内容の取得

```
<?php
$zip = new ZipArchive;
if ($zip->open('test.zip') === TRUE) {
    echo $zip->getFromIndex(2);
    $zip->close();
} else {
    echo '失敗';
}
?>
```

ZipArchive::getFromName

(No version information available, might be only in CVS)

ZipArchive::getFromName — 名前を使用して、エントリの内容を返す

説明

mixed **ZipArchive::getFromName** *(string $name [, int $flags])*

名前を使用して、エントリの内容を返します。

パラメータ

name

エントリの名前。

flags

アーカイブのオープン時に使用するフラグ。以下の値を *OR* で連結して使用します。

- **ZIPARCHIVE::FL_UNCHANGED**
- **ZIPARCHIVE::FL_COMPRESSED**

返り値

成功した場合にエントリの内容、失敗した場合に **FALSE** を返します。

例

Example#1 ファイルの内容の取得

```
<?php
$zip = new ZipArchive;
if ($zip->open('test1.zip') === TRUE) {
    echo $zip->getFromName('testfromfile.php');
    $zip->close();
} else {
    echo '失敗';
}
?>
```

Example#2 zip エントリの画像への変換

```
<?php
$z = new ZipArchive();
if ($z->open(dirname(__FILE__) . '/test_im.zip')) {
    $im_string = $z->getFromName("pear_item.gif");
    $im = imagecreatefromstring($im_string);
    imagepng($im, 'b.png');
}
```

```
?>
```

ZipArchive::getNameIndex

(No version information available, might be only in CVS)

ZipArchive::getNameIndex — インデックスを使用して、エントリの名前を返す

説明

string **ZipArchive::getNameIndex** (int $index)

インデックスを使用して、エントリの名前を返します。

パラメータ

index

エントリのインデックス。

返り値

成功した場合に名前、失敗した場合に **FALSE** を返します。

ZipArchive::getStream

(No version information available, might be only in CVS)

ZipArchive::getStream — 名前を使用して、エントリのファイルハンドラ（読み込み専用）を取得する

説明

resource **ZipArchive::getStream** (string $name)

名前を使用して、エントリのファイルハンドラを取得します。 現時点では読み込み操作のみに対応しています。

パラメータ

name

使用するエントリの名前。

返り値

成功した場合にファイルポインタ（リソース）、 失敗した場合に **FALSE** を返します。

例

Example#1 エントリの内容を fread で取得し、それを保存する

```
<?php
$content = '';
$z = new ZipArchive();
if ($z->open('test.zip')) {
    $fp = $z->getStream('test');
    if(!$fp) exit("失敗\n");

    while (!feof($fp)) {
        $contents .= fread($fp, 2);
    }

    fclose($fp);
    file_put_contents('t',$contents);
    echo "終了\n";
}
?>
```

Example#2 先ほどの例と同じことを、fopen および zip ストリームラッパーで行う

```
<?php
$fp = fopen('zip://' . dirname(__FILE__) . '/test.zip#test', 'r');
if (!$fp) {
    exit("オープンできません\n");
}
while (!feof($fp)) {
    $contents .= fread($fp, 2);
    echo "$contents¥n";
}
fclose($fp);
echo "終了\n";
?>
```

Example#3 ストリームラッパーと画像の組み合わせ。xml 関数とでも使用可能

```
<?php
$im = imagecreatefromgif('zip://' . dirname(__FILE__) . '/test_im.zip#pear_item.gif');
imagepng($im, 'a.png');
?>
```

ZipArchive::locateName

(No version information available, might be only in CVS)

ZipArchive::locateName — アーカイブ内のエントリのインデックスを返す

説明

mixed **ZipArchive::locateName** (string $name [, int $flags])

名前を使用して、エントリの場所を取得します。

パラメータ

name

探したいエントリの名前。

flags

この関数は、アーカイブ内の指定した名前のファイルのインデックスを返します。 フラグには、次の値を OR で連結して指定します。 あるいは何もしていしない場合は 0 とします。

- **ZIPARCHIVE::FL_NOCASE**
- **ZIPARCHIVE::FL_NODIR**

返り値

成功した場合にエントリのインデックス、失敗した場合に **FALSE** を返します。

例

Example#1 アーカイブを作成し、locateName を使用する

```
<?php
$file = 'testlocate.zip';

$zip = new ZipArchive;
if ($zip->open($file, ZIPARCHIVE::CREATE) !== TRUE) {
    exit('失敗');
}

$zip->addFromString('entry1.txt', 'entry #1');
$zip->addFromString('entry2.txt', 'entry #2');
$zip->addFromString('dir/entry2d.txt', 'entry #2');

if (!$zip->status == ZIPARCHIVE::ER_OK) {
    echo "zip の書き込みに失敗\n";
}
$zip->close();

if ($zip->open($file) !== TRUE) {
    exit('失敗');
}

echo $zip->locateName('entry1.txt') . "¥n";
echo $zip->locateName('eNtry2.txt') . "¥n";
echo $zip->locateName('eNtry2.txt', ZIPARCHIVE::FL_NOCASE) . "¥n";
echo $zip->locateName('enTRy2d.txt', ZIPARCHIVE::FL_NOCASE|ZIPARCHIVE::FL_NODIR) . "¥n";
$zip->close();

?>
```

ZipArchive::open

(No version information available, might be only in CVS)

ZipArchive::open — ZIP ファイルアーカイブをオープンする

説明

mixed **ZipArchive::open** (string $filename [, int $flags])

新しい zip アーカイブを、読み込み/書き込み/変更用にオープンします。

パラメータ

filename

オープンする ZIP アーカイブのファイル名。

flags

アーカイブのオープンに使用するモード。

- **ZIPARCHIVE::OVERWRITE**
- **ZIPARCHIVE::CREATE**

- **ZIPARCHIVE::EXCL**
- **ZIPARCHIVE::CHECKCONS**

返り値

エラーコード

成功した場合に **TRUE**、それ以外の場合にエラーコードを返します。

- **ZIPARCHIVE::ER_EXISTS**
- **ZIPARCHIVE::ER_INCONS**
- **ZIPARCHIVE::ER_INVAL**
- **ZIPARCHIVE::ER_MEMORY**
- **ZIPARCHIVE::ER_NOENT**
- **ZIPARCHIVE::ER_NOZIP**
- **ZIPARCHIVE::ER_OPEN**
- **ZIPARCHIVE::ER_READ**
- **ZIPARCHIVE::ER_SEEK**

例

この例は、ZIP ファイルアーカイブをオープンして内部の各ファイルを読み込み、 その内容を表示します。この例で使用するアーカイブ test2.zip は、ZZIPlib のソース配布物に含まれているテスト用アーカイブのひとつです。

Example#1 オープンおよび展開

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip')
if ($res === TRUE) {
    echo '成功';
    $zip->extractTo('test');
    $zip->close();
} else {
    echo '失敗、コード:' . $res;
}
?>
```

Example#2 アーカイブの作成

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip', ZipArchive::CREATE);
if ($res === TRUE) {
    $zip->addFromString('test.txt', 'ここにファイルの内容を書きます');
    $zip->addFile('data.txt', 'entryname.txt');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

ZipArchive::renameIndex

(No version information available, might be only in CVS)

ZipArchive::renameIndex — インデックスを使用してエントリ名を変更する

説明

bool **ZipArchive::renameIndex** (int $index , string $newname)

インデックスを使用して、エントリ名を変更します。

パラメータ

index

名前を変更するエントリのインデックス。

newname

新しい名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 エントリ名の変更

```php
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip')
if ($res === TRUE) {
    $zip->renameIndex(2,'newname.txt');
    $zip->close();
} else {
    echo '失敗、コード:' . $res;
}
?>
```

ZipArchive::renameName

(No version information available, might be only in CVS)

ZipArchive::renameName — 名前を使用してエントリ名を変更する

説明

bool **ZipArchive::renameName** (string $name , string $newname)

名前を使用して、エントリ名を変更します。

パラメータ

name

名前を変更するエントリの名前。

newname

新しい名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 エントリ名の変更

```php
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip');
if ($res === TRUE) {
    $zip->renameName('currentname.txt','newname.txt');
    $zip->close();
} else {
    echo '失敗、コード:' . $res;
}
?>
```

ZipArchive::setArchiveComment

(No version information available, might be only in CVS)

ZipArchive::setArchiveComment — ZIP アーカイブのコメントを設定する

説明

mixed **ZipArchive::setArchiveComment** (string $comment)

ZIP アーカイブのコメントを設定します。

パラメータ

comment

コメントの内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 アーカイブの作成およびコメントの設定

```php
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip', ZipArchive::CREATE);
if ($res === TRUE) {
    $zip->addFromString('test.txt', 'ここにファイルの内容を書きます');
    $z->setArchiveComment('新しいアーカイブのコメント');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
```

```
}
?>
```

ZipArchive::setCommentIndex

(No version information available, might be only in CVS)

ZipArchive::setCommentIndex — インデックスを使用してエントリのコメントを設定する

説明

mixed **ZipArchive::setCommentIndex** (int $index , string $comment)

インデックスを使用して、エントリのコメントを設定します。

パラメータ

index

エントリのインデックス。

comment

コメントの内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 アーカイブをオープンし、エントリのコメントを設定する

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip');
if ($res === TRUE) {
    $z->setCommentIndex(2, '新しいエントリのコメント');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

ZipArchive::setCommentName

(No version information available, might be only in CVS)

ZipArchive::setCommentName — 名前を使用してエントリのコメントを設定する

説明

mixed **ZipArchive::setCommentName** (string $name , string $comment)

名前を使用して、エントリのコメントを設定します。

パラメータ

name

エントリの名前。

comment

コメントの内容。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 アーカイブをオープンし、エントリのコメントを設定する

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip');
if ($res === TRUE) {
    $z->setCommentName('entry1.txt', '新しいエントリのコメント');
    $zip->close();
    echo '成功';
} else {
    echo '失敗';
}
?>
```

ZipArchive::statIndex

(No version information available, might be only in CVS)

ZipArchive::statIndex — インデックスを使用してエントリの詳細を取得する

説明

mixed **ZipArchive::statIndex** (int $index [, int $flags])

この関数は、インデックスを使用してエントリの詳細情報を取得します。

パラメータ

index

エントリのインデックス。

flags

ZIPARCHIVE::FL_UNCHANGED を OR で連結すると、 アーカイブ内に最初に記録された際の情報を取得します。 変更内容は無視されます。

返り値

エントリの詳細を含む配列、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 エントリの情報の出力

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip')
if ($res === TRUE) {
    print_r($zip->statIndex(3));
    $zip->close();
} else {
    echo '失敗、コード:' . $res;
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [name] => foobar/baz
    [index] => 3
     [crc] => 499465816
    [size] => 27
     [mtime] => 1123164748
     [comp_size] => 24
     [comp_method] => 8
)
```

ZipArchive::statName

(No version information available, might be only in CVS)

ZipArchive::statName — 名前を使用してエントリの詳細を取得する

説明

mixed **ZipArchive::statName** (name $name [, int $flags])

この関数は、名前を使用してエントリの詳細情報を取得します。

パラメータ

name

エントリの名前。

flags

名前をどのように探すのかを設定します。 また **ZIPARCHIVE::FL_UNCHANGED** を OR で連結すると、 アーカイブ内に最初に記録された際の情報を取得します。 変更内容は無視されます。

- **ZIPARCHIVE::FL_NOCASE**
- **ZIPARCHIVE::FL_NODIR**
- **ZIPARCHIVE::FL_UNCHANGED**

返り値

エントリの詳細を含む配列、あるいは失敗した場合に **FALSE** を返します。

例

Example#1 エントリの情報の出力

```
<?php
$zip = new ZipArchive;
$res = $zip->open('test.zip')
if ($res === TRUE) {
    print_r($zip->statName('foobar/baz'));
    $zip->close();
} else {
    echo '失敗、コード:' . $res;
}
?>
```

上の例の出力は、たとえば以下のようになります。

```
Array
(
     [name] => foobar/baz
     [index] => 3
     [crc] => 499465816
     [size] => 27
     [mtime] => 1123164748
     [comp_size] => 24
     [comp_method] => 8
)
```

ZipArchive::unchangeAll

(No version information available, might be only in CVS)

ZipArchive::unchangeAll — アーカイブに対するすべての変更を取り消す

説明

mixed **ZipArchive::unchangeAll** (void)

アーカイブに対するすべての変更を取り消します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

ZipArchive::unchangeArchive

(No version information available, might be only in CVS)

ZipArchive::unchangeArchive — アーカイブ全体に対して行われたすべての変更を取り消す

説明

mixed **ZipArchive::unchangeArchive** (void)

アーカイブ全体に対して行われたすべての変更を取り消します。 現在は、これはアーカイブのコメントに対する変更のみを取り消します。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

ZipArchive::unchangeIndex

(No version information available, might be only in CVS)

ZipArchive::unchangeIndex — 指定したインデックスのエントリに対するすべての変更を取り消す

説明

mixed **ZipArchive::unchangeIndex** (int $index)

指定したインデックスのエントリに対するすべての変更を取り消します。

パラメータ

index

エントリのインデックス。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

ZipArchive::unchangeName

(No version information available, might be only in CVS)

ZipArchive::unchangeName — 指定した名前のエントリに対するすべての変更を取り消す

説明

mixed **ZipArchive::unchangeName** (string $name)

エントリに対するすべての変更を取り消します。

パラメータ

name

エントリの名前。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

目次

---

zlib 圧縮関数

導入

このモジュールにより gzip (.gz) で圧縮されたファイルを読み書きすることが可能となります。 この際、ファイルシステム 関数の gzip 圧縮対応版（非圧縮ファイルも扱えますが、 ソケットは扱えません）を使用します。

> **注意**: バージョン 4.0.4 で .gz ファイルに対応した fopen のラッパー を導入しました。 fopen() に渡すファイル名または パス名に zlib: をつけると、通常の f*() ファイルアクセス関数を使用して圧縮ファイルに透過的にアクセスすることが できます。 PHP 4.3.0 において、';' を含むファイル名との曖昧さを避けるため zlib: は compress.zlib:// に変更されました。 fopencookie() 関数は もはや必要ありません。詳細な情報は、 圧縮ストリーム にあります。

要件

このモジュールは、Jean-loup Gailly および Mark Adler による » zlib の関数を使用します。 このモジュールを使用するには、zlib バージョン >= 1.0.9 を使用する必要があります。

インストール手順

PHPにおけるZlibサポートは、デフォルトでは利用できません。 Zlibサポートを有効にするには、PHPのコンパイル時にconfigureの オプションに --with-zlib[=DIR]を 指定してコンパイルする必要があります。

Windows 版の PHP にはこの拡張モジュールのサポートが組み込まれています。これらの関数を使用するために拡張モジュールを追加でロードする必要はありません。

> **注意**: PHP 4.3.0 以降、zlib モジュールは Windows 用 php バイナリにビルトインされています。

実行時設定

php.ini の設定により動作が変化します。

zlib 拡張モジュールは、 ブラウザがサポートする場合にページを透過的に圧縮するオプションを提供します。 ここで、 設定ファイル php.ini のオプションには、以下の 3 種類があります。

Zlib設定オプション

名前	デフォルト	変更の可否	変更履歴
zlib.output_compression	"0"	PHP_INI_ALL	PHP 4.0.5 から利用可能
zlib.output_compression_level	"-1"	PHP_INI_ALL	PHP 4.3.0 から利用可能
zlib.output_handler	""	PHP_INI_ALL	PHP 4.3.0 から利用可能

PHP_INI_* 定数の詳細および定義については php.ini ディレクティブ を参照してください。

以下に設定ディレクティブに関する簡単な説明を示します。

zlib.output_compression boolean/integer

透過的なページ圧縮を行うかどうか。php.ini または Apache の設定でこのオプションが、"On" に設定された場合、 ブラウザが "Accept-Encoding: gzip" または "deflate" ヘッダを送信する場合に、ページは圧縮されます。 "Content-Encoding: gzip" （および "deflate"） と "Vary: Accept-Encoding" ヘッダが出力に追加されます。 実行時、何らかのデータを送出する前にのみ設定することが可能です。

このオプションも論理値 "On"/"Off" のかわりに整数値をとることができ、 これを用いて出力のバッファサイズ（デフォルトは 4KB）を設定することができます。

> **注意**: このオプションに 'On' を設定した場合、 output_handler を空にする必要があります! かわりに zlib.output_handler を使用する必要があります。

zlib.output_compression_level integer

透過的出力圧縮で使用される圧縮レベル。

zlib.output_handler string

zlib.output_compression が有効な場合、 他の出力ハンドラを指定することはできません。 この設定は、output_handler と同じですが、順番が異なります。

リソース型

この拡張モジュールでは、ファイルポインタリソースを定義しています。これは gzopen() が返すものです。

定義済み定数

以下の定数が定義されています。この関数の拡張モジュールが PHP 組み込みでコンパイルされているか、実行時に動的にロードされている場合のみ使用可能です。

FORCE_GZIP (integer)
FORCE_DEFLATE (integer)

例

テンポラリファイルをオープンし、テスト用文字列を書きこみ、 続いて、このファイルの内容を 2 回出力します。

Example#1 簡単な Zlib の例

```
<?php

$filename = tempnam('/tmp', 'zlibtest') . '.gz';
echo "<html>¥n<head></head>¥n<body>¥n<pre>¥n";
$s = "Only a test, test, test, test, test, test, test, test!¥n";

// 最大限の圧縮を指定して書きこみ用にファイルをオープン
$zp = gzopen($filename, "w9");

// 文字列をファイルに書きこむ
gzwrite($zp, $s);
```

```
// ファイルを閉じる
gzclose($zp);

// 読みこみ用にファイルをオープン
$zp = gzopen($filename, "r");

// 3 文字読みこむ
echo gzread($zp, 3);

// ファイルの終端まで出力して閉じる
gzpassthru($zp);
gzclose($zp);

echo "¥n";

// ファイルをオープンし、内容を出力する (2回目)
if (readgzfile($filename) != strlen($s)) {
        echo "Error with zlib functions!";
}
unlink($filename);
echo "</pre>¥n</body>¥n</html>¥n";

?>
```

---

gzclose

(PHP 4, PHP 5)

gzclose — 開かれたgzファイルへのポインタを閉じる

説明

bool **gzclose** (resource $zp)

与えられた gz ファイルへのポインタを閉じます。

パラメータ

zp

gzファイルポインタ。有効なファイルポインタであり、かつ、 gzopen() によりオープンできたファイルを指している必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

例

Example#1 gzclose() の例

```
<?php
$gz = gzopen('somefile.gz','w9');
gzputs ($gz, 'I was added to somefile.gz');
gzclose($gz);
?>
```

参考

- gzopen()

---

gzcompress

(PHP 4 >= 4.0.1, PHP 5)

gzcompress — 文字列を圧縮する

説明

string **gzcompress** (string $data [, int $level])

この関数は、ZLIB データフォーマットを用いて 与えられた文字列を圧縮します。

ZLIB 圧縮アルゴリズムについての詳細は、 "» ZLIB Compressed Data Format Specification version 3.3" (RFC 1950) を参照ください。

> **注意**: これは、いくつかのヘッダデータを有する gzip 圧縮と同じでは ありません。 gzip圧縮については、gzencode()を参照ください。

パラメータ

data

圧縮するデータ

level

圧縮レベル。0 で圧縮無し、9 で最大限の圧縮を指定できます。

返り値

圧縮された文字列、もしくはエラーの場合 **FALSE** 。

例

Example#1 gzcompress() の例

```
<?php
$compressed = gzcompress('Compress me', 9);
echo $compressed;
?>
```

参考

- gzdeflate()
- gzinflate()
- gzuncompress()
- gzencode()

---

gzdecode

(No version information available, might be only in CVS)

gzdecode — gzip 圧縮された文字列をデコードする

説明

string **gzdecode** (string $data [, int $length])

この関数は、入力された data をデコードしたものを返します。

パラメータ

data

デコードするデータ。これは gzencode() でエンコードされたものです。

length

デコードするデータの最大長。

返り値

デコードされた文字列、あるいはエラーが発生した場合に **FALSE** を返します。

参考

- gzencode()

---

gzdeflate

(PHP 4 >= 4.0.4, PHP 5)

gzdeflate — 文字列を deflate 圧縮する

説明

string **gzdeflate** (string $data [, int $level])

この関数は、DEFLATE データフォーマットを用いて 与えられた文字列を圧縮します。

DEFLATE 圧縮アルゴリズムについての詳細は、 "» DEFLATE Compressed Data Format Specification version 1.3" (RFC 1951) を参照ください。

パラメータ

data

収縮させるデータ

level

圧縮レベル。圧縮しない場合に0、最大限の圧縮をする場合に9を指定可能です。 指定しない場合、デフォルトの圧縮レベルは zlib ライブラリのデフォルト圧縮レベルになります。

返り値

収縮された文字列、もしくはエラーの場合 **FALSE** 。

例

Example#1 gzdeflate() の例

```
<?php
$compressed = gzdeflate('Compress me', 9);
echo $compressed;
?>
```

参考

- gzinflate()
- gzcompress()
- gzuncompress()
- gzencode()

---

gzencode

(PHP 4 >= 4.0.4, PHP 5)

gzencode — gzip 圧縮された文字列を作成する

説明

string **gzencode** (string $data [, int $level [, int $encoding_mode]])

この関数は、入力 data を **gzip** プログラムの出力と互換性のある形式で圧縮 して返します。

GZIPファイルフォーマットに関する詳細な情報については、次のドキュ メントを参照ください。 » GZIP file format specification version 4.3 (RFC 1952)

パラメータ

data

エンコードするデータを指定します

level

圧縮レベルを指定します。 圧縮をしない場合に 0、最大限の圧縮を行う場合に9を指定可能です。 指定されない場合のデフォルト圧縮レベルは、zlib ライブラリのデフォルト圧縮レベルになります。

encoding_mode

エンコーディングモードを指定します。 **FORCE_GZIP**（デフォルト）もしくは **FORCE_DEFLATE** を指定可能です。

もし **FORCE_DEFLATE** を使用した場合、 crc32 チェックサムが付加されていない gzip ファイルヘッダに続いて標準的な zlib 収縮文字列 (zlib ヘッダを含む) を受け取るでしょう。

返り値

エンコードされた文字列、もしくはエラー発生時に **FALSE**

変更履歴

バージョン	説明
4.2.0	パラメータ encoding_mode が追加されました。

例

結果データは標準的な .gz ファイルを構成するための適当なヘッダとデータ構造を含みます。

Example#1 gzip ファイルの生成

```
<?php
$data = implode("", file("bigfile.txt"));
$gzdata = gzencode($data, 9);
$fp = fopen("bigfile.txt.gz", "w");
fwrite($fp, $gzdata);
fclose($fp);
?>
```

参考

- gzdecode()
- gzdeflate()
- gzinflate()
- gzuncompress()
- gzcompress()

---

gzeof

(PHP 4, PHP 5)

gzeof — gz ファイルポインタがファイル終端かどうか調べる

説明

int **gzeof** (resource $zp)

与えられた GZ ファイルポインタが EOF（ファイル終端）を指すかどうかを調べます。

パラメータ

zp

gzファイルポインタは、有効なファイルポインタであり、かつ、 gzopen()によりオープンできたファイルを指している 必要があります。

返り値

gz ファイルポインタが EOF を指す、もしくはエラーが発生した場合 **TRUE**、 そうでなければ **FALSE** を返します。

例

Example#1 gzeof() の例

```
<?php
$gz = gzopen('somefile.gz', 'r');
while (!gzeof($gz)) {
   echo gzgetc($gz);
}
gzclose($gz);
?>
```

---

gzfile

(PHP 4, PHP 5)

gzfile — gzファイル全体を配列に読み込む

説明

array **gzfile** (string $filename [, int $use_include_path])

ファイルを配列にして返すこと以外は readgzfile() と同じです。

パラメータ

filename

ファイル名を指定します。

use_include_path

もし include_path にあるファイルも検索したい場合、 このオプションパラメータに 1 を設定することができます。

返り値

ファイルを含む配列で、要素毎に 1 行ずつ含んでいます。

例

Example#1 gzfile() の例

```
<?php
$lines = gzfile('somefile.gz');
foreach ($lines as $line) {
    echo $line;
}
?>
```

参考

- readgzfile()
- gzopen()

---

gzgetc

(PHP 4, PHP 5)

gzgetc — gz ファイルへのポインタから文字を得る

説明

string **gzgetc** (resource $zp)

与えられた gz ファイルポインタから（非圧縮の）1 文字を読み込み、これを含む 文字列を返します。

パラメータ

zp

gz ファイルポインタを指定します。 これは有効なファイルポインタであり、かつ、 gzopen() によりオープンできたファイルを指している必要があります。

返り値

非圧縮の文字列、もしくは EOF (gzeof() とは異なる）の場合の **FALSE**

例

Example#1 gzgetc() の例

```
<?php
$gz = gzopen('somefile.gz', 'r');
while (!gzeof($gz)) {
   echo gzgetc($gz);
}
gzclose($gz);
?>
```

参考

- gzopen()
- gzgets()

---

gzgets

(PHP 4, PHP 5)

gzgets — ファイルポインタから 1 行を得る

説明

string **gzgets** (resource $zp , int $length)

与えられたファイルポインタから最大 length - 1 バイトの文字を読み込み、 これを含む（非圧縮の）文字列を返します。 length - 1 バイトを読み込むか、改行または EOF になった場合、 （どれかが最初にきた時点で）読み込みを終了します。

パラメータ

zp

gz ファイルポインタを指定します。 これは有効なファイルポインタであり、かつ、 gzopen() によりオープンできたファイルを指している必要があります。

length

取得するデータ長を指定します。

返り値

非圧縮の文字列、もしくはエラー時に **FALSE**

例

Example#1 gzgets() の例

```
<?php
$handle = gzopen('somefile.gz', 'r');
while (!gzeof($handle)) {
   $buffer = gzgets($handle, 4096);
   echo $buffer;
}
gzclose($handle);
?>
```

参考

- gzopen()
- gzgetc()
- gzwrite()

---

gzgetss

(PHP 4, PHP 5)

gzgetss — gzファイルへのポインタから1行を得て、HTMLタグを取り除く

説明

string ***gzgetss*** *(resource $zp , int $length [, string $allowable_tags])*

gzgetss()*は読み込んだテキストから HTML および PHP タグを全て取り除こうとすることを除いて、 gzgets()と同じです。*

パラメータ

zp

gz ファイルポインタを指定します。 これは有効なファイルポインタであり、かつ、 gzopen() によりオープンできたファイルを指している必要があります。

length

取得するデータ長を指定します。

allowable_tags

このオプションパラメータにより、 取り除かないタグを指定することができます。

返り値

非圧縮かつタグが取り除かれた文字列、もしくはエラー時に ***FALSE***

変更履歴

バージョン	説明
3.0.13 と 4.0.0	*allowable_tags* が追加されました。

例

Example#1 gzgetss() の例

```
<?php
$handle = gzopen('somefile.gz', 'r');
while (!gzeof($handle)) {
    $buffer = gzgetss($handle, 4096);
    echo $buffer;
}
gzclose($handle);
?>
```

参考

- gzopen()
- gzgets()
- strip_tags()

---

gzinflate

(PHP 4 >= 4.0.4, PHP 5)

gzinflate — deflate圧縮された文字列を解凍する

説明

string ***gzinflate*** *(string $data [, int $length])*

この関数は収縮された文字列を伸長します。

パラメータ

data

gzdeflate() により圧縮されたデータを指定します

length

デコードする最大データ長を指定します

返り値

オリジナルの無圧縮なデータ、もしくはエラー時に ***FALSE***

この関数は、もし無圧縮なデータが圧縮された入力 data の 32768 倍、もしくはオプションのパラメータ length 以上の場合、エラーを返します。

例

Example#1 gzinflate() の例

```
<?php
$compressed   = gzdeflate('Compress me', 9);
$uncompressed = gzinflate($compressed);
echo $uncompressed;
?>
```

参考

- gzdeflate()
- gzcompress()
- gzuncompress()
- gzencode()

---

gzopen

(PHP 4, PHP 5)

gzopen — gz ファイルを開く

説明

resource **gzopen** (string $filename , string $mode [, int $use_include_path])

読み込みもしくは書き込みのために gzip (.gz) ファイルをオープンします。

gzopen() は、gzip フォーマットでないファイルの読み込みについても使用することができます。 この場合、gzread()は、ファイルを解凍せずに直接読み 込まれます。

パラメータ

filename

ファイル名を指定します

mode

fopen() と同じ (rb または wb) ですが、 圧縮レベル (wb9) または圧縮の方策、 つまり、wb6f のようにフィルターを通したデータを f で指定したり、h でハフマン圧縮のみを行うことを指定したりすることができます。 (方策に関するパラメータの詳細については、zlib.h の中の deflateInit2 の説明を参照ください。)

use_include_path

このオプションパラメータを 1 にすることにより、 include_pathにあるファイルも 検索することができます。

返り値

オープンしたファイルへのファイルポインタを返します。その後、 このファイルディスクプリタから読み込んだ全ては透過的に解凍され、 書き込んだものは圧縮されます。

オープンに失敗した場合、この関数は **FALSE** を返します。

例

Example#1 gzopen() の例

```
<?php
$fp = gzopen("/tmp/file.gz", "r");
?>
```

参考

- gzclose()

---

gzpassthru

(PHP 4, PHP 5)

gzpassthru — gzファイルへのポインタから残りのデータ全部を出力する

説明

int **gzpassthru** (resource $zp)

gz ファイルポインタについて現在位置から EOF までデータを読み込み、 標準出力に (解凍された) 結果を書き込みます。

> **注意**: すでにデータをファイルに書き込んでいる場合、 ファイルポインタをファイルの先頭にセットするために gzrewind() をコールする必要があるかも知れません。

ヒント

修正や特定のオフセットへの移動なしに ファイルの内容を出力バッファにダンプしたいだけの場合、 readgzfile() 関数を使用することもできます。 これにより、gzopen() をコールする手間を省くことができます。

パラメータ

zp

gz ファイルポインタを指定します。これは有効なファイルポインタであり、 かつ、gzopen() によりオープンできたファイルを指している必要があります。

返り値

gz から読み込んで解凍され、入力に渡された文字数。 もしくはエラー時に **FALSE** 。

例

Example#1 gzpassthru() の例

```
<?php
$fp = gzopen('file.gz', 'r');
gzpassthru($fp);
gzclose($fp);
?>
```

---

gzputs

(PHP 4, PHP 5)

gzputs — のエイリアス gzwrite()

説明

この関数は次の関数のエイリアスです。 gzwrite().

---

gzread

(PHP 4, PHP 5)

gzread — バイナリ対応のgzファイル読み込み

説明

string **gzread** (resource $zp , int $length)

gzread() は、最大 length バイトのデータを zp が指す gz ファイルポインタ から読み込みます。(解凍された) length バイトのデータが読み込まれたか、 EOF に達したとき、読み込みは終了します。

パラメータ

zp

gz ファイルポインタを指定します。これは有効なファイルポインタであり、 かつ、gzopen() によりオープンできたファイルを指している必要があります。

length

読み込むバイト数を指定します。

返り値

読み込まれたデータ

例

Example#1 gzread() の例

```
<?php
// gz ファイルの内容を文字列に読み込む
$filename = "/usr/local/something.txt.gz";
$zd = gzopen($filename, "r");
$contents = gzread($zd, 10000);
gzclose($zd);
?>
```

参考

- gzwrite()
- gzopen()
- gzgets()
- gzgetss()
- gzfile()
- gzpassthru()

---

gzrewind

(PHP 4, PHP 5)

gzrewind — gz ファイルポインタの示す位置を元に戻す

説明

bool **gzrewind** (resource $zp)

与えられた gz ファイルポインタが指すファイルのファイル位置記述子を、 ファイルストリームの先頭にセットします。

パラメータ

zp

gz ファイルポインタを指定します。これは有効なファイルポインタであり、 かつ、gzopen() によりオープンできたファイルを指している必要があります。

返り値

成功した場合に **TRUE** を、失敗した場合に **FALSE** を返します。

参考

- gzseek()
- gztell()

---

gzseek

(PHP 4, PHP 5)

gzseek — gz ファイルポインタの位置を移動する

説明

int **gzseek** (resource $zp , int $offset)

与えられたファイルポインタが指すファイルのファイル位置記述子を、 ファイルストリーム上の与えられた offset バイトにセットします。 gzseek(zp, offset, SEEK_SET) を (C 言語において) コールするのと同じです。

ファイルが読み込み用にオープンされた場合、この関数は、 エミュレーションされますが、極端に遅くなる可能性があります。 ファイルを書き込み用にオープンした場合、 前方への移動のみがサポートされます。この場合、**gzseek()** は、新しい開始位置までゼロの並びのデータを圧縮します。

パラメータ

zp

gz ファイルポインタを指定します。これは有効なファイルポインタであり、 かつ、gzopen() によりオープンできたファイルを指している必要があります。

offset

移動するオフセットを指定します。

返り値

成功した場合、0を返します。それ以外の場合は、-1を返します。 移動がEOFを超える場合にもエラーは発生しないことに注意してください。

例

Example#1 gzseek() の例

```
<?php
$gz = gzopen('somefile.gz', 'r');
gzseek($gz,2);
echo gzgetc($gz);
gzclose($gz);
?>
```

参考

- gztell()
- gzrewind()

---

gztell

(PHP 4, PHP 5)

gztell — gz ファイルポインタの読み込み/書き込み位置を返します

説明

int **gztell** (resource $zp)

与えられたファイルポインタが指す位置、つまり、 圧縮されていないファイルストリームにおけるオフセット値を返します。

パラメータ

zp

gz ファイルポインタを指定します。これは有効なファイルポインタであり、 かつ、gzopen() によりオープンできたファイルを指している必要があります。

返り値

*ファイルポインタの位置、もしくはエラーが発生した場合 **FALSE***

参考

- *gzopen()*
- *gzseek()*
- *gzrewind()*

---

gzuncompress

(PHP 4 >= 4.0.1, PHP 5)

gzuncompress — 圧縮された文字列を解凍する

説明

*string **gzuncompress** (string $data [, int $length])*

この関数は圧縮された文字列を解凍します。

パラメータ

data

gzcompress() によって圧縮されたデータを指定します。

length

デコードするデータの最大長を指定します。

返り値

*オリジナルの無圧縮なデータ、もしくはエラー時に **FALSE***

この関数は、もし無圧縮なデータが圧縮された入力 data の 32768 倍、もしくはオプションのパラメータ length 以上の場合、エラーを返します。

例

Example#1 gzuncompress() の例

```
<?php
$compressed   = gzcompress('Compress me', 9);
$uncompressed = gzuncompress($compressed);
echo $uncompressed;
?>
```

参考

- *gzcompress()*
- *gzinflate()*
- *gzdeflate()*
- *gzencode()*

---

gzwrite

(PHP 4, PHP 5)

gzwrite — バイナリセーフな gz ファイル書き込み

説明

*int **gzwrite** (resource $zp , string $string [, int $length])*

***gzwrite()** は string の内容を与えられた gz ファイルに書き込みます。*

パラメータ

zp

gz ファイルポインタを指定します。これは有効なファイルポインタであり、 かつ、gzopen() によりオープンできたファイルを指している必要があります。

string

書き込む文字列を指定します。

length

書き込む解凍されたバイト数を指定します。 もし指定された場合、 length バイトのデータが書き込まれたか、 string の終わりに達した時に 書き込みは終了します。

> ***注意**: 引数 length が指定された場合、 magic_quotes_runtime 設定オプションは無視されて string から スラッシュが取り除かれなくなることに注意してください。*

返り値

与えられた gz ファイルストリームに書き込まれた（解凍された）バイト数を返します。

例

Example#1 gzwrite() の例

```
<?php
$string = 'Some information to compress';
$gz = gzopen('somefile.gz','w9');
gzwrite($gz, $string);
gzclose($gz);
?>
```

参考

- gzread()
- gzopen()

---

readgzfile

(PHP 4, PHP 5)

readgzfile — gz ファイルを出力する

説明

*int **readgzfile** (string $filename [, int $use_include_path])*

ファイルを読み込み、解凍し、標準出力に書き込みます。

***readgzfile()**は、gzip フォーマットでないファイルの読込にも使用可能です。この場合、 **readgzfile()** はファイルを解凍せずに直接読込みます。*

パラメータ

filename

ファイル名を指定します。このファイルはファイルシステムからオープンされ、 内容は標準出力に書き込まれます。

use_include_path

include_path にあるファイルも検索したい場合、このオプションパラメータを 1 に設定してください。

返り値

*ファイルから読み込んだ（解凍された）データのバイト数を返します。 エラーが発生した場合、**FALSE** が返され、@readgzfile としてコールされている場合を除き、エラーメッセージが出力されます。*

参考

- gzpassthru()
- gzfile()
- gzopen()

---

zlib_get_coding_type

(PHP 4 >= 4.3.2, PHP 5)

zlib_get_coding_type — 出力圧縮に使用されたコーディングの種類を返す

説明

*string **zlib_get_coding_type** (void)*

出力圧縮に使用されたコーディングの種類を返します。

返り値

*返される値は gzip, deflate, もしくは **FALSE** です。*

参考

- zlib.output_compression ディレクティブ

目次

PHP のコア: Zend Engine ハッカーの手引き

目次

Zend API は、時を経て見違えるほどの成長を遂げました。 PHP がより堅牢なものとなり、幅広く使われるようになったのもそのおかげです。 PHP 5 とともに登場したのが Zend Engine 2 (ZE2) です。 ZE2 は、ほぼすべてを新たに書き直したもので、 新しいオブジェクト指向プログラミング (OOP) モデルを提供します。 また、多くの API が高速化されています。 PHP 6 は、このドキュメントの執筆時点ではまだ開発段階です。 PHP 6 で提供される Zend Engine 3 (ZE3) では、 完全な Unicode サポートが提供されるようになります。

警告

このドキュメントは、まだまだ完成には程遠いものです。 もともとあった Zend のドキュメントをそのまま Zend Engine 1 に残しているので、必要な方はそちらをご覧ください。

このセクションでは、ZE2 について扱います。 PHP 4.3 もいまだに幅広く使われていますが、 ZE1 と ZE2 で拡張モジュールの書き方が大幅に変わるわけではありません。 相違点については、このセクションの付録として簡単にまとめます。 ZE3 の API は今後も大きく変わる可能性があります。 ZE3 についても、別途付録で扱います。ZE3 のドキュメントは、 PHP 6 がベータテスト段階に達した時点できちんと仕上げる予定です。

このセクションのドキュメントの内容は、 執筆時の最新安定版である PHP 5.2.3 を基準としています。 PHP 5 の各マイナーリリース (5.0、5.1 そして 5.2) の間の相違点については、重要なものは適宜説明します。

PHP 5 ビルドシステム

目次

PHP 5 のさまざまな機能と柔軟性を考えると、それが何千ものファイルの 何百万行というコードから出来上がっていると聞いても驚かないことでしょう。 また、そのようなデータを管理するには 何らかのビルドシステムが必要となるのも無理もないことです。 このセクションでは、拡張モジュール開発用に PHP をセットアップする方法や PHP ソースツリー内における拡張モジュールの配置、 そして拡張モジュールをビルドする方法を説明します。

拡張モジュール開発用に PHP をビルドする方法

PHP を普通にインストールすると、 デバッグ機能よりもパフォーマンスの向上に主眼を置いて最適化された設定になります。 実際に運用することを考えると、これは妥当な選択でしょう。 しかし、拡張モジュールを開発する環境としてはあまりよくありません。 何か問題がおこったときに、どこが悪いのかを調べやすくするように PHP をビルドする必要があります。

Zend Engine ではメモリマネージャが提供されており、 拡張モジュール内で発生したメモリリークを追跡することができます。 また、詳細なデバッグ情報を取得することもできます。 しかし、この機能はデフォルトでは無効になっており、 スレッドセーフであることを優先しています。 この機能を使うには、configure のオプションに --enable-debug と --enable-maintainer-zts を追加します。 PHP をソースからビルドする方法については インストールにあたっての一般的な注意事項 の説明を参照ください。典型的な configure の設定は次のようになります。

```
$ ./configure --prefix=/where/to/install/php --enable-debug --enable-maintainer-zts --enable-cgi --enable-cli --with-mys
```

---

ext_skel スクリプト

Zend 拡張モジュールには、共通に含まれるファイルがいくつかあります。 その多くは拡張モジュールの種類に関わらずほぼ同じ内容ですが、 毎回それをコピーするのは面倒です。 幸いなことに、そのような初期作業を代わりに行ってくれるスクリプトがあります。 それが ext_skel で、 PHP 4.0 以降に同梱されています。

PHP 5.2.2 で、何もパラメータを指定せずに ext_skel を実行した結果は、次のようになります。

```
php-5.2.2/ext$ ./ext_skel
./ext_skel --extname=module [--proto=file] [--stubs=file] [--xml[=file]]
              [--skel=dir] [--full-xml] [--no-help]

   --extname=module    module is the name of your extension
   --proto=file         file contains prototypes of functions to create
   --stubs=file        generate only function stubs in file
   --xml                generate xml documentation to be added to phpdoc-cvs
   --skel=dir          path to the skeleton directory
   --full-xml           generate xml documentation for a self-contained extension
                       (not yet implemented)
   --no-help           don't try to be nice and create comments in the code
                        and helper functions to test if the module compiled
```

一般に、新しい拡張モジュールを開発する際に使うであろうパラメータは --extname と --no-help のみです。 拡張モジュールの構成に慣れるまでは、 --no-help を指定しないようにしましょう。 そうすることで、ext_skel が生成するファイルに有用なコメントがたくさん含められるようになります。

残ったオプションは --extname です。 これは、作成する拡張モジュールの名前を ext_skel に伝えるためのものです。ここで指定する名前は、すべて小文字からなるものです。 使用できる文字は英字とアンダースコアのみで、PHP の ext/ フォルダ配下で一意なものでなければなりません。

--proto オプションは、 作成したい PHP 関数の情報を含むヘッダファイルを指定するためのものです。 既存のライブラリを使用する拡張モジュールの開発時に使用できるということらしいですが、 最近のヘッダファイルではうまく機能しないことが多いようです。 試しに zlib.h を指定してみたところ、 ext_skel の出力に 空っぽで何の意味もないプロトタイプが大量に含まれてしまいました。 --xml と --full-xml は、まったく機能しません。--skel を使用すると、独自の雛形ファイル群を使用することができます。 このオプションは、このセクションでは対象外とします。

UNIX 用のビルドシステム: config.m4

拡張モジュールの config.m4 ファイルは、 UNIX のビルドシステムに対して「その拡張モジュールがサポートしている configure オプション」「依存する外部ライブラリ」 「コンパイル対象となるソースファイル」などの情報を伝えるものです。 一般的に用いられる autoconf マクロ (PHP 固有のものおよび autoconf 組み込みのもの) については Zend Engine 2 API リファレンス を参照ください。

ヒント

PHP の拡張モジュールを開発する際には、**autoconf** のバージョン 2.13 を使用することを 強く 推奨します (もっと新しいバージョンもありますが、それでもあえて 2.13 です)。 バージョン 2.13 は、(機能や使い勝手、利用者数など) いろんな意味で最も一般的な **autoconf** です。 これより新しいバージョンを使用すると、**configure** の内容が期待とは多少異なるものになるかもしれません。

Example#1 config.m4 ファイルの例

```
dnl $Id: configunix.xml,v 1.1 2007/11/17 08:05:35 takagi Exp $
dnl config.m4 for extension example

PHP_ARG_WITH(example, for example support,
[   --with-example[=FILE]          Include example support. File is the optional path to example-config])
PHP_ARG_ENABLE(example-debug, whether to enable debugging support in example,
[   --enable-example-debug           example: Enable debugging support in example], no, no)
PHP_ARG_WITH(example-extra, for extra libraries for example,
[   --with-example-extra=DIR         example: Location of extra libraries for example], no, no)

dnl Check whether the extension is enabled at all
if test "$PHP_EXAMPLE" != "no"; then

   dnl Check for example-config. First try any path that was given to us, then look in $PATH
    AC_MSG_CHECKING([for example-config])
    EXAMPLE_CONFIG="example-config"
   if test "$PHP_EXAMPLE" != "yes"; then
      EXAMPLE_PATH=$PHP_EXAMPLE
   else
```

```
    EXAMPLE_PATH=`$php_shtool path $EXAMPLE_CONFIG`
  fi

  dnl If a usable example-config was found, use it
  if test -f "$EXAMPLE_PATH" && test -x "$EXAMPLE_PATH" && $EXAMPLE_PATH --version > /dev/null 2>&1; then
    AC_MSG_RESULT([$EXAMPLE_PATH])
    EXAMPLE_LIB_NAME=`$EXAMPLE_PATH --libname`
    EXAMPLE_INCDIRS=`$EXAMPLE_PATH --incdirs`
    EXAMPLE_LIBS=`$EXAMPLE_PATH --libs`

    dnl Check that the library works properly
     PHP_CHECK_LIBRARY($EXAMPLE_LIB_NAME, example_critical_function,
   [
      dnl Add the necessary include dirs
       PHP_EVAL_INCLINE($EXAMPLE_INCDIRS)
      dnl Add the necessary libraries and library dirs
       PHP_EVAL_LIBLINE($EXAMPLE_LIBS, EXAMPLE_SHARED_LIBADD)
   ],[
     dnl Bail out
      AC_MSG_ERROR([example library not found. Check config.log for more information.])
    ],[$EXAMPLE_LIBS]
   )
  else
    dnl No usable example-config, bail
    AC_MSG_RESULT([not found])
    AC_MSG_ERROR([Please check your example installation.])
  fi

  dnl Check whether to enable debugging
  if test "$PHP_EXAMPLE_DEBUG" != "no"; then
    dnl Yes, so set the C macro
    AC_DEFINE(USE_EXAMPLE_DEBUG,1,[Include debugging support in example])
  fi

  dnl Check for the extra support
  if test "$PHP_EXAMPLE_EXTRA" != "no"; then
    if test "$PHP_EXAMPLE_EXTRA" == "yes"; then
      AC_MSG_ERROR([You must specify a path when using --with-example-extra])
   fi

     PHP_CHECK_LIBRARY(example-extra, example_critical_extra_function,
   [
     dnl Add the neccessary paths
      PHP_ADD_INCLUDE($PHP_EXAMPLE_EXTRA/include)
       PHP_ADD_LIBRARY_WITH_PATH(example-extra, $PHP_EXAMPLE_EXTRA/lib, EXAMPLE_SHARED_LIBADD)
      AC_DEFINE(HAVE_EXAMPLEEXTRALIB,1,[Whether example-extra support is present and requested])
      EXAMPLE_SOURCES="$EXAMPLE_SOURCES example_extra.c"
   ],[
     AC_MSG_ERROR([example-extra lib not found. See config.log for more information.])
    ],[-L$PHP_EXAMPLE_EXTRA/lib]
   )
  fi

  dnl Finally, tell the build system about the extension and what files are needed
   PHP_NEW_EXTENSION(example, example.c $EXAMPLE_SOURCES, $ext_shared)
   PHP_SUBST(EXAMPLE_SHARED_LIBADD)
fi
```

autoconf の構文についての簡単な説明

config.m4 は、GNU **autoconf** の構文で書かれています。簡単にいうと、これはシェルスクリプトに強力なマクロ言語を追加したようなものです。コメントを記述する際には文字列 dnl を使用し、文字列のクォートには角括弧（[および]）を使用します。文字列のクォートは、必要に応じて何段階でもネストすることができます。この構文の完全な解説は、» http://www.gnu.org/software/autoconf/manual/ にある **autoconf** のマニュアルを参照ください。

PHP_ARG_*: ユーザに対するオプションの提供

先ほどの config.m4 のサンプルで、ちょっとしたコメントを除いていちばん最初にあるのは **PHP_ARG_WITH()** および **PHP_ARG_ENABLE()** を使用した 3 行です。これらは、**configure** コマンドで **./configure --help** を実行したときに表示される オプションとその説明文を指定するものです。その名が示すように、両者の違いはそれぞれ --with-* 系のオプションを作成するか --enable-* 系のオプションを作成するかです。すべての拡張モジュールは、このどちらかに拡張モジュール名とつけたオプションを提供する必要があります。これにより、その拡張モジュールを PHP に組み込むかどうかを選択できるようになります。規約上では、何らかのパラメータ（その拡張モジュールが必要とするライブラリやプログラムの場所など）を指定させる場合には **PHP_ARG_WITH()** を使うことになっています。一方、単なるフラグとして使用するオプションの場合は **PHP_ARG_ENABLE()** を使用します。

Example#2 configure の出力例

```
$ ./configure --help
...
  --with-example[=FILE]         Include example support. FILE is the optional path to example-config
  --enable-example-debug          example: Enable debugging support in example
  --with-example-extra=DIR        example: Location of extra libraries for example
...

$ ./configure --with-example=/some/library/path/example-config --disable-example-debug --with-example-extra=/another/lib
...
checking for example support... yes
checking whether to enable debugging support in example... no
checking for extra libraries for example... /another/library/path
...
```

> **注意**: **configure** のコール時にどんな順でオプションを指定したかにかかわらず、内部では config.m4 で指定した順にチェックを行います。

ユーザの選択内容の処理

ここまでで、config.m4 を使ってユーザに対して選択肢を提供できるようになりました。 次は、選択内容に応じて実際の処理を行う番です。 上の例で、3 つのオプションすべてのデフォルト、つまり何も指定しなかったときの値は "no" です。拡張モジュールを有効にするオプションでは、 デフォルトを no にしておくのがおすすめです。 phpize で個別にビルドするときにはこれは上書きされますし、 PHP に組み込む際にデフォルトで拡張モジュールの空間を乱してはいけないからです。 これらの 3 つのオプションを処理するコードは、より複雑なものとなります。

--with-example[=FILE] オプションの処理

最初のチェック、つまり --with-example[=FILE] オプションのチェックは、それが設定されているかどうかを調べます。 このオプションは拡張モジュールそのものを組み込むかどうかを決めるものです。 省略されていたり否定形式 (--without-example) で指定されていたり、あるいは値として "no" が指定されていたりした場合は、それ以降は何も行いません。 上の例では、値として /some/library/path/example-config が指定されているので、このチェックは成功します。

次に、このコードは AC_MSG_CHECKING() をコールします。 これは autoconf のマクロで、標準的な "checking for something" の行を出力してユーザが example-config にパスを明示したかどうかを調べます。 この例では PHP_EXAMPLE は値 /some/library/path/example-config を受け取り、 それが変数 EXAMPLE_PATH にコピーされます。ユーザが --with-example だけしか指定しなかった場合は、 このコードは $php_shtool path $EXAMPLE_CONFIG を実行します。これは、そのユーザの現在の PATH を使用して example-config の場所を探します。 どちらにしても、次に行うのは EXAMPLE_PATH が通常の実行可能ファイルであるかどうか、 そして正常に実行できるかどうかの調査となります。 実行できた場合は AC_MSG_RESULT() がコールされ、 AC_MSG_CHECKING() で始まる出力行を補完します。 それ以外の場合は AC_MSG_ERROR() がコールされ、 指定されたメッセージを表示して configure を即時に終了させます。

次に、このコードは example-config を何度か実行してサイト固有の設定情報を取得します。 その次にコールするのは PHP_CHECK_LIBRARY() です。これは PHP のビルドシステムが autoconf の AC_CHECK_LIB() のラッパーとして用意しているマクロです。 PHP_CHECK_LIBRARY() は、 最初のパラメータで指定したライブラリの 2 番目のパラメータで指定したシンボルをコールするプログラムを コンパイル、リンクして実行します。 成功すると、3 番目のパラメータで指定したスクリプトを実行します。 このスクリプトは PHP ビルドシステムに対して インクルードパスやライブラリパス、ライブラリ名を example-config が返す文字列から通知します。 失敗すると、かわりに 4 番目のパラメータで指定したスクリプトを実行します。 この場合は、AC_MSG_ERROR() がコールされて処理を停止します。

--enable-example-debug オプションの処理

--enable-example-debug の処理はずっと単純です。真値が設定されているかどうかだけを調べます。 チェックに成功すると、AC_DEFINE() をコールして C のマクロ USE_EXAMPLE_DEBUG を作成し、拡張モジュールのソースで使用できるようにします。 3 番目のパラメータは config.h 用のコメント文字列です。 これを空のままにしておいても問題はありません。たいていは空のままにします。

--with-example-extra=DIR オプションの処理

このサンプルでの説明用に用意した架空の機能 "extra" は --with-example-extra=DIR オプションで要求されるもので、架空のプログラム example-config とは別のものであり、 デフォルトのサーチパスもありません。 そのためユーザは、必要なライブラリのインストール先を指定する必要があります。 このような設定は実際の拡張モジュールではあまりないかもしれませんが、 説明用として用意しました。

このコードは、あまりなじみのない方法で PHP_EXAMPLE_EXTRA が真値であるかどうかを調べます。 否定形が指定された場合は、それ以降の処理を行いません。 これは、そのユーザが追加機能を必要としなかったことを表します。 パラメータなしで肯定形が指定された場合は、 AC_MSG_ERROR() がコールされて処理を停止します。 その次に、また PHP_CHECK_LIBRARY() を実行します。 今度は、定義済みのコンパイラオプションはないので、PHP_ADD_INCLUDE() と PHP_ADD_LIBRARY_WITH_PATH() を使用して必要なインクルードパスやライブラリパス、 追加機能のためのライブラリのフラグを作成します。 AC_DEFINE() もコールされ、 追加機能が要求されていてそれが使用可能であることを通知します。 また、ビルドすべきファイルが別に存在することを示す変数も設定します。 チェックに失敗すると、おなじみの AC_MSG_ERROR() がコールされます。 失敗時のもうひとつの処理法は、次のように AC_MSG_WARNING() をコールすることです。

```
AC_MSG_WARNING([example-extra lib not found. example will be built without extra functionality.])
```

この場合は、configure はエラーではなく警告メッセージを表示し、処理を続行します。 いずれにせよ、失敗をどう処理するのかは拡張モジュールの開発者が決めることです。

ビルドシステムに対しての決定内容の通知

必要なインクルードファイルやライブラリはすべて指定しました。 必要なオプションやマクロもすべて定義しました。 でも、あとひとつ残っていることがあります。 ビルドシステムに対して、拡張モジュール自身をビルドすることと そのために使用するファイル群を教えてやらなければならないのです。 そのためには PHP_NEW_EXTENSION() マクロをコールします。 このマクロの最初のパラメータは拡張モジュールの名前で、 これはディレクトリ名と一致します。2 番目のパラメータは、 その拡張モジュールを構成するすべてのソースファイルのリストです。 サブディレクトリ内のソースファイルをビルド処理に追加する方法については PHP_ADD_BUILD_DIR() を参照ください。 3 番目のパラメータは、常に $ext_shared でなければなりません。この値は、 configure の際に --with-example[=FILE] 用に PHP_ARG_WITH() がコールされたときに決まります。 4 番目のパラメータでは "SAPI クラス" を指定します。 これは CGI SAPI や CLI SAPI を必要とする場合にのみ有用です。 その他のクラスでは、ここは空のままにしておかなければなりません。 5 番目のパラメータには、拡張モジュールのビルド時の CFLAGS に追加するフラグのリストを指定します。 6 番目のパラメータは boolean 値です。"yes" を指定すると、 拡張モジュール全体のビルドを $CC ではなく $CXX を用いて行います。 3 番目以降のパラメータは、すべてオプションです。 最後に PHP_SUBST() をコールして、 拡張モジュールの共有ビルドを有効にします。 共有モードでの拡張モジュールのビルドをサポートしないようにする方法についての詳細は 拡張モジュールに関する FAQ を参照ください。

Windows 用のビルドシステム: config.w32

拡張モジュールに含まれている config.w32 ファイルの使用法は config.m4 とほぼ同じですが、 決定的な違いがふたつあります。まず第一に、これは Windows 版のビルド用に使うものです。 次に、これは JavaScript で書かれています。 このセクションでは、JavaScript の文法については特に扱いません。 現時点では、このセクションは未完成です。とりあえずのたたき台として用意したものであり、 ごらんいただけるのは config.m4 のサンプルだけです。

Example#1 config.w32 ファイルのサンプル

```
// $Id: configwin.xml,v 1.3 2007/07/02 14:56:14 takagi Exp $
//  vim:ft=javascript

ARG_WITH("example", "for example support", "no");
ARG_ENABLE("example-debug", "for debugging support in example", "no")
ARG_WITH("example-extra", "for extra functionality in example", "no")
if (PHP_EXAMPLE != "no") {
      if (CHECK_LIB("libexample.lib", "example", PHP_EXAMPLE) &&
            CHECK_HEADER_ADD_INCLUDE("example.h", "CFLAGS_EXAMPLE", PHP_EXAMPLE + "¥¥include")) {

          if (PHP_EXAMPLE_DEBUG != "no") {
                AC_DEFINE('USE_EXAMPLE_DEBUG', 1, 'Debug support in example');
        }

          if (PHP_EXAMPLE_EXTRA != "no" &&
                CHECK_LIB("libexample-extra.lib", "example", PHP_EXAMPLE) &&
                 CHECK_HEADER_ADD_INCLUDE("example-extra.h", "CFLAGS_EXAMPLE", PHP_EXAMPLE + ";" + PHP_PHP_BUILD + "¥¥include"
                AC_DEFINE('HAVE_EXAMPLEEXTRA', 1, 'Extra functionality in example');
            HAVE_EXTRA = 1;
        } else {
```

```
                WARNING( "extra example functionality not enabled, lib not found" );
        }

          EXTENSION("example", "example.c");
        if (HAVE_EXTRA == 1) {
                ADD_SOURCES("example-extra.c");
        }
    } else {
          WARNING( "example not enabled; libraries not found" );
    }
}
```

拡張モジュールの構造

目次

拡張モジュールの作成方法を説明した文書の多くは、 「まずはシンプルな例からはじめましょう」式のものであり、 実際に複雑なものを実装していくにあたって必要となる箇所の多くを省略しています。 その結果、新しい機能を説明するたびに これまでの内容を何度も何度も繰り返すというはめに陥ってしまっています。 このセクションでは、そのようなものではなく、 実用に耐えうるレベルに作りこんだ拡張モジュールの構造を説明していきます。 実際に拡張モジュールを作成するにあたって出会うであろうさまざまな問題への対処法もわかることでしょう。

拡張モジュールを構成するファイル群

手作業で作ったか **ext_skel** を使用したか、 あるいは » CodeGen のようなツールを用いたのかにかかわらず、 すべての拡張モジュールには少なくとも次の 4 つのファイルが含まれます。

config.m4

UNIX ビルドシステムの設定ファイル (UNIX 用のビルドシステム: config.m4 を参照ください)。

config.w32

Windows ビルドシステムの設定ファイル (Windows 用のビルドシステム: config.w32 を参照ください)。

php_example.h

拡張モジュールを PHP バイナリに静的モジュールとして組み込む場合は、 php_ の後に拡張モジュールの名前を続けたヘッダファイルに その拡張モジュールの構造についての宣言があるものとします。 通常は、このファイルに記述するのは、 マクロやプロトタイプ宣言、グローバル変数といった他のヘッダと同じ内容です。

example.c

拡張モジュールのソースファイル。 規約上は拡張モジュールの名前と同じ名前にすることになっていますが、 これは必須ではありません。このファイルに含まれる内容は、 モジュールの構造体定義や INI エントリ、 管理用関数、ユーザに公開する関数、その他拡張モジュールに必要となるものです。

ビルドシステム関連のファイルについては別の場所で説明することにして、 ここでは残りのファイルについて取り上げます。 これらの 4 つのファイルはあくまでも必要最小限のものであり、 実際にはこれら以外にもさまざまなヘッダファイルやソースファイル、 ユニットテストなどのファイルが含まれることになるでしょう。 より実情に近いファイル一覧は、このようになります。

Example#1 より "現実的な" 拡張モジュールのファイル一覧 (順不同)

```
ext/
  example/
    .cvsignore
    config.m4
    config.w32
    example_util.h
    example_util.c
    php_example.h
    example.c
    package.xml
    CREDITS
    tests/
      critical_function_001.phpt
      critical_function_002.phpt
      optional_function_001.phpt
      optional_function_002.phpt
```

ソース以外のファイル

作成した拡張モジュールを PHP の **CVS** リポジトリのどこか (通常は » PECL) にチェックインする際には、 .cvsignore というファイルを使用します。 **ext_skel** が作成するこのファイルの雛形は、 次のようになります。

```
.deps
*.lo
*.la
```

これは、PHP のビルドシステムが作成する中間ファイルを無視するよう **CVS** に指示するためのものです。 単に利便性のためだけに存在するものであり、 省略しても悪影響はありません。

CREDITS ファイルには、 その拡張モジュールの開発に協力してくれた人たちやメンテナの情報を プレーンテキストで記述します。このファイルの主

要な目的は、 バンドルされている拡張モジュールが phpcredits() で使用する情報を作成することにあります。 このファイルの最初の行には拡張モジュールの名前、 そしてその次の行には協力者をカンマ区切りで指定します。 協力者の一覧は、通常は開発に参加した順に並べます。 » PECL パッケージでは、これらの情報はすでに package.xml などで管理されています。 ですので、このファイルを省略しても特に問題はありません。

package.xml ファイルは、» PECL 拡張モジュールに特有のものです。これはメタ情報ファイルであり、 その拡張モジュールの依存性や作者、インストール要件などの情報を含みます。 拡張モジュールを » PECL で公開するつもりがないのであれば、 このファイルは不要です。

---

zend_module 構造体

PHP 拡張モジュールのソースファイルの中には、 C プログラマにとって目新しいものがいくつか含まれています。 これらの中でも最も重要であり、拡張モジュールを開発するにあたって最初にさわることになるのが zend_module 構造体です。 この構造体には豊富な情報が格納されており、 その拡張モジュールの依存性やバージョン、コールバック、 その他重要なデータを Zend Engine に伝える役割を果たします。 この構造体の中身は、何度も大幅に変更されています。 ここでは、PHP 5.0 の時点の情報をもとにして説明します。 PHP 5.1 や 5.2 では、少々変更されている点もあります。

example.c での zend_module の宣言は、次のようになります (これは、 **ext_skel --extname=example** で生成したものに何も手を加えていない状態です)。

Example#1 example 拡張モジュールにおける zend_module の宣言部

```
/* {{{ example_module_entry
 */
zend_module_entry example_module_entry = {
#if ZEND_MODULE_API_NO >= 20010901
    STANDARD_MODULE_HEADER,
#endif
    "example",
    example_functions,
    PHP_MINIT(example),
    PHP_MSHUTDOWN(example),
    PHP_RINIT(example),          /* Replace with NULL if there's nothing to do at request start */
    PHP_RSHUTDOWN(example),     /* Replace with NULL if there's nothing to do at request end */
    PHP_MINFO(example),
#if ZEND_MODULE_API_NO >= 20010901
    "0.1", /* Replace with version number for your extension */
#endif
    STANDARD_MODULE_PROPERTIES
};
/* }}} */
```

最初はちょっとひるむかも知れませんが、 大半の部分はよく見れば非常に単純です。 次に示すのは、PHP 5.2 の zend_modules.h における zend_module の宣言部です。 関連する定数の定義も含めています。

Example#2 PHP 5.2 における zend_module の定義

```
#define ZEND_MODULE_API_NO 20060613
struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    struct _zend_ini_entry *ini_entry;
    struct _zend_module_dep *deps;
    char *name;
    struct _zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    char *version;
    size_t globals_size;
#ifdef ZTS
    ts_rsrc_id* globals_id_ptr;
#else
    void* globals_ptr;
#endif
    void (*globals_ctor)(void *global TSRMLS_DC);
    void (*globals_dtor)(void *global TSRMLS_DC);
    int (*post_deactivate_func)(void);
    int module_started;
    unsigned char type;
    void *handle;
    int module_number;
};
#define STANDARD_MODULE_HEADER_EX sizeof(zend_module_entry), ZEND_MODULE_API_NO, ZEND_DEBUG, USING_ZTS
#define STANDARD_MODULE_HEADER ¥
    STANDARD_MODULE_HEADER_EX, NULL, NULL
#define ZE2_STANDARD_MODULE_HEADER ¥
    STANDARD_MODULE_HEADER_EX, ini_entries, NULL

#define STANDARD_MODULE_PROPERTIES_EX 0, 0, NULL, 0
```

拡張モジュールのグローバル変数

拡張モジュールのライフサイクル

拡張モジュールのテスト

メモリ管理

変数の作成

関数の作成

PHP はグルー言語（訳注: 異なるシステムをつなぎ合わせるための "接着材" としてはたらく言語）としても知られており、 ジェネレータを使用して簡単に拡張することができます。 ext_skel を使って C 言語の関数のスタブを作成すると、 エクスポートされるすべての関数が PHP_FUNCTION(func_name) のようなシンプルなプロトタイプであることに気づかれるでしょう。

クラスやオブジェクトの作成

リソースの作成

INI 設定の作成

ストリームの作成

PDO ドライバ How-To

目次

この How-To の目的は、PDO 用レイヤーとのインターフェイスとなる データベースドライバを書くために必要な基礎知識を身につけることです。 この API はまだ発展途上のものであり、変更される可能性があることに注意しましょう。 このドキュメントは、PDO のバージョン 0.3 をもとにしています。 学習曲線はかなりの急勾配です。まず最初の段階を身につけるのにかなりの時間を要するでしょう。

要件

PDO データベースドライバを書くために必要な条件は、以下のとおりです。

1. 対象となるデータベースの稼働環境・例・デモなど、ベンダの仕様どおりに 動作するもの。
2. 開発環境
    1. その他の Unix: ベンダが提供する標準開発ツールに加え、 GNU の開発ツールセット
    2. Linux: 標準的な開発ツール、gcc・ld・make・autoconf・automake など…。ディストリビューションに応じたバージョンのもの
    3. Win32: Visual Studio コンパイラ
3. PHP バージョン 5.0.3 以降および PEAR バージョン 1.3.5 以降の環境
4. PDO の動作環境 ('sudo pecl install PDO' を使用してインストール できます)。ここには PDO の型定義や関数定義にアクセスするための ヘッダが含まれます
5. C 言語についての十分な知識
6. PHP の拡張モジュールを書く方法についての十分な知識。 George Schlossnagle の Advanced PHP Programming (Developer's Library 発行、第 21 章および第 22 章) がお勧めです。
7. 最後に、PHP の心臓部である Zend API に（特にメモリ管理の面で）習熟していること。

準備

ソースディレクトリの構造

典型的な PDO ドライバのソースディレクトリは以下のようになっています。 ここで、SKEL はドライバが接続しようとしている データベース名の短縮形式を表します。ここでは SKEL が大文字で 表記されていますが(存在をはっきりさせるためです)、実際は 小文字を使用するのが慣例です。

```
pdo_SKEL/
   config.m4                    # unix build script
   config.w32                   # win32 build script
    CREDITS
   package.xml                   # meta information about the package
   pdo_SKEL.c                    # standard PHP extension glue
    php_pdo_SKEL.h
   php_pdo_SKEL_int.h           # driver private header
   SKEL_dbh.c                     # contains the implementation of the PDO driver interface
   SKEL_stmt.c                    # contains the implementation of the PDO statement interface
   tests/
```

これらのファイルの内容については、この文書の後半で説明します。

雛形の作成

開発を始めるにあたっての最もお手軽な方法は、PHP ビルドツリーの ext ディレクトリにあるシェルスクリプト **ext_skel** を使用することです。このスクリプトは、上で挙げた多くのファイルを含む雛形ディレクトリを 作成します。スクリプトは、ext ディレクトリの 中で以下のように実行します。

```
./ext_skel  --extname=pdo_SKEL
```

これにより pdo_SKEL というディレクトリが作成され、その中に これから書き換えていく元となるファイルが作成されます 次に、このディレクトリを PHP のエクステンションディレクトリに 移動します。PDO は PECL 拡張モジュールなので、標準的な エクステンションディレクトリには含まれません。 PHP および PDO がインストールされていれば、どのディレクトリからでも ビルドすることが可能です。

標準的な include ファイル

ビルド固有のヘッダ

configure の際に、ドライバのビルドに使用するプラットフォーム固有の ヘッダファイル config.h が作成されます。このヘッダが存在する場合、コンパイラ変数 HAVE_CONFIG_H が設定されます。コンパイル時には この変数が存在するかどうかが調べられ、もし設定されていれば config.h が include されます。

PHP のヘッダ

各ソースモジュールには、以下の標準 PHP ヘッダが include されます。

1. php.h
2. php_ini.h
3. ext/standard/info.h

PDO インターフェイスのヘッダ

各ソースモジュールには、以下の標準 PDO ヘッダファイルも include されます。

pdo/php_pdo.h

このヘッダファイルには、ドライバの初期化やシャットダウンのための 関数の定義やグローバル PDO 変数の定義が含まれます。

pdo/php_pdo_driver.h

このヘッダには、PDO ドライバを書く際に使用する型や API の規約が 含まれます。PDO レイヤへのコールバックの定義、PDO にドライバを 登録したり登録を解除したりする関数の定義も含まれます。 最も重要なものとして、このヘッダファイルには PDO データベース ハンドルやステートメントの型定義が含まれています。 ドライバが主に使用する 2 つの構造体 pdo_dbh_t および pdo_stmt_t については、それぞれ 付録 A および B で詳細に説明します。

ドライバ固有のヘッダ

典型的な PDO ドライバは 2 つのヘッダファイルを保持しており、ここに データベースの実装に依存する内容が記述されます。実装内容によって これが変わることがあってもかまいませんが、規約では以下の 2 つの ヘッダが標準となっています。

php_pdo_SKEL.h

このヘッダファイルは、先ほど挙げた pdo/php_pdo.h の機能と内容を 完全に複製したもので、データベースに応じて変更していきます。 ドライバがグローバル変数を使用する場合、マクロ ZEND_BEGIN_MODULE_GLOBALS および ZEND_END_MODULE_GLOBALS を 使用してそれを定義しておく必要があります。これらの変数に アクセスする際に、マクロが使用されます。このマクロは 通常は PDO_SKEL_G(v) という名前で、v がアクセスされるグローバル変数となります。 詳細な情報は、Zend のプログラマ向けドキュメントを参照ください。

php_pdo_SKEL_int.h

このヘッダファイルには、ドライバの実装固有の型定義や関数宣言が 含まれます。また、データベース固有の構造体 pdo_SKEL_handle および pdo_SKEL_stmt の定義も含みます。これらは private 構造体で、ハンドル構造体やステートメント構造体のメンバ driver_data として参照されます。

オプションのヘッダ

ドライバの実装方法によっては、以下のヘッダを include する 必要があるかもしれません。

```
#include  <zend_exceptions.h>
```

雛形への肉付け

主要な構造体および属性

主要な構造体である pdo_dbh_t および pdo_stmt_t については、それぞれ 付録 A および B で説明します。データベースやステートメントの属性に ついては 付録 C、エラー処理については付録 D で説明します。

pdo_SKEL.c: PHP 拡張モジュールとの橋渡し

関数エントリ

```
static  function_entry  pdo_SKEL_functions[]  =  {
   { NULL, NULL, NULL }
};
```

この構造体は、グローバルな PHP 関数名前空間に関数を登録するために 使用されます。PDO ドライバでは、できるだけ使用を避けるべきです。 上の例で示しているように、NULL で初期化した状態にしておくことを 推奨します。

モジュールエントリ

```
/*  {{{  pdo_SKEL_module_entry  */
#if  ZEND_EXTENSION_API_NO  >=  220050617
static  zend_module_dep  pdo_SKEL_deps[]  =  {
      ZEND_MOD_REQUIRED("pdo")
     {NULL, NULL, NULL}
};
#endif
/*  }}}  */

zend_module_entry  pdo_SKEL_module_entry  =  {
#if  ZEND_EXTENSION_API_NO  >=  220050617
      STANDARD_MODULE_HEADER_EX,  NULL,
     pdo_SKEL_deps,
#else
      STANDARD_MODULE_HEADER,
#endif
     "pdo_SKEL",
      pdo_SKEL_functions,
      PHP_MINIT(pdo_SKEL),
      PHP_MSHUTDOWN(pdo_SKEL),
    NULL,
    NULL,
      PHP_MINFO(pdo_SKEL),
       PHP_PDO_<DB>_MODULE_VERSION,
      STANDARD_MODULE_PROPERTIES
};
/*  }}}  */

#ifdef  COMPILE_DL_PDO_<DB>
ZEND_GET_MODULE(pdo_db)
#endif
```

pdo_SKEL_module_entry という名前で zend_module_entry 型の構造体を宣言し、先ほど定義した pdo_SKEL_functions テーブルへの参照を含める必要があります。

標準 PHP 拡張モジュール関数

PHP_MINIT_FUNCTION

```
/*  {{{  PHP_MINIT_FUNCTION  */
PHP_MINIT_FUNCTION(pdo_SKEL)
{
      return  php_pdo_register_driver(&pdo_SKEL_driver);
}
/*  }}}  */
```

この標準 PHP 拡張モジュール関数は、ドライバを PDO に登録するために 使用されます。登録するには、**php_pdo_register_driver()** 関数に pdo_driver_t 型の構造体へのポインタを渡して コールします。この構造体の名前は、一般的には pdo_SKEL_driver となります。 pdo_driver_t には、マクロ PDO_DRIVER_HEADER(SKEL) を使用して生成した ヘッダおよび **pdo_SKEL_handle_factory()** 関数へのポインタが含まれます。実際の関数については SKEL_dbh.c の説明の中で述べます。

PHP_MSHUTDOWN_FUNCTION

```
/*  {{{  PHP_MSHUTDOWN_FUNCTION  */
PHP_MSHUTDOWN_FUNCTION(pdo_SKEL)
{
      php_pdo_unregister_driver(&pdo_SKEL_driver);
     return  SUCCESS;
}
/*  }}}  */
```

この標準 PHP 拡張モジュール関数は、ドライバを PDO から登録解除する ために使用されます。解除するには、 **php_pdo_unregister_driver()** 関数に 上で渡したのと同じ構造体 pdo_SKEL_driver を渡してコールします。

PHP_MINFO_FUNCTION

これもまた標準 PHP 拡張モジュール関数です。この関数の目的は、 スクリプト内で phpinfo() がコールされた際に モジュールの情報を表示するこ

とです。規約では、モジュールの バージョンおよび対応するデータベースのバージョン、そして 関連する設定情報を表示することになっています。

SKEL_driver.c: ドライバの実装

このファイルでは、PDO データベースハンドルオブジェクトがサポートする すべてのデータベース処理メソッドを実装します。また、エラー情報の取得ルーチンもここに含まれます。これらの関数では、グローバル変数 プールへのアクセスが必要になることでしょう。そのため、これらの ステートメントの最後には、Zend マクロ TSRMLS_DC を使用する必要が あります。このマクロについての詳細な情報は、Zend の プログラマ向けドキュメントを参照ください。

pdo_SKEL_error

```
static int pdo_SKEL_error(pdo_dbh_t *dbh,
   pdo_stmt_t *stmt, const char *file, int line TSRMLS_DC)
```

この関数の目的は、ドライバ内での一般的なエラー処理関数として使用する ことです。ドライバ内でエラーが発生した場合に、ドライバによって この関数がコールされます。SQLSTATE に関連しないエラーが発生した場合、 ドライバはエラーの内容にもっとも近い SQLSTATE あるいは一般的な SQLSTATE エラー "HY000" を、 dbh->error_code あるいは stmt->error_code にセットする必要があります。 PDO ソース内のファイル pdo_sqlstate.c には、PDO が明示的に理解する 一般的な SQLSTATE コードのテーブルがあります。エラーコードの設定は、 この関数がコールされる前に終わっていなければなりません。この関数は、 グローバル変数 pdo_err に dbh あるいは stmt (stmt が NULL でない場合) で見つかったエラーを設定します。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

stmt

現在のステートメントへのポインタあるいは NULL。 NULL の場合、エラーには dbh で見つかったエラーコードが設定されます。

file

エラーが発生したソースファイルあるいは取得できない場合は NULL。

line

取得可能な場合に、ソースファイル内の行番号。

dbh メンバメソッドが NULL の場合 (PDO コンストラクタ内でエラーが 発生したことを意味します)、この関数は zend_throw_exception_ex() 関数をコールしなければなりません。それ以外の場合はエラーコードを 返します。通常、この関数はヘルパマクロを使用してコールされます。 このマクロは、データベース処理エラーおよびステートメント処理エラーの それぞれについて関数のコール手順をカスタマイズしたものです。

Example#1 pdo_SKEL_error をコールするマクロの例

```
#define pdo_SKEL_drv_error(what) ¥
      pdo_SKEL_error(dbh, NULL, what, __FILE__, __LINE__ TSRMLS_CC)
#define pdo_SKEL_drv_error(what) ¥
      pdo_SKEL_error(dbh, NULL, what, __FILE__, __LINE__ TSRMLS_CC)
```

エラー処理についての詳細は、エラー処理 を参照ください。

> **注意**: このように記述されていますが、PDO ドライバインターフェイスでは特に この関数が存在することを指定していません。これは単にエラー処理を 便利にするための方法であり、大半のデータベースのクライアントライブラリ API ではこの方法でドライバを実装すると好都合です。

pdo_SKEL_fetch_error_func

```
static int pdo_SKEL_fetch_error_func(pdo_dbh_t *dbh, pdo_stmt_t *stmt,
     zval *info TSRMLS_DC)
```

この関数の目的は、直近に発生したエラーについての追加情報を 取得することです。ここには、ドライバ固有のエラーコードや 人間が理解できる形式のメッセージが含まれます。また、必要に応じて それ以外の追加情報も含められます。この関数は、PHP スクリプトで **PDO::errorInfo()** メソッドをコールした際に 呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

stmt

現在のステートメントへのポインタあるいは NULL。NULL の場合は dbh で見つかったエラーコードから情報を取得します。

info

エラーコードおよびメッセージを含むハッシュテーブル。

error_func は、情報を 2 つの部分に分けた上で、配列の連続する要素として 返さなければなりません。最初の要素は数値形式のエラーコードで、次の 項目が文字列の説明となります。この項目を設定する方法としては add_next_index を使用するのが最適です。最初の要素の型は long である必要がないことに注意しましょう。元になるデータベース API が返すエラーコードに応じた型を選びます。

```
/* ここではエラー情報を追加します。 */
/* 指定した順に追加する必要があります。 */
add_next_index_long(info, error_code);    /* ドライバ固有のエラーコード */
add_next_index_string(info, message, 0); /* 可読形式のエラーメッセージ */
```

この関数は、情報が取得可能な場合に 1、ドライバが追加情報を 保持していない場合に 0 を返します。

SKEL_handle_closer

```
static int SKEL_handle_closer(pdo_dbh_t *dbh TSRMLS_DC)
```

この関数は、オープンしているデータベースを閉じるために PDO からコールされます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

オープン中のデータベースを閉じるために必要な処理は、すべてここで 済ませる必要があります。PDO は、この関数の返す値を無視します。

SKEL_handle_preparer

```
static int SKEL_handle_preparer(pdo_dbh_t *dbh, const char *sql,
long sql_len, pdo_stmt_t *stmt, zval *driver_options TSRMLS_DC)
```

この関数は、PHP スクリプトで **PDO::query()** および **PDO::prepare()** がコールされた場合に PDO から呼び出されます。この関数の目的は、 実行する SQL を準備し、渡された stmt に しかるべき値を格納することです。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

sql

準備される SQL ステートメントを含む文字列へのポインタ。

sql_len

SQL ステートメントの長さ。

Stmt

返される statement へのポインタか、エラーが発生した場合には NULL。

driver_options

ドライバ固有の（ドライバで定義した）オプション。

この関数は、本質的には stmt オブジェクトのコンストラクタです。 ステートメントのオプションを処理し、ドライバ固有のオプションを pdo_stmt_t 構造体に格納することがこの関数の役割となります。

prepare 関数がコールされる前に、PDO がドライバの代わりにオプションを 処理してくれることはありません。未知のオプションが渡された際に エラーを発生させるなどの処理は、あなた（ドライバ）の役割となります。

この関数の非常に重要な役割のひとつは、SQL ステートメントのパラメータを 処理することです。この関数をコールした際に、PDO は 「ドライバがプリペアドステートメントへのパラメータのバインドをサポート している か」や「名前で指定するパラメータあるいは位置で指定するパラメータの どちらをサポートしているか」を知りません。

元となるデータベースにあわせて、ドライバが適切に stmt->supports_placeholders を設定しなければなりません。 接続先のサーバのバージョンによってこの設定が変化するなどの理由で、 この設定を実行時に行わなければならないこともあるかもしれません。 ドライバが、名前で指定するパラメータ・位置で指定するパラメータのいずれも サポートしていない場合は、**pdo_parse_params()** API を使用して PDO にクエリを書き換えさせることでこの機能を サポートさせなければなりません。

Example#2 pdo_parse_params の使用

```
     int ret;
     char *nsql = NULL;
     int nsql_len = 0;

    /* クエリを準備する前に、ちょっとその中身を確かめる必要があります。
     * もし名前で指定するパラメータが用いられていれば、その処理を
     * PDO に任せます */
      stmt->supports_placeholders = PDO_PLACEHOLDER_POSITIONAL;
      ret = pdo_parse_params(stmt, (char*)sql, sql_len, &nsql, &nsql_len TSRMLS_CC);

     if (ret == 1) {
         /* クエリが書き換えられました */
          sql = nsql;
     } else if (ret == -1) {
         /* 失敗しました */
            strcpy(dbh->error_code, stmt->error_code);
          return 0;
    }

    /* "sql" の中のクエリを準備します */
```

supports_placeholders に指定できる値は **PDO_PLACEHOLDER_NAMED**、 **PDO_PLACEHOLDER_POSITIONAL** および **PDO_PLACEHOLDER_NONE** です。 ドライバがプリペアドステートメントをまったくサポートしていない場合、 この関数は単に必要なデータを割り当てたうえで、それを返す必要があります。

Example#3 プリペアドステートメントをネイティブにサポートしていないドライバでの実装

```
static int SKEL_handle_preparer(pdo_dbh_t *dbh, const char *sql,
      long sql_len, pdo_stmt_t *stmt, zval *driver_options TSRMLS_DC)
{
       pdo_SKEL_db_handle *H = (pdo_SKEL_db_handle *)dbh->driver_data;
       pdo_SKEL_stmt *S = ecalloc(1, sizeof(pdo_SKEL_stmt));

     S->H = H;
      stmt->driver_data = S;
       stmt->methods = &SKEL_stmt_methods;
        stmt->supports_placeholders = PDO_PLACEHOLDER_NONE;

     return 1;
}
```

この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_handle_doer

```
static long SKEL_handle_doer(pdo_dbh_t *dbh, const char *sql, long sql_len TSRMLS_DC)
```

この関数は、SQL ステートメントを直接実行する際に PDO から 呼び出されます。pdo_stmt_t は作成されません。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

sql

準備される SQL ステートメントを含む文字列へのポインタ。

sql_len

SQL ステートメントの長さ。

この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_handle_quoter

```
static int SKEL_handle_quoter(pdo_dbh_t *dbh, const char *unquoted,
   int unquoted_len, char **quoted, int quoted_len, enum pdo_param_type param_type TSRMLS_DC)
```

この関数は、クエリで使用するために文字列をクォートする際に PDO から呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

unquoted

クォートされる文字列を含む文字列へのポインタ。

unquoted_len

クォートされる文字列の長さ。

quoted

クォートされた文字列へのポインタが返されるアドレスへのポインタ。

quoted_len

新しい文字列の長さ。

param_type

別のクォート形式を使用しているドライバ用の、ドライバ固有のヒント

この関数は、 **PDO::quote()** がコールされた場合、 あるいはドライバが supports_placeholder を **PDO_PLACEHOLDER_NONE** に設定した場合に PDO から呼び出されます。この関数の目的は、SQL ステートメントを 作成する際にパラメータをクォートすることです。

ドライバがネイティブのプリペアドステートメントをサポートしていない場合、 この関数を実装する必要があります。

この関数は、クォート処理が正しく行われた場合や文字列を変更する必要が なかった場合に 1、文字列の変更に失敗した場合に 0 を返します。 0 が返された場合、もとの文字列がそのまま使用されます。

SKEL_handle_begin

```
static int SKEL_handle_begin(pdo_dbh_t *dbh TSRMLS_DC)
```

この関数は、データベースのトランザクションを開始する際に PDO から呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

トランザクションを開始するために必要な処理は、すべてここで 済ませる必要があります。この関数は、成功した場合に 1、 エラーが発生した場合に 0 を返します。

SKEL_handle_commit

```
static int SKEL_handle_commit(pdo_dbh_t *dbh TSRMLS_DC)
```

この関数は、データベースのトランザクションを終了する際に PDO から呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

トランザクションをコミットするために必要な処理は、すべてここで 済ませる必要があります。この関数は、成功した場合に 1、 エラーが発生した場合に 0 を返します。

SKEL_handle_rollback

```
static int SKEL_handle_rollback( pdo_dbh_t *dbh TSRMLS_DC)
```

この関数は、データベースのトランザクションをロールバックする際に PDO から呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

トランザクションをロールバックするために必要な処理は、すべてここで 済ませる必要があります。この関数は、成功した場合に 1、 エラーが発生した場合に 0 を返します。

SKEL_handle_get_attribute

```
static int SKEL_handle_get_attribute(pdo_dbh_t *dbh, long attr, zval *return_value TSRMLS_DC)
```

この関数は、データベースの属性を取得するために PDO から呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

attr

long 値。PDO_ATTR_xxxx 型のいずれか。使用可能な 値については 定数 を参照ください。

return_value

返される属性の値。

実装においてどの属性をサポートするかは、ドライバ次第です。ドライバは 必ずこの関数を提供しなければならないわけではありません。PDO_ATTR_PERSISTENT、PDO_ATTR_CASE、 PDO_ATTR_ORACLE_NULLS および PDO_ATTR_ERRMODE については PDO ドライバが 直接処理します。

この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_handle_set_attribute

*static int SKEL_handle_set_attribute(pdo_dbh_t *dbh, long attr, zval *val TSRMLS_DC)*

*この関数は、データベースの属性を設定するために PDO から呼び出されます。 通常、これはスクリプトから **PDO::setAttribute()** をコールした場合に発生します。*

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

attr

long 値。PDO_ATTR_xxxx 型のいずれか。使用可能な 値については 定数 を参照ください。

val

属性の新しい値。

実装においてどの属性をサポートするかは、ドライバ次第です。 追加の属性をサポートする必要がないのであれば、ドライバは 必ずしもこの関数を提供しなくてもかまいません。 PDO_ATTR_CASE、 PDO_ATTR_ORACLE_NULLS および PDO_ATTR_ERRMODE については PDO ドライバが 直接処理します。

この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_handle_last_id

*static char * SKEL_handle_last_id(pdo_dbh_t *dbh, const char *name, unsigned int len TSRMLS_DC)*

この関数は、最後に挿入した行の ID を取得するために PDO から 呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

name

テーブル名あるいはシーケンス名を表す文字列。

len

パラメータ name の長さ。

この関数は、成功した場合には最後に挿入された行の ID を含む文字列、 失敗した場合には NULL を返します。これは、オプションの関数です。

SKEL_check_liveness

*static int SKEL_check_liveness(pdo_dbh_t *dbh TSRMLS_DC)*

この関数は、データベースとの持続的接続が現在確立されているかどうかを 調べるために PDO から呼び出されます。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

この関数は、データベース接続が確立されており使用可能な状態の場合に 1、 それ以外の場合（接続に失敗した場合や機能をサポートしていない場合など）に 0 を返します。

> ***注意**: これは、オプションの関数です。*

SKEL_get_driver_methods

*static function_entry *SKEL_get_driver_methods(pdo_dbh_t *dbh, int kind TSRMLS_DC)*

この関数は、PDO あるいは PDOStatement クラスに属さないメソッドが コールされた際に PDO から呼び出されます。この関数の目的は、 ドライバ固有のメソッドをクラスに追加できるようにすることです。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

kind

以下のいずれか。

PDO_DBH_DRIVER_METHOD_KIND_DBH

> *PDO クラスのインスタンスに対して メソッドのコールが試みられた場合に設定します。 ドライバは、クラスに追加したいメソッドのための function_entry テーブルへのポインタか、それが存在しない場合には NULL を返す必要があります。*

PDO_DBH_DRIVER_METHOD_KIND_STMT

> *PDOStatement クラスのインスタンスに対して メソッドのコールが試みられた場合に設定します。 ドライバは、クラスに追加したいメソッド*

のための function_entry テーブルへのポインタか、それが存在しない場合には NULL を返す必要があります。

この関数は、要求された function_entry テーブルへのポインタか、 ドライバ固有のメソッドが存在しない場合に NULL を返します。

SKEL_handle_factory

```
static int SKEL_handle_factory(pdo_dbh_t *dbh, zval *driver_options TSRMLS_DC)
```

この関数は、データベースハンドルを作成するために PDO から呼び出されます。 ほとんどのデータベースでは、データベースへの接続がここに含まれます。 中には、持続的な接続が求められる場合もあります。あるいは接続ぷーリングが 求められる場合もあります。これらのすべては、データベース/ドライバ に依存します。

dbh

ハンドルファクトリで初期化したデータベースハンドルへのポインタ。

driver_options

ドライバのオプションの配列。整数値のオプション番号をキーとします。 指定できる属性の一覧は、 定数 を参照ください。

この関数は、成功した場合には、渡されたデータベースハンドル構造体に ドライバ固有の情報を格納して 1 を返し、それ以外の場合には 0 を返して失敗したことを示します。

PDO は、handle_factory をコールする前に ドライバオプション AUTOCOMMIT および PERSISTENT を処理します。 その他のオプションを処理するのは、ハンドルファクトリの役目となります。

ドライバメソッドテーブル

pdo_dbh_methods 型のスタティックな構造体を SKEL_methods という名前で 宣言し、定義された関数へのポインタでそれを初期化しておく必要があります。 関数がサポートされていなかったり実装されていなかったりする場合は、 この関数ポインタの値を NULL に設定します。

pdo_SKEL_driver

pdo_driver_t 型の構造体を pdo_SKEL_driver という名前で宣言しなければ なりません。マクロ PDO_DRIVER_HEADER(SKEL) を使用して構造体の ヘッダを定義し、またハンドルファクトリ関数へのポインタを 設定しなければなりません。

SKEL_statement.c: ステートメントの実装

ここでは、PDO ステートメントオブジェクトがサポートする すべてのステートメント処理メソッドを実装します。

SKEL_stmt_dtor

```
static int SKEL_stmt_dtor(pdo_stmt_t *stmt TSRMLS_DC)
```

この関数は、事前に作成されたステートメントオブジェクトを 削除するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

ステートメントのために確保したドライバ固有の領域は、すべてここで 開放する必要があります。PDO は、この関数の返す値を無視します。

SKEL_stmt_execute

```
static int SKEL_stmt_execute(pdo_stmt_t *stmt TSRMLS_DC)
```

この関数は、渡されたステートメントオブジェクト内のプリペアド SQL ステートメントを実行するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_stmt_fetch

```
static int SKEL_stmt_fetch(pdo_stmt_t *stmt, enum pdo_fetch_orientation ori,
    long offset TSRMLS_DC)
```

この関数は、実行されたステートメントオブジェクトから行を取得するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

ori

どの行を取得するのかを、PDO_FETCH_ORI_xxx のいずれかで指定します。

offset

ori が PDO_FETCH_ORI_ABS あるいは PDO_FETCH_ORI_REL の場合、 offset はそれぞれ取得したい行の位置 あるいは現在の位置からの相対位置を表します。それ以外の場合は この値は無視されます。

この取得結果はドライバに依存し、データは通常 pdo_stmt_t オブジェクトの メンバ driver_data に格納されます。パラメータ ori および offset は、ステートメントがスクロール可能なカーソルを指している場合にのみ 意味を持ちます。この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_stmt_param_hook

```
static int SKEL_stmt_param_hook(pdo_stmt_t *stmt,
   struct pdo_bound_param_data *param, enum pdo_param_event event_type TSRMLS_DC)
```

この関数は、バインドされたパラメータやカラムを処理するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

param

ステートメントのパラメータあるいはバインドされたカラムを表す構造体。

event_type

このパラメータに発生するイベントの型。以下のいずれかです。

PDO_PARAM_EVT_ALLOC

PDO がバインドを割り当てる際にコールされます。 **PDOStatement::bindParam()** や **PDOStatement::bindValue()** のコール、あるいは **PDOStatement::execute()** による暗黙的な バインドの際に発生します。この時点で、何らかのアクションを とることができます。プリペアドステートメントをネイティブに 実装するドライバの場合、典型的なアクションとしては パラメータの情報の取得・スクリプトで要求された型との調整・ 適切な大きさのバッファの確保・そしてバッファへのパラメータの バインドなどがあるでしょう。 この時点では、param->parameter の zval の 型や値をあてにすべきではありません。

PDO_PARAM_EVT_FREE

各パラメータが開放される際にコールされます。パラメータに 関連するすべてのリソースはここで開放しなければなりません。

PDO_PARAM_EXEC_PRE

SKEL_stmt_execute がコールされる直前に、パラメータごとにコールされます。 実行前の最後の調整をここで行います。特に注意すべき点として、 **PDOStatement::bindParam()** による変数のバインドは ここでのみ行うべきで、それより前の段階で行ってはなりません。

PDO_PARAM_EXEC_POST

SKEL_stmt_execute がコールされた直後に、パラメータごとにコールされます。 ドライバが必要とする後処理をここで行います。

PDO_PARAM_FETCH_PRE

SKEL_stmt_fetch がコールされる直前に、パラメータごとにコールされます。

PDO_PARAM_FETCH_POST

SKEL_stmt_fetch がコールされた直後に、パラメータごとにコールされます。

このフックは、ステートメント内でバインドされたパラメータおよびカラムの それぞれについて、個々にコールされます。ALLOC および FREE イベントは、 各パラメータあるいはカラムについて 1 度コールされます。 param 構造体は driver_data フィールドを含み、これは 各パラメータについての実装固有の情報を格納するために使用されます。

その他のすべてのイベントでは、スクリプトが **PDOStatement::execute()** および **PDOStatement::fetch()** をコールするたびに PDO からコールされることになります。

バインドされたのが変数の場合、param 構造体の is_param フラグが 設定されています。それ以外の場合は param 構造体はバインドカラムを 表します。

この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_stmt_describe_col

```
static int SKEL_stmt_describe_col(pdo_stmt_t *stmt, int colno TSRMLS_DC)
```

この関数は、特定のカラムについての情報を問い合わせるために PDO からコールされます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

colno

調べたいカラムの番号。

ドライバは、pdo_stmt_t のメンバ columns[colno] に適切な情報を格納する 必要があります。この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_stmt_get_col_data

```
static int SKEL_stmt_get_col_data(pdo_stmt_t *stmt, int colno,
   char **ptr, unsigned long *len, int *caller_frees TSRMLS_DC)
```

この関数は、指定したカラムからデータを取得するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

colno

取得したいカラムの番号。

ptr

取得したデータへのポインタ。

len

ptf が指すデータの長さ。

caller_frees

設定されている場合は ptf が指すメモリは emalloc されたものであり、 使用終了後にメイン PDO ドライバがそれを開放する必要があります。 設

定されていない場合は、このコールの結果として確保されたメモリを 開放するのはドライバの役目となります。

ドライバは、結果のデータおよびその長さをそれぞれ ptr および len に 返す必要があります。メイン PDO ドライバは、データの生存期間を ドライバが管理するものと想定していることに注意しましょう。 この関数は、成功した場合に 1、失敗した場合に 0 を返します。

SKEL_stmt_set_attr

```
static int SKEL_stmt_set_attr(pdo_stmt_t *stmt, long attr, zval *val TSRMLS_DC)
```

この関数は、ステートメントオブジェクトのドライバ固有の属性を 設定するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

attr

long 値。PDO_ATTR_xxxx 型のいずれか。使用可能な 値については 定数 を参照ください。

val

属性の新しい値。

この関数はドライバに依存しており、ステートメントにデータベース固有の 属性を設定する機能を与えます。この関数は、成功した場合に 1、 失敗した場合に 0 を返します。これはオプションの関数です。 もし設定可能な追加属性をドライバがサポートしていない場合は、 メソッドテーブルで NULL を設定しておくことも可能です。PDO ドライバは、 データベースドライバに代わって設定可能属性の処理を行うことはありません。

SKEL_stmt_get_attr

```
static int SKEL_stmt_get_attr(pdo_stmt_t *stmt, long attr, zval
    *return_value TSRMLS_DC)
```

この関数は、ステートメントオブジェクトのドライバ固有の属性を 取得するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

attr

long 値。PDO_ATTR_xxxx 型のいずれか。使用可能な 値については 定数 を参照ください。

return_value

返される属性の値。

この関数はドライバに依存しており、ステートメントからデータベース固有の 属性を取得する機能を与えます。この関数は、成功した場合に 1、 失敗した場合に 0 を返します。これはオプションの関数です。 もし取得可能な追加属性をドライバがサポートしていない場合は、 メソッドテーブルで NULL を設定しておくことも可能です。PDO ドライバは、 データベースドライバに代わって取得可能属性の処理を行うことはありません。

SKEL_stmt_get_col_meta

```
static int SKEL_stmt_get_col_meta(pdo_stmt_t *stmt, int colno,
    zval *return_value TSRMLS_DC)
```

警告

この関数はうまく定義されておれず、変更する必要があります。

この関数は、指定したカラムのメタデータを取得するために PDO から呼び出されます。

stmt

SKEL_handle_preparer で初期化されたステートメント構造体へのポインタ。

colno

データを取得するカラム番号。

return_value

返されるメタデータを保持します。

ドライバの作者は、php_pdo_driver.h ヘッダに書かれているこの関数の ドキュメントを参照し、最新の情報を得てください。 この関数は、成功した場合に 1、失敗した場合に 0 を返します。 データベースドライバは、この関数を提供する必要はありません。

ステートメント操作メソッドテーブル

pdo_stmt_methods 型のスタティックな構造体を SKEL_stmt_methods という名前で 宣言し、定義された関数へのポインタでそれを初期化しておく必要があります。 関数がサポートされていなかったり実装されていなかったりする場合は、 この関数ポインタの値を NULL に設定します。

ビルド

ビルド手順は、PEAR を使用できるように設計されています (PEAR についての 詳細な情報は » http://pear.php.net/ を参照ください)。 パッケージをビルドするための設定を手助けするために用いられるファイルが 2 つあります。ひとつめは config.m4 で、これは Win32 以外のすべてのプラットフォームで使用される **autoconf** の設定ファイルです。 ふたつめが config.w32 で、これは Win32 で使用されるビルド設定ファイルです。 最初のプロジェクトをセットアップした際に、これらのファイルの雛形が 作成されます。必要に応じて、この雛形をカスタマイズする必要があります。 設定ファイルのカスタマイズを済ませると、以下の手順でドライバのビルドが 可能になります。

はじめてのビルドの前のみ

```
$ sudo pecl install PDO
```

毎回のビルドごとに

```
$ cd pdo_SKEL
$ phpize
$ ./configure
$ make
$ sudo make install
```

開発期間中は、必要に応じてこの手順を繰り返すことが可能です。

テスト

PDO のドライバをリリースする前には「コア」テストをパスする必要があります。 このテストは PHP のソース配布物から実行するように設計されており、 ドライバのテストを行うには多少の手続きが必要になります。最新の PHP 5.1 スナップショットを取得し、以下の手順で進めることを推奨します。

```
$ cp -r pdo_SKEL /path/to/php-5.1/ext
```

これにより、テストを実行するためのハーネスをセットします。 次にすべきことは、PDO の共通コアテストにリダイレクトするテストを 作成することです。このファイルの名前は common.phpt とし、 **ext_skel** で雛形を作成したときに同時に作成された tests ディレクトリの中に置かなければなりません。 このファイルの中身は以下のようになります。

```
--TEST--
SKEL
--SKIPIF--
<?php # vim:ft=php
if (!extension_loaded('pdo_SKEL')) print 'skip'; ?>
--REDIRECTTEST--
if (false !== getenv('PDO_SKEL_TEST_DSN')) {
# user set them from their shell
     $config['ENV']['PDOTEST_DSN'] = getenv('PDO_SKEL_TEST_DSN');
     $config['ENV']['PDOTEST_USER'] = getenv('PDO_SKEL_TEST_USER');
     $config['ENV']['PDOTEST_PASS'] = getenv('PDO_SKEL_TEST_PASS');
    if (false !== getenv('PDO_SKEL_TEST_ATTR')) {
        $config['ENV']['PDOTEST_ATTR'] = getenv('PDO_SKEL_TEST_ATTR');
   }
    return $config;
}
return array(
    'ENV' => array(
               'PDOTEST_DSN' => 'SKEL:dsn',
               'PDOTEST_USER' => 'username',
               'PDOTEST_PASS' => 'password'
        ),
    'TESTS' => 'ext/pdo/tests'
   );
```

これは共通コアテストを実行し、 PDOTEST_DSN、PDOTEST_USER および PDOTEST_PASS をそれぞれ PDO コンストラクタの パラメータ dsn 、username および password として渡します。 テストハーネスの実行時には、まずはじめに環境変数をチェックします。 もし存在すれば、テストファイル内にハードコーディングされた条件の かわりにそれらの適切な値を渡すようにします。

テストハーネスは以下のように起動します。

```
$ cd /path/to/php-5.1
$ make TESTS=ext/pdo_SKEL/tests PDO_SKEL_TEST_DSN="skel:dsn" ¥
  PDO_SKEL_TEST_USER=user PDO_SKEL_TEST_PASS=pass test
```

パッケージングおよび配布

パッケージを作成する

PDO ドライバは PECL 経由で公開します。そのため、PECL 拡張モジュールにおける通常のルールがすべてあてはまります。 パッケージを作成するには、正しい形式の package.xml ファイルを作成したうえで 以下のコマンドを実行します。

```
$ pecl package
```

これは、PDO_SKEL-X.Y.Z.tgz という名前の tar ボールを作成します。

パッケージを公開する前には、それがきちんとビルドできるかどうかを テストすべきです。config.m4 や package.xml の記述に間違いがあった場合、 パッケージは正常に機能しません。以下のコマンドを実行すると、 インストールは行わずにビルドのみをテストすることができます。

```
$ pecl build package.xml
```

これがうまく動作することが確認できたら、 インストールのテストを行います。

```
$ pecl package
$ sudo pecl install PDO_SKEL-X.Y.X.tgz
```

package.xml についての詳細は、PEAR プログラマむけドキュメント (» http://pear.php.net/manual/) を参照ください。

パッケージを公開する

PDO ドライバは PHP Extension Community Library (PECL) 経由で公開します。 PECL についての情報は »

http://pecl.php.net/index.php を参照ください。

---

pdo_dbh_t の定義

明示的に述べられている場合を除き、ドライバからはすべてのフィールドが 読み込み専用となります。

pdo_dbh_t

```
/* データベースへの接続を表す */
struct _pdo_dbh_t {
    /* ドライバ固有のメソッド */
     struct pdo_dbh_methods *methods;

*

    /* ドライバ固有のデータ */
    void *driver_data;

**

    /* 認証データ */
     char *username, *password;

***

    /* true の場合は、このハンドルが指すデータはすべて
     * 持続的に割り当てられる */
     unsigned is_persistent:1;

****

    /* true の場合は、ステートメントを実行するたびに COMMIT を行うかのように
     * 動作する。それ以外の場合は COMMIT を手動で実行しなければならない
     * */
     unsigned auto_commit:1;

*****

    /* true の場合は、返されるカラムのためのメモリを明示的に確保することが
     * 必要となる */
     unsigned alloc_own_columns:1;

******

     /* true の場合は、commit あるいは rollBack をコールすることができる */
    unsigned in_txn:1;

    /* しかるべきクォート処理をした後での 1 文字の長さの最大値 */
     unsigned max_escaped_char_length:3;

*******

    /* このハンドルをオープンする際に使用されるデータソース文字列 */
     const char *data_source;

********

      unsigned long data_source_len;

    /* グローバルエラーコード */
     pdo_error_type error_code;

*********

      enum pdo_case_conversion native_case

**********

, desired_case;
};
```

* ドライバは、SKEL_handle_factory() の中で これを設定しなければなりません。

** この項目はドライバが使用します。想定される使用法は、データベースとの 接続を保つために必要なインスタンスデータへのポインタを (SKEL_handle_factory() 内で) 格納することです。

*** PDO のコンストラクタに渡すユーザ名およびパスワードです。 データベースとの接続を初期化する際に、ドライバはこの値を 使用しなければなりません。

**** これが 1 に設定されている場合、dbh が参照しているすべてのデータや ドライバが割り当てた構造体はすべて持続的に 確保しなければなりません。これを実現するのは 簡単です。通常の emalloc() の代わりに pemalloc() を使用し、最後のパラメータとして このフラグの値を渡せばよいのです。メモリを適切に使用するように しないと深刻な問題を引き起こし、結果として (いちばんましな場合でも) プログラムをクラッシュさせ、最悪の場合には攻撃者が悪用可能な状態に なってしまうでしょう。

どのような理由であれ、もしドライバを持続的に実行することができないのなら **SKEL_handle_factory()** の中で*必ず* このフラグをチェックし、適切なエラーを発生させるようにしましょう。

**** この値は、関数 **SKEL_handle_doer()** および **SKEL_stmt_execute()** の中でチェックする必要が あります。これが true の場合はクエリを実行するたびに commit しなければなりません。ほとんどのデータベースは自動コミットモードを 提供しており、これを自動的に処理してくれます。

***** データベースクライアントライブラリの API が、呼び出し元が提供する バッファに取得したデータを格納するようになっている場合、**SKEL_handle_factory()** でこのフラグを 1 に 設定する必要があります。設定しておくと、PDO は それ以外の場合より早く **SKEL_stmt_describer()** をコールします。このことによりバッファの大きさを知ることができ、 データベースクライアントライブラリに対して適切なコールが できるようになります。

もしデータベースクライアントライブラリの API の実装が、 ライブラリ自身の内部バッファに格納したデータへのポインタを返す 実装になっており、フェッチコールの後でそれをコピーして使用すると いうことならば、この値は 0 のままにしておきます。

****** もしドライバがネイティブのプリペアドステートメントをサポートしない (*supports_placeholders* が **PDO_PLACEHOLDER_NONE** となっている) 場合、**SKEL_handle_quoter()** 関数によって 1 文字の長さが最大どれだけになるかをこの値に設定する必要があります。 この値は、PDO がステートメントを実行する際に必要なバッファの領域を 計算するために使用されます。

******* これは、PDO のコンストラクタに渡す DSN の値を保持します。 もし何らかの理由でドライバが DSN を変更する必要がある場合は、**SKEL_handle_factory()** の中で更新しなければ なりません。このメンバを変更することは避けるべきです。 もし変更する際には、*data_source_len* が正しい値であることを確かめる必要があります。

******** ドライバのメソッドのコール中にエラーが発生した場合は、エラーの内容に 該当する SQLSTATE コードをこのメンバに設定し、エラーを返さなければ なりません。この HOW-TO では、エラーが検出された際には **SKEL_handle_error()** をコールし、そこでエラーコードを 設定するという手法を推奨します。

********* この値は **SKEL_handle_factory()** の中で 設定しなければなりません。この値は、結果セット内のカラム名を データベースがどのように返すかを表します。クエリで指定された とおりに大文字小文字を使用する場合は、この値を **PDO_CASE_NATURAL** (デフォルト値です) に設定します。カラム名をつねに大文字で返す場合は、この値を **PDO_CASE_UPPER** に設定します。 カラム名をつねに小文字で返す場合は、この値を **PDO_CASE_LOWER** に設定します。 ここで設定した値は、ユーザが **PDO_ATTR_CASE** 属性を設定した場合に PDO が大文字小文字変換をすべきかどうかを 決定するために使用されます。

---

pdo_stmt_t の定義

明示的に述べられている場合を除き、ドライバからはすべてのフィールドが 読み込み専用となります。

pdo_stmt_t

```
/* プリペアドステートメントを表す */
struct _pdo_stmt_t {
    /* ドライバ固有 */
     struct pdo_stmt_methods *methods;

**********

    void *driver_data;

***********


    /* true の場合は、すくなくとも 1 回はこのステートメントの実行に
     * 成功していることを表す */
    unsigned executed:1;

*************

    /* true の場合は、ステートメントがプレースホルダをサポートしており
     * プリペアドステートメント用に bindParam() を実装できることを表す
     * false の場合は、プリペアおよびバインドを PDO がエミュレートする
     * 必要があることを表す */
     unsigned supports_placeholders:2;

**************


    /* 結果セットのカラム数。ステートメントがすくなくとも 1 度実行される
     * までは有効な値は設定されない。時には、(ドライバレベルでの)
     * フェッチがコールされるまでは有効にならないこともある。
     * */
    int column_count;

***************

     struct pdo_column_data *columns;

****************


    /* このステートメントが準備された dbh を指す */
     pdo_dbh_t *dbh;

    /* 入力バインドパラメータを指す。ドライバの中には入出力パラメータを
     * サポートしているものもあるが、その動作に依存することはできない */
```

```
    HashTable *bound_params;
    /* 名前から位置への置き換えを行う際に、このマップを使用する */
    HashTable *bound_param_map;
    /* 結果セット内で名前で指定した（あるいは位置で指定した）カラムに
     * 関連付けた PHP 変数を指す */
    HashTable *bound_columns;

    /* 意味のある値が設定されていない場合もある */
    long row_count;

    /* ステートメントの現在のクエリを保持する */
    char *query_string;
    int query_stringlen;

    /* バインド変数を展開した後のクエリのコピーで、
     * ドライバがプリペアドステートメントをエミュレートしている
     * 場合にのみ使用する */
    char *active_query_string;
    int active_query_stringlen;

    /* カーソル固有のエラーコード */
    pdo_error_type error_code;

    /* ドライバ固有のパラメータ命名規則（例: pgsql ドライバ）
     * のためにクエリパーサが使用する */
    const char *named_rewrite_template;
};
```

* ドライバは、**SKEL_handle_preparer()** の中で これを設定*しなければなりません*。

** この項目はドライバが使用します。想定される使用法は、データベースとの 接続を保つために必要なインスタンスデータへのポインタを (**SKEL_handle_factory()** 内で) 格納することです。

*** ステートメントが最初に実行された後に、PDO がこれを設定します。 ドライバは、この値を調べることで初回にのみ必要な処理を飛ばす 最適化を行います。

**** 詳細は 雛形への肉付け で説明します。

***** 結果セット内のカラム数を、ドライバがこのフィールドに設定します。 通常これは **SKEL_stmt_execute()** で設定しますが、 データベースの実装によっては **SKEL_stmt_fetch()** を最低 1 回コールするまではカラム数がわからないことがあります。 **SKEL_stmt_next_rowset()** を実装するドライバで 新しい行セットが使用可能になった場合、ドライバはカラム数を更新する必要があります。

****** カラム数に設定した値に応じて、PDO がこのフィールドを確保します。 **SKEL_stmt_describe()** の中で、各カラムの内容を 設定する必要があります。各カラムについて設定する必要があるのは、 *precision* 、*maxlen* 、 *name* 、*namelen* および *param_type* です。 *name* は、**emalloc()** を使用して確保することが期待されています。PDO は、しかるべき時に **efree()** を実行します。

定数

データベースおよびステートメントの属性

属性	値
PDO_ATTR_AUTOCOMMIT	BOOL 自動コミットが設定されている場合に TRUE、それ以外の場合に FALSE。 dbh->auto_commit がこの値を含みます。PDO によって直接処理されます。
PDO_ATTR_PREFETCH	LONG ドライバがサポートしているプリフェッチサイズの値。
PDO_ATTR_TIMEOUT	LONG データベース操作がタイムアウトするまでの長さ。
PDO_ATTR_ERRMODE	LONG PDO によって処理されます。
PDO_ATTR_SERVER_VERSION	STRING このドライバが現在接続しているサーバとそのバージョンを、 "人間が理解できる形式" の文字列で表したもの。
PDO_ATTR_CLIENT_VERSION	STRING このドライバがサポートしているクライアントとそのバージョンを、 "人間が理解できる形式" の文字列で表したもの。
PDO_ATTR_SERVER_INFO	STRING "人間が理解できる形式" のサーバの説明。
PDO_ATTR_CONNECTION_STATUS	LONG 未定義の値。
PDO_ATTR_CASE	LONG PDO によって処理・操作が行われます。
PDO_ATTR_CURSOR_NAME	STRING "where current in <名前>" 形式の SQL ステートメントで使用する、データベースカーソルの名前を 表す文字列。
PDO_ATTR_CURSOR	LONG PDO_CURSOR_FWDONLY 先送りのみのカーソル PDO_CURSOR_SCROLL スクロール可能なカーソル

上で示した属性の値は、すべて Zend API で定義されています。Zend API には *zval を値に変換するためのマクロが含まれています。これらのマクロは、PHP ビルドディレクトリの下にある Zend ディレクトリ内の Zend ヘッダファイル、zend_API.h で定義されています。 これらの属性の中には、PDO_ATTR_CURSOR および PDO_ATTR_CURSOR_NAME のように、ステートメント属性ハンドラとともに使用するものもあります。 詳細な情報は、ステートメント属性処理関数を参照ください。

エラー処理

エラー処理は、PDO とデータベースドライバとの間のハンドシェイキング プロトコルを使用して実装されています。データベースドライバのコードは、 インターフェイス関数から失敗 (0) を返すことにより、 エラーが発生したことを PDO に通知します。ゼロが返されると、 (pdo_dbh_t あるいは pdo_stmt_t のいずれかの) 制御ブロックの error_code フィールドに値が設定されます。 正しい値が使用されることを保証するため、両方のブロックのフィールドに 同じ値を設定しておくことを推奨します。

error_mode フィールドの大きさは 6 バイトであり、5 文字の ASCIIZ SQLSTATE 識別コードが含まれます。このコードがエラーメッセージを処理します。 内部の PDO エラーメッセージテーブルから、SQLSTATE コードを使用して エラーメッセージを検索します (エラーコードおよびそのメッセージの 一覧は、pdo_sqlstate.c を参照ください)。PDO で定義されていないコードが 指定された場合は、デフォルトの値 "Unknown Message" が使用されます。

SQLSTATE コードおよびエラーメッセージに加え、PDO は ドライバ固有の fetch_err() ルーチンをコールすることにより エラーの追加データを取得します。このルーチンには配列が渡され、 ドライバはその配列に追加情報を書き込みます。配列の各項目には、 以下のような追加情報が格納されます。

1. ネイティブエラーコード。たいていの場合、これはデータベースの API から取得したエラーコードになるでしょう。
2. 内容を表す文字列。ここには、エラーに関連する追加情報を何でも 含めることができます。一般的には、エラーメッセージ・コード上で エラーが発生した位置やその他開発者にとって有益であると思われる 情報を含めます。エラーが発生した際にデータベースインターフェイスから 得られる診断情報は、すべて含めておくとよいでしょう。 ドライバが検出したエラー (例えばメモリ確保の問題など) については、 どのような情報を提供すべきかはドライバの開発者が判断することに なります。

拡張モジュールに関する FAQ

Zend Engine 2 API リファレンス

Zend Engine 1

目次

Zend Engine 1 は、PHP 4 系のバージョンの内部で使われているエンジンです。 すでに第一線を退いていますが、PHP 4 はいまだに多くの場所で使われています。 そこで、以前の ZE1 のドキュメントについてもここにそのままの形で残しておきます。

以前の導入ページ

PHP そのもの、あるいは Zend 拡張モジュールの開発を始めようと思うなら、 その膨大な API 群について学ぶ必要があります。ここでは、現在公開されている さまざまなバージョンの PHP および Zend エンジンが提供する API 群について 紹介します。ここに挙げられている情報の多くは若干古くなっています。 そのため、このマニュアルのほかに、PHP ソースに含まれている README.SELF-CONTAINED-EXTENSIONS や README.EXT_SKEL といったファイルについても 読んでみたくなることでしょう。

PHP拡張モジュールの作者用のストリームAPI

概要

PHPストリームAPIは、PHP拡張モジュールにファイルおよびソケット処理 用の統一化された手段を導入するものです。 共通の操作を行なうための標準関数を有する単一のAPIを使用することに より、ストリームAPIは拡張モジュールがファイル、ソケット、URL、メモリ スクリプトが定義したオブジェクトにアクセスすることを可能にします。 ストリームは、新規ストリームを登録するために動的にロードされる モジュール(およびスクリプト!)とすることができる実行時に拡張可能な APIです。

ストリームAPIの目的は、ファイル、URL、その他のストリームにできるデータ ソースを平易な統一されたAPIにより、開発者が容易にオープンできるように することです。APIは、ほぼANSI C stdio関数と(多くの主な関数について同等 の意味を有しており、)類似しています。このため、Cプログラマは、 ストリームに慣れている印象を受けるはずです。

ストリームAPIは、いくつかの異なるレベルを処理します。 基本レベルでは、APIはストリーマブルなデータソースを表す php_streamオブジェクトを定義します。 やや高いレベルでは、APIは、URLからのデータおよびメタデータの取得を サポートするために低レベルAPIをラップした php_stream_wrapper オブジェクトを定義します。 追加のパラメータ context は、 ほとんどのストリーム作成関数で使用できます。 これはラッパーの stream_opener メソッドに渡され、ラッパーの挙動を微調整します。

あらゆるストリームは、一度オープンされると任意の数の filters を適用することができます。 これは、ストリームがデータを読み書きする際にそのデータを加工します。

ストリームは、ファイル処理の他の形式にキャスト(変換)でき、 大きな問題もなくサードパーティ製のライブラリと組み合わせて使用する ことができます。これにより、これらのライブラリがURLソースからデータに 直接アクセスできるようになります。 使用するシステムに**fopencookie()**または**funopen()**関数がある場合、 任意のPHPストリームをANSI stdioを使う任意のライブラリに渡すことさえ できます!

> **注意**: この章で説明する関数は PHP のソースコード内で用いられるものであり、 PHP の関数ではありません。PHP のユーザ用のストリーム関数については ストリーム関数のリファレンス をご覧ください。

ストリームの基本

ストリームの使用方法は、ANSI stdio関数の使用と非常に似ています。 主な違いは、使用を開始するストリームを得る方法です。 多くの場合、ストリームのハンドルを得るために **php_stream_open_wrapper()**を使用します。 この関数の動作は、以下の例で示すようにfopenと非常によくにています。

Example#1 PHPホームページを表示するための簡単なストリームの例

```
php_stream * stream = php_stream_open_wrapper("http://www.php.net", "rb", REPORT_ERRORS, NULL);
if (stream) {
      while(!php_stream_eof(stream)) {
          char buf[1024];

            if (php_stream_gets(stream, buf, sizeof(buf))) {
               printf(buf);
          } else {
               break;
          }
    }
      php_stream_close(stream);
}
```

以下の表にその他の一般的なANSI stdio関数と等価なストリーム関数を 示します。注記で除外されていない限り、関数の意味は同じです。

ANSI stdioと等価なストリームAPI関数

ANSI stdio関数	PHP ストリーム関数	注意
fopen	php_stream_open_wrapper	ストリームではパラメータが増えています
fclose	php_stream_close	
fgets	php_stream_gets	
fread	php_stream_read	パラメータnmembの値を1と仮定すると、プロトタイプはread(2)により似ることになります
fwrite	php_stream_write	パラメータnmembの値を1と仮定すると、プロトタイプはwrite(2)により似ることになります
fseek	php_stream_seek	
ftell	php_stream_tell	
rewind	php_stream_rewind	
feof	php_stream_eof	
fgetc	php_stream_getc	
fputc	php_stream_putc	
fflush	php_stream_flush	
puts	php_stream_puts	fputsではなく、putsと同じ意味
fstat	php_stream_stat	ストリームはより情報の多いstat構造体を有しています

リソースとしてのストリーム

すべてのストリームは、作成されるとリソースとして登録されます。これにより、 たとえ致命的なエラーが発生したとしても適切な後処理が行われることが保障されます。 PHP のすべてのファイルシステム関数は、ストリームリソースに対して操作することができます。 つまり、あなたの作成した拡張モジュールは、 通常の PHP ファイルポインタをパラメータとして受け取って 結果をストリームで返すことができるということです。 ストリーム API により、この処理が楽にできるようになっています。

Example#2 ストリームをパラメータとして受け取る方法

```
PHP_FUNCTION(example_write_hello)
{
     zval *zstream;
     php_stream *stream;

      if (FAILURE == zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "r", &zstream))
         return;

      php_stream_from_zval(stream, &zstream);

    /* これでストリームを使うことができます。しかし、ストリームの
       "所有者" はこの関数ではなく、呼び出し元のスクリプトです。
       つまり、この関数内でストリームを閉じてはいけないということです。
       そんなことをすると PHP がクラッシュしてしまいます! */

      php_stream_write(stream, "hello¥n");

     RETURN_TRUE();
}
```

Example#3 関数からストリームを返す方法

```
PHP_FUNCTION(example_open_php_home_page)
{
     php_stream *stream;

      stream = php_stream_open_wrapper("http://www.php.net", "rb", REPORT_ERRORS, NULL);

      php_stream_to_zval(stream, return_value);

    /* これ以降、ストリームの "所有者" は呼び出し元スクリプトとなります。
       もしここでストリームを閉じると、PHP がクラッシュしてしまいます! */
}
```

ストリームの後始末が自動的に行われることから、 わざわざ後始末を気にしたりしないずさんなプログラマでいても大丈夫と思われるかもしれません。 確かにそれでもうまくいくでしょうが、いくつかの理由からこれはお勧めできません。 ストリームのオープン中はシステムリソースがロックされるので、 使用済みのファイルをオープンしたままにしておくと、 他のプロセスがファイルにアクセスできなくなります。 大量のファイルを扱うスクリプトでは、 使用済みのリソースを溜め込み続けるとメモリやファイル記述子の番号がいっぱいになってしまいます。 その結果ウェブサーバがリクエストを受け付けられないようになります。 どうです? あまりいい話ではないでしょう? ストリーム API には、すっきりとしたコードが書けるような細工が組み込まれています。 ストリームを閉じるべき場所で閉じていない場合は、 ウェブサーバのエラーログに有用なデバッグ情報が出力されます。

> **注意**: 拡張モジュールの開発中は、常にデバッグビルド版の PHP を使用するようにしましょう (configure の際に --enable-debug を指定します)。 そうすることで、メモリリークやストリームのリークに関する重要な警告を受け取れるようになります。

時には、リクエストを持続させるためにストリームをオープンし続けることが有用なこともあるでしょう。 例えばログを記録したり結果をファイルにトレースする場合などです。 このようなストリームについて、確実に後始末を行うコードを書くことはさほど難しくありません。 しかしその数行のコードがどうしても必要なのかというと、そうではないでしょう。 このような場合にコードを書く手間を省くため、ストリームに対して 「このストリームの後始末は自動処理にまかせる」という印をつけることができます。 こうすると、ストリームの後始末が自動的に行われた際に、 ストリーム API は何の警告も発しなくなります。この印をつけるには **php_stream_auto_cleanup()** を使用します。

ストリームをオープンする際のオプション

これらの定数は、ストリームファクトリ関数の操作に影響を及ぼします。

IGNORE_PATH
これはストリームのデフォルトのオプションです。要求されたファイルに ついて、include_path に列挙されたパスを検索しないようにします。
USE_PATH
要求されたファイルについて、include_path で列挙されたパスも検索します。
IGNORE_URL
ストリームを開く際に、登録された URL ラッパーの存在を無視します。 非 URL ラッパーについては考慮され、これらがパスをデコードします。 このフラグの逆はありません; ストリームAPIは登録されたすべての ラッパーをデフォルトで使用しようとします。
IGNORE_URL_WIN
Windows においては、IGNORE_URL と等価です。 他のシステムでは効果はありません。
ENFORCE_SAFE_MODE
ファイルを開く前に、ストリームの背後の実装が safe_mode チェックを ファイルに対して行うよう指示します。 このフラグを省くと、safe_mode チェックが行われず、PHP プロセスが アクセス権をもつすべてのファイルに対してオープンが可能になります。
REPORT_ERRORS
このフラグがセットされていて、ファイルまたは URL を開く際に何らかのエラーが 発生した場合に、ストリーム API は php_error 関数をあなたの代わりに 実行します。これは、パスや URL がユーザ名やパスワードなど エラー時にブラウザに表示されるべきでない情報を含むときに (それがセキュリティ上のリスクになるため) 有用です。 ストリーム API がエラーを出すときは、まずユーザ名やパスワードといった 情報をパスから取り除いた後で、エラーメッセージを安全な形にしてから エラーを出力します。
STREAM_MUST_SEEK
このフラグは、拡張モジュール内で、本当にストリームをランダムにシーク する必要があるときに有用です。 いくつかの種類のストリームはそのままの形ではシークができないことが あるので、このフラグをセットしておくと、ストリーム API は、 まず開かれたストリームがシーク可能かどうかを調べ、シークできない場合は、 ストリームのデータをシーク可能な一時的なストレージ (テンポラリファイル あるいはメモリーストリーム) にコピーします。 このフラグは、ストリームに対してシーク動作を行ってから書き込みを行うような 場合には適していません。アクセス対象のストリームは当初アクセスを要求した リソースに必ずしも関連付けられているとは限らないからです。

> **注意**: もし、要求されたリソースがネットワークベースであった場合、 この関数は、すべてのデータが読み込まれるまでブロックします。

STREAM_WILL_CAST
もしあなたの拡張モジュールがサードパーティのライブラリを利用していて、 そのライブラリには FILE* かファイルディスクリプタを渡さなくてはならないとき、このフラグを使うと、ストリーム API にリソースを オープンしても、バッファリングは行わないよう指示することができます。 その後、**php_stream_cast()** を使い、そのライブラリの 必要とする FILE* や ファイルディスクリプタを取得できます。 このフラグは HTTP URL にアクセスしたとき、 実際のストリームのデータが不定なオフセットの先から始まるような 場合において、特に有用です。 このオプションはストリーム API のレベルでのバッファリングを 無効にするため、ストリーム関数のパフォーマンスが悪くなるかも しれませんが、このオプションを利用するということは、あなたが ストリームの背後にある実装に合わせるよう ストリーム API 関数を使うことをストリームに宣言したということ ですから、それは許容範囲内と考えられます。 このオプションは本当に必要だと確信があるときにのみ用いてください。

Zend API: PHP のコアをハックする

導入

Those who know don't talk. (知ってる人は教えようとしない)

Those who talk don't know. (教えてる人はあまり知らない)

時には、「あるがままの」PHP では要件を満たさないことがあります。平均的な ユーザがこのような事態に陥ることはまずありませんが、プロがアプリケーションを 作成していると、速度面や機能面ですぐに PHP の限界に達してしまいます。 言語の制限により、新機能が常にネイティブ実装されるとは限りません。 そんな場合には、たった 1 行のコードのために大げさなライブラリを利用 しなければならず非常に不便です。このような PHP の欠点に打ち勝つための なんらかの手段が必要です。

もしこの域に達したなら、PHP の心臓部に手を触れ、その中身 (PHP を動かして いる C のコード) を探ってみるとよいでしょう。

警告

この情報は、現在はかなり時代遅れになっています。 PHP 4 の初期バージョンで使用されていた、 初期の ZendEngine 1.0 の API しか網羅していない部分もあります。

より新しい情報は、PHP のソースに含まれる各種 README ファイル、 あるいは Zend の Web サイトにある » Internals を参照ください。

概要

「PHP を拡張する」と言うのは簡単ですが、実際に行うのは大変なことです。 PHP は数メガバイトのソースコードからなる成熟したツールに発展してきており、このようなシステムをハックするには、学ばねばならないことが あります。この章を構成するにあたり、我々は「実際に作業することによって 学んでもらう」という手法をとることにしました。これは科学的とは言えないし プロフェッショナルな方法でもありません。しかし、楽しく学ぶことが できて最終的によい結果に終わると考えたのです。 以下の節では、最も基本的な拡張モジュールをお手軽に作成する方法を 学びます。その後で、Zend API の高度な機能について学びます。 もうひとつの方法としては、機能・デザイン・ヒント・裏技などを 一気に学ぶ、つまり実践的な手法の前にまず大きな絵の全体を見せてしまう というものがあります。何の小細工もせずにすむという点で、こちらの 方法のほうが「よりよい」ものではあるのですが、この方法は 非常にフラストレーションがたまり、時間と気力をかなり消費するもの でもあります。そのため、我々は前者の方法をとることにしたのです。

この章では PHP の内部動作についてできる限りの知識を伝えるように 心がけています。しかしながら、いつどんなときでも 100% 動作する 完璧な PHP 拡張モジュール作成ガイドを提供するのは不可能であることを 知っておいてください。PHP は巨大で複雑なパッケージであるので、 実際に手を動かしながら勉強していかないと内部構造は理解できないでしょう。 そのため、実際のソースとともに作業を進めていくことを推奨します。

Zend とは何 ? そして PHP とは何 ?

Zend という名前は PHP のコアとなる言語エンジンを 指します。PHP は、外部から見た際のシステム全体を 指す単語です。最初はすこし紛らわしく感じるかもしれませんが、これは そんなに複雑な話ではありません (以下を参照ください)。 Web スクリプトのインタプリタを実装するには、3 つのパートが必要です。

1. インタプリタ (interpreter) は 入力コードを解析し、変換し、そして実行します。
2. 機能 (functionality) は、言語の機能 (関数など) を実装します。
3. インターフェイス (interface) は、 Web サーバなどと会話をします。

1 のすべてと 2 の一部を担当するのが Zend で、2 および 3 を 担当するのが PHP です。この 2 つをあわせることで PHP パッケージが 完成します。Zend 自身は言語のコアのみを受け持つもので、 事前に定義された関数などの PHP の基本部分のみを実装します。 PHP が、この言語の優れた可能性を作り出しているすべての 拡張モジュールを受け持ちます。
PHP の内部構造

これ以降のセクションでは、PHP を拡張するにはどこをどのように変更すれば よいのかを説明します。

拡張の可能性

先ほど 示したとおり、PHP を拡張するには 3 つの方法があります。それは 外部モジュール・組み込みモジュール・Zend エンジン の 3 つです。以下の節で、これらそれぞれについて考えます。

外部モジュール

外部モジュールは、スクリプトの実行時に dl() 関数を使用して読み込みます。この関数は、ディスクから共有オブジェクトを 読み込んで、呼び出し元のスクリプトからその機能を使用できるように します。スクリプトが終了すると、外部モジュールはメモリから削除 されます。この方法には利点も欠点もあります。それを以下の表にまとめます。

利点	欠点
PHP をコンパイルせずに外部モジュールを使用できる。	スクリプトが実行されるたび (アクセスされるたび) に 共有オブジェクトを読み込む必要があり、速度が非常に遅くなる。
その機能を「外部にまかせる」ことにより、PHP のサイズを 小さく抑えられる。	外部の追加ファイルによってディスクを散らかしてしまう。
	そのモジュールの機能を使用するすべてのスクリプトが dl() をコールするか、あるいは *php.ini* の中の *extension* タグを変更する (この方法が適切でない場合もある) 必要がある。

要するに、外部モジュールが有用なのは以下のような場合です。 サードパーティの製品・PHP のちょっとした機能拡張で、めったに 使われないもの・単なるテスト目的で作成するものなど。 追加機能を手っ取り早く開発するには、外部モジュールが最適です。 頻繁に使用するものであったり大規模な実装であったり、あるいは 複雑なコードの場合などは、欠点が利点を上回ってしまうでしょう。

サードパーティは、php.ini の extension タグを使用して PHP に拡張モジュールを 追加することを考えるかもしれません。これらの拡張モジュールは 本体のパッケージとは完全に切り離されます (これは、商用の環境では とても便利な機能です)。商用製品を配布する場合は、単に追加モジュール のみを含むディスクやアーカイブを出荷すればよいのであり、他の 拡張モジュールが使用できなくなるような PHP バイナリを作成する 必要がなくなります。

組み込みモジュール

組み込みモジュールは、PHP のコンパイル時に直接組み込まれ、PHP プロセスと一緒に動作します。すべてのスクリプトで、その機能が 使用可能となります。外部モジュールの場合と同様、組み込みモジュールに ついても利点と欠点があります。それを以下の表にまとめます。

利点	欠点
いちいちモジュールを読み込む必要がない。 なにもしなくてのその機能が使用できる。	組み込みモジュールを更新するには PHP を再コンパイル する必要がある。
別の外部ファイルでディスクを散らかすことはない。 すべては PHP のバイナリに組み込まれる。	PHP バイナリのサイズが大きくなり、メモリ消費量も増える。

比較的変更の少ない安定した関数のライブラリで、平均以上の速度を 要求するものであったり、あるいは多くのスクリプトから頻繁に 使用されるようなライブラリなどの場合は、組み込みモジュールに するのが最適です。PHP を再コンパイルしなければならないという 問題は、それによって得られるスピードや使いやすさにくらべたら たいしたことはありません。しかし、小さな変更が頻繁に繰り返される ような場合には、組み込みモジュールは不適当です。

Zend エンジン

もちろん、拡張機能を Zend エンジンに直接組み込むこともできます。 言語そのものの振る舞いを変更したい場合、あるいは言語のコアに 特別な関数を直接組み込む必要がある場合などには、この方式がよいでしょう。 しかし、一般的には Zend エンジンを変更することは避けるべきです。 ここを変更してしまうとその他の部分で非互換性が発生する可能性があり、 特別に変更された Zend エンジンに対応できる人は誰もいなくなるでしょう。変更内容を PHP のソースから切り離すことができなくなりますし、 また「公式」なソースリポジトリからアップデートを行うと、変更内容が 上書きされてしまいます。そのため、この手法はよくないものと 考えられています。使用することはめったにないため、この文書では この手法については取り上げません。

ソース配置

> **注意**: この章の残りの部分に進む前に、お好みの Web サーバの (変更されていない) ソースを取得しておきましょう。ここでは、Apache (» http://www.apache.org/ で取得できます) を使用します。また、もちろん PHP のソース (» http://www.php.net/ にあります - 言うまでもないですよね?) も必要です。
> PHP の動作環境を自分でコンパイルして作成できるようにしておいて ください! この方法についてはここでは触れませんが、この章の内容を 学習しようとするのなら、最低限知っておくべき基本的な内容です。

コードの内容について説明する前に、PHP のファイルを探索する助けになるよう ソースツリーの内容に慣れておくべきです。これは、拡張モジュールを 開発したりデバッグしたりする際の必須技能です。

以下の表では、主なディレクトリの内容について説明しています。

ディレクトリ　内容

php-src	PHP 本体のソースファイルおよびヘッダファイル。 PHP の API 定義やマクロなどはここにあります (重要)。 それ以外のものは、このディレクトリの下位階層にあります。
php-src/ext	動的モジュール、組み込みモジュールのソース置き場。 デフォルトでは、PHP 本体のソースツリーに統合された「公式」 モジュールが配置されています。 PHP 4.0 以降、これらの標準 拡張モジュールを (そのモジュールがサポートしていれば) 動的モジュールとしてコンパイルすることが可能となりました。
php-src/main	このディレクトリには PHP 本体のマクロや定義があります (重要)。
php-src/pear	PHP Extension and Application Repository (PEAR) のディレクトリです。 ここには PEAR のコアファイルが含まれま

	す。
php-src/sapi	さまざまなサーバ用の抽象化レイヤのコードを含みます。
TSRM	Zend および PHP の "Thread Safe Resource Manager" (TSRM) の場所です。
ZendEngine2	Zend エンジンのファイルがあります。この中で、Zend の API 定義やマクロなどのすべてが見つけられるでしょう (重要)。

PHP パッケージに含まれるすべてのファイルについて取り上げることは この章の範囲を超えています。しかし、以下のファイルについては 詳しく見ておくべきでしょう。

- *PHP の main ディレクトリにある php-src/main/php.h。 このファイルには PHP のマクロおよび API 定義の大半が含まれています。*
- *Zend ディレクトリにある php-src/Zend/zend.h。 このファイルには Zend のマクロおよび定義の大半が含まれています。*
- *これもまた Zend ディレクトリにある php-src/Zend/zend_API.h。 ここでは Zend の API を定義しています。*

これらのファイルからインクルードされているいくつかのファイルについても 見ておきましょう。例えば、Zend エンジンの実行や PHP の初期化ファイルの サポートに関連するファイルが含まれます。これらのファイルを読んだ後で、 パッケージ全体を見回し、ファイルやモジュールの相互依存性 - 各ファイル・モジュールがお互いにどのようにかかわりあっているのか、 どのようにお互いを使用しているのか - を調べましょう。これにより、 PHP のコーディングスタイルにも慣れることができます。PHP を 拡張しようと思うなら、早いうちにこのスタイルに適応すべきです。

拡張規約

Zend は、ある規約に基づいて構築されています。 この標準規約を破ることを避けるため、 以下の節で説明する規則を守らなければなりません。

マクロ

重要なタスクのほぼ全てについて、Zend では便利なマクロを定義しています。 以下の表および図で、基本的な関数・構造体およびマクロについて 説明しています。マクロ定義のほとんどは zend.h あるいは zend_API.h にあります。この章を勉強したあとで、 これらのファイルをじっくり読んでみることをお勧めします（もちろん今この場で読んでもよいのですが、 まだこの段階ではすべてを理解することはできないでしょう)。

メモリ管理

特にサーバソフトウェアにとって、リソース管理は重大な問題です。 メモリは最も貴重なリソースのひとつなので、 メモリ管理には最大限の注意を払わねばなりません。 メモリ管理の一部は Zend によって抽象化されており、 この抽象化を使用すべきなのは明白です。この抽象化を使用することにより、 Zend はすべてのメモリ割り当てを完全に制御できるようになります。 そのブロックが使用中なのかどうかを Zend が判断し、 未使用のブロックや参照されていないブロックを自動的に開放することで メモリリークを防ぐことができます。 このために使用する関数を、以下の表にまとめます。

関数	説明
emalloc()	malloc() の代わりに使用します。
efree()	free() の代わりに使用します。
estrdup()	strdup() の代わりに使用します。

	strndup() の代わりに使用します。
	estrdup() より高速で、バイナリセーフです。
estrndup()	複製する文字列の長さが事前にわかっている場合には、 この関数を使用することを推奨します。
	calloc() の代わりに使用します。
ecalloc()	
	realloc() の代わりに使用します。
erealloc()	

emalloc() および estrdup()、estrndup()、 ecalloc()、erealloc() は、内部メモリを確保します。efree() は、これらの関数で確保したブロックを開放します。 e() 関数が管理するメモリは、 現在のプロセス内でローカルであるものとして扱われます。 このプロセスによって実行されているスクリプトが終了すると、 すぐにメモリが破棄されます。*

警告

スクリプトの終了後も残り続けるメモリを確保するために、 malloc() および free() を使用することも可能です。しかし、これらの関数を使用するのは Zend API がどうしてもそれを要求している場合に限定し、 最大限の注意を払うようにしてください。それ以外の場合に使用すると、 メモリリークが発生する恐れがあります。

Zend は、マルチスレッド Web サーバをサポートするための スレッドセーフなリソース管理機能も提供しています。 この機能を使用する場合は、複数スレッドを同時に実行できるようにするため、 すべてのグローバル変数をローカルの構造体に割り当てなければなりません。 この文書が書かれた時点では Zend のスレッドセーフモードはまだ完成していません。 そのため、この文書ではこれ以上この機能について取り上げません。

ディレクトリ関数およびファイル関数

Zend モジュール内では、以下のディレクトリ関数およびファイル関数を使用しなければなりません。 これらの関数の機能はそれぞれ対応する C 関数と同じですが、 さらにスレッドレベルでの仮想実行ディレクトリがサポートされています。

Zend 関数	標準 C 関数
V_GETCWD()	getcwd()
V_FOPEN()	fopen()
V_OPEN()	open()
V_CHDIR()	chdir()
V_GETWD()	getwd()
V_CHDIR_FILE()	ファイルのパスを引数として受け取り、 現在の実行ディレクトリをそのファイルのディレクトリに移動します。
V_STAT()	stat()
V_LSTAT()	lstat()

文字列の処理

Zend エンジンでは、文字列はその他の値（整数値、論理値など）と少し異なる方法で処理されます。 これらの値を保存するために、追加のメモリを確保する必要はありません。 関数から文字列を返したい場合は、新しい文字列変数を シンボルテーブルかそれに類似のものに登録します。 その文字列が使用するメモリは、先ほど説明した e() 関数で事前に確保しておかなければなりません（この段階では、まだあまりピンとこないかもしれません。 とりあえずは頭の片隅に置いておいてください。あとでもう一度説明します）。*

複雑な型

配列やオブジェクトのような複雑な型については、扱いかたが異なります。 Zend はこれらの型を扱うための API を提供しており、 これらの型はハッシュテーブルとして保存されます。

> ***注意**: これ以降のサンプルソースでは、 読みやすさを考慮して integer などの単純な型のみを使用します。 より高度な型を作成する方法については、この章の後半で説明します。*

PHP の自動ビルドシステム

PHP 4 には、非常に柔軟な自動ビルドシステムがあります。 すべてのモジュールは、ext ディレクトリ以下に配置されています。各モジュールは、モジュール自身のソースに加えて config.m4 というファイルを持っています。 これは拡張モジュールの設定用のファイルです (» http://www.gnu.org/software/m4/manual/m4.html を参照ください)。

これらの全てのファイルの雛形および .cvsignore は、ext ディレクトリ内にある ext_skel というシェルスクリプトで作成できます。 作成したいモジュールの名前を、スクリプトの引数として渡します。 このスクリプトは引数と同じ名前のディレクトリを作成し、 適切な雛形ファイルを作成します。

順を追って見ていくと、この手順は次のようになります。

```
:~/cvs/php4/ext:>  ./ext_skel  --extname=my_module
Creating  directory  my_module
Creating  basic  files:  config.m4  .cvsignore  my_module.c  php_my_module.h  CREDITS  EXPERIMENTAL  tests/001.phpt  my_module.php

To  use  your  new  extension,  you  will  have  to  execute  the  following  steps:

1.   $ cd ..
2.    $  vi  ext/my_module/config.m4
3.    $  ./buildconf
4.    $  ./configure  --[with|enable]-my_module
5.   $ make
6.    $  ./php  -f  ext/my_module/my_module.php
7.    $  vi  ext/my_module/my_module.c
8.   $ make

Repeat  steps  3-6  until  you  are  satisfied  with  ext/my_module/config.m4  and
step  6  confirms  that  your  module  is  compiled  into  PHP.  Then,  start  writing
code  and  repeat  the  last  two  steps  as  often  as  necessary.
```

この手順により、先ほど説明したファイルが作成されます。 新しく作成したモジュールを自動ビルドシステムに組み込むには、 buildconf を実行しなければなりません。 これは、ext ディレクトリ内を検索し、 見つかった全ての config.m4 ファイルをもとにして configure スクリプトを再作成します。

Zend API: PHP のコアをハックする に示すデフォルトの config.m4 は、すこし複雑です。

Example#1 デフォルトの config.m4

```
dnl  $Id:  build.xml,v  1.3  2007/11/05  13:50:20  takagi  Exp  $
dnl  config.m4  for  extension  my_module

dnl  Comments  in  this  file  start  with  the  string  'dnl'.
dnl  Remove  where  necessary.  This  file  will  not  work
dnl  without  editing.

dnl  If  your  extension  references  something  external,  use  with:

dnl  PHP_ARG_WITH(my_module,  for  my_module  support,
dnl  Make  sure  that  the  comment  is  aligned:
dnl  [   --with-my_module                Include  my_module  support])

dnl  Otherwise  use  enable:

dnl  PHP_ARG_ENABLE(my_module,  whether  to  enable  my_module  support,
dnl  Make  sure  that  the  comment  is  aligned:
dnl  [   --enable-my_module              Enable  my_module  support])

if  test  "$PHP_MY_MODULE"  !=  "no";  then
   dnl  Write  more  examples  of  tests  here...

   dnl  # --with-my_module  ->  check  with-path
   dnl  SEARCH_PATH="/usr/local  /usr"       # you might want to change this
   dnl  SEARCH_FOR="/include/my_module.h"   # you  most  likely  want  to  change  this
   dnl  if  test  -r  $PHP_MY_MODULE/;  then  # path  given  as  parameter
   dnl     MY_MODULE_DIR=$PHP_MY_MODULE
   dnl  else  # search  default  path  list
   dnl     AC_MSG_CHECKING([for  my_module  files  in  default  path])
   dnl    for  i  in  $SEARCH_PATH  ;  do
   dnl       if  test  -r  $i/$SEARCH_FOR;  then
   dnl          MY_MODULE_DIR=$i
   dnl          AC_MSG_RESULT(found  in  $i)
  dnl       fi
   dnl    done
   dnl  fi
   dnl
   dnl  if  test  -z  "$MY_MODULE_DIR";  then
   dnl     AC_MSG_RESULT([not  found])
   dnl     AC_MSG_ERROR([Please  reinstall  the  my_module  distribution])
   dnl  fi

   dnl  # --with-my_module  ->  add  include  path
    dnl  PHP_ADD_INCLUDE($MY_MODULE_DIR/include)

   dnl  # --with-my_module  ->  chech  for  lib  and  symbol  presence
   dnl  LIBNAME=my_module  # you  may  want  to  change  this
   dnl  LIBSYMBOL=my_module  # you  most  likely  want  to  change  this

    dnl  PHP_CHECK_LIBRARY($LIBNAME,$LIBSYMBOL,
   dnl  [
    dnl     PHP_ADD_LIBRARY_WITH_PATH($LIBNAME,  $MY_MODULE_DIR/lib,  MY_MODULE_SHARED_LIBADD)
    dnl     AC_DEFINE(HAVE_MY_MODULELIB,1,[ ])
   dnl  ],[
    dnl     AC_MSG_ERROR([wrong  my_module  lib  version  or  lib  not  found])
   dnl  ],[
    dnl     -L$MY_MODULE_DIR/lib  -lm  -ldl
   dnl  ])
   dnl
    dnl  PHP_SUBST(MY_MODULE_SHARED_LIBADD)
```

```
    PHP_NEW_EXTENSION(my_module, my_module.c, $ext_shared)
fi
```

もし M4 ファイルにあまりなじみがないのなら（この機会に覚えてしまいましょう）、最初はこの例が難しく感じられるかもしれません。 しかし、実際はこれはとても簡単なものです。

注意: dnl で始まる行はすべてコメントであり、パースされません。

config.m4 ファイルの役割は、 configure に渡されたコマンドラインオプションを パースすることです。つまり、 必要な外部ファイルを読み込んで同じような設定タスクを行わなければならないということです。

デフォルトのファイルは、 configure スクリプトのオプションとして 2 つの設定ディレクティブ --with-my_module および --enable-my_module を作成します。 外部のファイルを参照している場合（例えば --with-apache ディレクティブが Apache のディレクトリを参照しているように）は最初のオプションを使用します。 拡張モジュールを有効にするかどうかを指定させるだけの場合には 2 番目のオプションを使用します。どちらを使用するかを決めたら、 使用しないほうを削除しなければなりません。つまり、 もし --enable-my_module を使用するのなら --with-my_module のサポートを削除しなければなりません。 逆もまた同様です。

デフォルトでは、ext_skel が作成した config.m4 ファイルは 両方のディレクティブを受けつけ、自動的に拡張モジュールを有効にします。 拡張モジュールを有効にする作業は、PHP_EXTENSION マクロで行われます。ユーザが （--enable-my_module あるいは --with-my_module を明示的に指定して）、 モジュールを組み込むように指示した場合にのみ PHP バイナリにモジュールを組み込むように変更するには、 $PHP_MY_MODULE のチェックを == "yes" に変更します。

```
if test "$PHP_MY_MODULE" == "yes"; then dnl
      Action.. PHP_EXTENSION(my_module, $ext_shared)
    fi
```

こうすると、PHP を再コンパイルするたびに --enable-my_module を使用しなければならなくなります。

注意: config.m4 を変更した後は、常に buildconf を実行してください!

設定スクリプトで使用可能な M4 マクロについては、後で詳細に説明します。 この段階では、デフォルトのファイルを使用することにします。

拡張モジュールの作成

まず最初に、非常に単純な拡張モジュールを作成してみましょう。この拡張モジュールは、 パラメータとして受け取った整数値をそのまま返すというだけの関数を実装しています。 ソースを Zend API: PHP のコアをハックする に示します。

Example#2 単純な拡張モジュール

```
/* 標準ヘッダを include します */
#include "php.h"

/* エクスポートする関数を宣言します */
ZEND_FUNCTION(first_module);

/* Zend がこのモジュールの内容を知るための、関数リスト */
zend_function_entry firstmod_functions[] =
{
     ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};

/* モジュールについての情報 */
zend_module_entry firstmod_module_entry =
{
     STANDARD_MODULE_HEADER,
    "First Module",
     firstmod_functions,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
     NO_VERSION_YET,
     STANDARD_MODULE_PROPERTIES
};

/* Zend にこのモジュールについて説明するための、標準「スタブ」ルーチンを実装します */
#if COMPILE_DL_FIRST_MODULE
ZEND_GET_MODULE(firstmod)
#endif

/* PHP で使用するための関数を実装します */
ZEND_FUNCTION(first_module)
{
    long parameter;

     if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &amp;parameter) == FAILURE) {
        return;
    }

     RETURN_LONG(parameter);
}
```

これは、PHP モジュールとして動作する完全なコードです。 ソースコードについてはあとですぐに説明しますが、 その前にまずビルド方法について説明しておきましょう（API についての議論の前に、とりあえず実際に動かしてみたいという 短気な人たちのためです）。

注意: この例のソースは、PHP 4.1.0 以降に含まれる Zend の新機能を使用しています。 PHP 4.0.x ではコンパイルできません。

モジュールのコンパイル

モジュールをコンパイルするには、基本的に 2 種類の方法があります。

- ext ディレクトリ内で、 提供されている "make" 機構を使用します。 これは、動的ロードモジュールを作成することも可能です。
- ソースを手動でコンパイルします。

最初の方法のほうが断然お勧めです。PHP 4.0 以降、 この仕組みは標準化され、洗練されたビルド手順に組み込まれています。 あまりに洗練されすぎているという事実は、裏を返せば欠点でもあります。 つまり、最初のうちはなかなか理解できないということです。 この章の後半ではより詳細に説

明しますが、 まず最初はデフォルトのファイルを使用して進めていきましょう。

2 番目の方法は、(何らかの理由で) PHP の完全なソースツリーを もっていない、すべてのファイルに対するアクセス権限がない、 あるいは単にキーボードと戯れていたいなどといった人たちにお勧めです。 これらはどれもめったにないケースではありますが、 この文書を完全なものにするためにこちらの方法についても説明しておきます。

Make を使用したコンパイル

サンプルソースを標準的な仕組みでコンパイルするには、 PHP ソースツリーの ext ディレクトリ以下に サンプルのディレクトリをコピーします。それから buildconf を実行すると、 新しい拡張モジュール用の適切なオプションを含む configure スクリプトが作成されます。 デフォルトでは、すべてのサンプルソースは無効になります。 そのため、この作業でビルド手順が破壊されてしまう恐れはありません。

buildconf を実行すると、configure --help の結果に以下が追加されます。

```
  --enable-array_experiments     BOOK: Enables array experiments
  --enable-call_userland          BOOK: Enables userland module
  --enable-cross_conversion       BOOK: Enables cross-conversion module
  --enable-first_module           BOOK: Enables first module
  --enable-infoprint               BOOK: Enables infoprint module
  --enable-reference_test         BOOK: Enables reference test module
  --enable-resource_test          BOOK: Enables resource test module
  --enable-variable_creation     BOOK: Enables variable-creation module
```

先ほど Zend API: PHP のコアをハックする で示したモジュールは、 --enable-first_module あるいは --enable-first_module=yes で使用可能になります。

手動でのコンパイル

モジュールを手動でコンパイルするには、以下のコマンドを実行しなければなりません。

作業	コマンド
コンパイル	cc -fpic -DCOMPILE_DL_FIRST_MODULE=1 -I/usr/local/include -I. -I.. -I../Zend -c -o <your_object_file> <your_c_file>
リンク	cc -shared -L/usr/local/lib -rdynamic -o <your_module_file> <your_object_file(s)>

モジュールをコンパイルするためのコマンドでは、コンパイラに対して 位置独立なコードを出力するように指示しています (-fpic は省略できません)。 また、定数 COMPILE_DL_FIRST_MODULE を指定することで、 モジュールのコードを動的ロード可能なモジュールとしてコンパイルするようにしています (上のテストモジュールではこれをチェックしています。これについては後で説明します)。 これらのオプションに続いて、ソースファイルをコンパイルするために必要な 最小限の標準インクルードパスを指定しています。

注意: サンプルのインクルードパスは、すべて ext ディレクトリからの相対パスです。 別の場所でコンパイルする場合は、それに応じてパスを変更してください。 必要なのは、PHP ディレクトリおよび Zend ディレクトリ、そして (必要に応じて) モジュールの存在するディレクトリです。

リンク用のコマンドも、動的モジュールとしてリンクするためのごく平凡なものです。

コンパイル用のコマンドに最適化オプションを含めることもできますが、 この例では省略しています (しかし、いくつかのオプションは 先に説明した makefile のテンプレートに含められています)。

注意: 手動でのコンパイルおよびリンクで PHP に静的モジュールとして組み込む手順は、非常に面倒なものとなります。 そのため、ここでは扱いません (これらのコマンドをすべてタイプするのは効率的ではありません)。

拡張モジュールの使用

選択したビルド方式によって、新しい PHP バイナリを Web サーバにリンクする (あるいは CGI として実行する) か .so (共有オブジェクト) ファイルを 作成するかのいずれかとなります。ファイル first_module.c を共有オブジェクトとしてコンパイルしたのなら、結果として first_module.so が出来上がります。これを 使用するために最初にしなければならないことは、出来上がったファイルを PHP からアクセス可能な位置に配置することです。とりあえず試してみるなら、 そのファイルを htdocs ディレクトリにコピーして Zend API: PHP のコアをハックする のソースで試してみましょう。 PHP バイナリに組み込む形式でコンパイルした場合は dl() のコールを省略します。そのモジュールの 機能はスクリプト内からすぐに利用可能です。

警告

セキュリティを確保するため、動的モジュールを公開ディレクトリに 配置してはいけません。公開ディレクトリに 配置することもできなくはないですし それによってテストも簡単にできますが、実運用環境では 別のディレクトリに配置すべきです。

Example#3 first_module.so をテストするためのファイル

```php
<?php

// 必要に応じて次の行のコメントを解除します
//  dl("first_module.so");

$param  = 2;
$return  = first_module($param);

print("'$param'  を送信すると、'$return'  が返されました");

?>
```

この PHP ファイルをコールした結果は、以下のようになります。

'2' を送信すると、'2' が返されました

必要なら、dl() 関数をコールすることによって 動的モジュールを読み込みます。この関数は 指定した共有オブジェクトを探し、それを読み込み、そして その関数を PHP から使用できるようにします。 このモジュールが提供する関数は **first_module()** で、ひとつのパラメータを受け取ってそれを整数に変換し、 変換結果を返します。

同じ結果が得られましたか? おめでとう! はじめての PHP 拡張のビルドがこれで完了しました。

トラブルシューティング

実際のところ、静的モジュールおよび動的モジュールをコンパイルする際に 行えるトラブルシューティングはそれほど多くありません。発生する 可能性のある唯一の問題は、コンパイラが「……が未定義」のような メッセージを出すことくらいです。このような場合は、 すべてのヘッダファイルが存在し、コンパイルコマンドでそれらへのパスを 正しく設定しているかどうかを確認しましょう。 すべてが正しく配置されていることを確実にするには、PHP ソースツリーを 新しく展開しなおし、ext ディレクトリ内に ファイルをコピーして自動ビルドを使用します。これにより、 安全なコンパイル環境が保証されます。これが失敗した場合には 手動でのコンパイルを試みます。

モジュール内で関数が見つからないというメッセージを PHP から受け取る こともあるかもしれません (もしサンプルソースを変更せずにそのまま 使用したのなら、これは起こりえません)。モジュール内から アクセスしようとしている外部関数にスペルミスがあれば、それは シンボルテーブルに "未リンクのシンボル" として残るでしょう。 PHP から動的に読み込んでリンクされる際に、これらは型エラーの ために読み込みに失敗します。本体のバイナリに対応するシンボルが 存在しないからです。このような場合は、モジュールファイルの中から 間違った宣言や外部参照を見つけ出します。この問題は、動的モジュール 固有のものであることに注意しましょう。静的モジュールとして 作成した場合は、この問題はコンパイル時に発覚します。

ソースコードについての議論

さあ、今やあなたは安全なビルド環境を手にいれ、 モジュールを PHP に組み込めるようになりました。 そろそろ全体像について話を始める時期でしょう。

モジュールの構造

すべての PHP モジュールは、共通の構造に従っています。

- ヘッダファイルのインクルード (必要なすべてのマクロ、API、define などをインクルードするため)
- エクスポートする関数の C での宣言 (Zend 関数ブロックの宣言のために必要)
- Zend 関数ブロックの宣言
- Zend モジュールブロックの宣言
- **get_module()** の実装
- エクスポートするすべての関数の実装

ヘッダファイルのインクルード

モジュールに組み込まなければならない唯一のヘッダファイルは php.h で、これは PHP ディレクトリにあります。 このファイルは、新しいモジュールを使用できるようにビルドするための すべてのマクロ定義、API 定義を含みます。

豆知識: モジュール固有の定義を含むヘッダファイルを、 別に分けて作成しておくとよいでしょう。 エクスポートされるすべての関数の定義をこのヘッダファイルに含め、 そしてこのファイルで php.h をインクルードします。 ext_skel を使用して雛形を作成したのなら、 すでにこの形式のヘッダファイルが出来上がっているはずです。

エクスポートする関数の宣言

エクスポートする (つまり PHP の新しいネイティブ関数として使用できるようにする) 関数を宣言するためには、Zend が提供するマクロを使用します。 宣言は、このように行います。

```
ZEND_FUNCTION ( my_function );
```

ZEND_FUNCTION は、Zend の内部 API を満たす新しい C 関数を宣言します。内部 API を満たすとは、 その関数の型が void であり、パラメータとして INTERNAL_FUNCTION_PARAMETERS (別のマクロ) を受け取るということです。さらに、関数名の先頭に zif を付加します。これらの定義を満たす宣言の例は、次のようになります。

```
void zif_my_function ( INTERNAL_FUNCTION_PARAMETERS );
```

INTERNAL_FUNCTION_PARAMETERS を展開した結果は、次のようになります。

```
void zif_my_function( int ht
                    , zval * return_value
                    , zval * this_ptr
                    , int return_value_used
                    , zend_executor_globals * executor_globals
                    );
```

インタプリタおよびエグゼキュータのコアは PHP 本体のパッケージとは分離されているので、 マクロや関数群を定義するもうひとつの API が作られました。これが Zend API です。今のところ Zend API は、 かつて PHP が担当していた機能のうちのごくわずかしか処理できません。 PHP の関数の多くは Zend のマクロに書き下ろされており、 その内部で Zend API をコールしています。 おすすめのやりかたは、まずできる限り Zend API を使用することです。 というのは古い API は互換性のためだけに残されているものだからです。 例えば、zval 型と pval 型はまったく同じものです。 zval は Zend での定義で、pval は PHP での定義です (実際のところ、現在 pval は zval のエイリアスとなっています)。INTERNAL_FUNCTION_PARAMETERS は Zend のマクロなので、 上の宣言には zval が用いられています。 コードを書く際には、新しい Zend API のために常に zval を使うようにしましょう。

この宣言のパラメータリストは非常に重要です。これらは常に頭に入れておくようにしましょう。 (Zend API: PHP のコアをハックする を参照ください)。

PHP からコールされた際に関数に渡される Zend のパラメータ

パラメータ	説明
ht	Zend の関数に渡す引数の数。これを直接操作してはいけません。値を取得する際には ZEND_NUM_ARGS() を使用してください。
return_value	この変数は、あなたの関数から PHP に返す値を渡すために使用します。 この変数にアクセスするには、定義済みマクロを使用するのがベストです。 マクロについては後で説明します。
this_ptr	この変数を使用すると、関数がオブジェクト内で使用されている場合に そのオブジェクトにアクセスすることができます。このポインタを取得するには 関数 **getThis()** を使用します。
return_value_used	このフラグは、この関数の返す値が実際に呼び出し元のスクリプトで使われるのかどうかを指定します。 *0* は、返り値が使われないことを表します。 *1* は、呼び出し元が返り値を期待していることを表します。 このフラグの値により、関数が正しく使用されているかどうかを確認します。また、 値を返す処理に負荷がかかる場合などに、このフラグによって速度の最適化を行います (*array.c* で、このような使用法を用いています)。
executor_globals	この変数は、Zend エンジンのグローバル設定へのポインタです。 これが便利なのは、例えば新しい変数を作成するときです (詳細はあとで説明します)。 エグゼキュータのグローバル設定は、マクロ

Zend 関数ブロックの宣言

エクスポートする関数の宣言が完了しました。こんどはさらにそれを Zend に登録しなければなりません。関数のリストを登録するには、zend_function_entry の配列を使用します。 この配列には、外部に公開するすべての関数について PHP で使用する場合の名前と C ソース内で定義されている名前が含まれています。 内部的には、zend_function_entry は Zend API: PHP のコアをハックする のように定義されています。

Example#4 内部での zend_function_entry の宣言

```
typedef struct _zend_function_entry {
     char *fname;
      void (*handler)(INTERNAL_FUNCTION_PARAMETERS);
     unsigned char *func_arg_types;
} zend_function_entry;
```

エントリ	説明
fname	PHP で使用する場合の名前を表します (例えば *fopen* や *mysql_connect* など。 今回の例では *first_module*)。
handler	この関数コールを処理する C 関数へのポインタ。 例えば、先ほど説明した標準マクロ *INTERNAL_FUNCTION_PARAMETERS* を参照ください。
func_arg_types	パラメータに対して、参照渡しを強制させるようにできます。 通常は NULL を設定しておくべきです。

上の例では、このような宣言になります。

```
zend_function_entry firstmod_functions[] =
{
      ZEND_FE(first_module, NULL)
     {NULL, NULL, NULL}
};
```

リストの最後のエントリは、必ず {NULL, NULL, NULL} でなければなりません。 エクスポートされる関数の一覧が終わることを Zend が知るために、 このエントリが必要となります。

> ***注意**: 最後を表す印として、定義済みマクロを使用することは できません。そうすると、 "NULL" という名前の関数を探しにいってしまいます!*

*マクロ ZEND_FE ('Zend Function Entry' を省略したものです) は、構造体のエントリを単純に zend_function_entry に展開します。 これらのマクロは特別な命名規約を持っていることに注意しましょう。 あなたが作成する C 関数の名前には、前に zif_ をつけることになります。つまり、ZEND_FE(first_module) は **zif_first_module()** という名前の C 関数を参照するということです。このマクロと手書きのエントリを混ぜて使用する場合には (お勧めしません) これを頭に入れておきましょう。*

*豆知識: **zif_*()** という名前の関数でコンパイルエラーが出た場合は、 ZEND_FE で定義した関数がかかわっています。*

Zend API: PHP のコアをハックする が、 関数の定義に使用できるすべてのマクロの一覧です。

関数定義用のマクロ

マクロ名	説明
ZEND_FE(name, arg_types)	zend_function_entry に name という名前の関数エントリを定義します。対応する C の関数が必要です。arg_types は *NULL* に設定しなければなりません。 この関数は、自動的に C の関数名を生成します。その名前は PHP の関数名の先頭に *zif_* をつけたものになります。 例えば *ZEND_FE("first_module", NULL)* とすると PHP の関数 **first_module()** を登録したことになり、 それを C の関数 **zif_first_module()** と関連付けます。*ZEND_FUNCTION* と組み合わせて使用します。
ZEND_NAMED_FE(php_name, name, arg_types)	php_name という名前で PHP の関数を定義し、 それを対応する C の関数 name に関連付けます。arg_types は *NULL* に設定しなければなりません。*ZEND_FE* のように自動的に名前を決められたくない場合にこの関数を使用します。*ZEND_NAMED_FUNCTION* と組み合わせて使用します。
ZEND_FALIAS(name, alias, arg_types)	alias という名前で、 name のエイリアスを定義します。arg_types は *NULL* に設定しなければなりません。対応する C の関数は不要です。その代わりに、エイリアスの対象を参照します。
PHP_FE(name, arg_types)	古い PHP の API で、*ZEND_FE* と同じものです。
PHP_NAMED_FE(runtime_name, name, arg_types)	古い PHP の API で、*ZEND_NAMED_FE* と同じものです。

注意: ZEND_FE と PHP_FUNCTION を組み合わせて使用したり、あるいは PHP_FE と ZEND_FUNCTION を組み合わせて使用したりすることはできません。しかし、 各関数について ZEND_FE と ZEND_FUNCTION、あるいは PHP_FE と PHP_FUNCTION という組み合わせが守られているのなら、 それらを混用することは可能です。 しかし、混用することは 推奨しません。 ZEND_* マクロだけを使用するようにしましょう。

Zend モジュールブロックの宣言

このブロックは構造体 zend_module_entry に保存され、モジュールの内容について Zend に示すために必要な すべての情報が含まれます。このモジュールの内部定義は Zend API: PHP のコアをハックする で確認できます。

Example#5 内部での zend_module_entry の宣言

```
typedef struct _zend_module_entry zend_module_entry;

     struct _zend_module_entry {
     unsigned short size;
     unsigned int zend_api;
     unsigned char zend_debug;
     unsigned char zts;
     char *name;
      zend_function_entry *functions;
       int (*module_startup_func)(INIT_FUNC_ARGS);
       int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
       int (*request_startup_func)(INIT_FUNC_ARGS);
       int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
       void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
     char *version;
```

[構造体の残りの部分は、ここではあまり関係がありません]

};

エントリ	説明
size, zend_api, zend_debug および zts	通常は *"STANDARD_MODULE_HEADER"* を指定します。 これは、4 つのメンバにそれぞれ zend_module_entory 全体のサイズ、*ZEND_MODULE_API_NO*、 デバッグビルドか通常ビルドのどちらであるか (*ZEND_DEBUG*) そして ZTS が有効かどうか (*USING_ZTS*) を代入します。
name	モジュール名を指定します (例えば *"File functions"*、*"Socket functions"*、 *"Crypt"* など)。この名前は、 phpinfo() の "Additional Modules" 欄で使用されます。
functions	先ほど説明した Zend 関数ブロックへのポインタ。
module_startup_func	この関数はモジュールの初期化時にコールされ、 最初の一度だけ行う初期化処理 (例えばメモリの確保など) で使用します。初期化に失敗した場合には *FAILURE*、 成功した場合には *SUCCESS* を返します。 このフィールドを使用しない場合は、*NULL* を指定します。関数を宣言するには、マクロ *ZEND_MINIT* を使用します。
module_shutdown_func	この関数はモジュールのシャットダウン時にコールされ、 最後に一度だけ行う後処理 (例えばメモリの開放など) で使用します。 これは **module_startup_func()** に対応するものです。 で使用します。後処理に失敗した場合には *FAILURE*、 成功した場合には *SUCCESS* を返します。 このフィールドを使用しない場合は、*NULL* を指定します。関数を宣言するには、マクロ *ZEND_MSHUTDOWN* を使用します。
request_startup_func	この関数はページがリクエストされるたびにコールされ、 リクエストを処理する際の前処理で使用します。 処理に失敗した場合には *FAILURE*、 成功した場合には *SUCCESS* を返します。 *注意:* 動的モジュールの場合は リクエストがあるまでは読み込まれないので、リクエストスタートアップ関数は、 モジュールスタートアップ関数の直後にコールされます (これら二つの初期化イベントが同時に発生します)。 このフィールドを使用しない場合は、*NULL* を指定します。関数を宣言するには、マクロ *ZEND_RINIT* を使用します。
request_shutdown_func	この関数はページのリクエストが終了するたびにコールされます。 ちょうど **request_startup_func()** に対応するものです。 処理に失敗した場合には *FAILURE*、 成功した場合には *SUCCESS* を返します。 *注意:* 動的モジュールの場合は リクエストがあるまでは読み込まれないので、 リクエストシャットダウン関数の直後に モジュールシャットダウンハンドラがコールされます (これら二つの後処理イベントが同時に発生します)。 このフィールドを使用しない場合は、*NULL* を指定します。関数を宣言するには、マクロ *ZEND_RSHUTDOWN* を使用します。

エントリ	説明
info_func	スクリプト内で phpinfo() がコールされると、 Zend は現在読み込まれているすべてのモジュールについてこの関数をコールします。 つまり、その出力結果に何らかの "あしあと" を残すチャンスが すべてのモジュールに与えられるわけです。一般的には、 これを使用して環境情報や東経情報を出力します。 このフィールドを使用しない場合は、*NULL* を指定します。関数を宣言するには、マクロ *ZEND_MINFO* を使用します。
version	モジュールのバージョン。バージョン番号をまだつけたくない場合は *NO_VERSION_YET* が使用できます。しかし、 何らかのバージョン文字列を指定することを推奨します。 バージョン文字列は、例えば次のようなものになります (バージョンの若い順に並べています): *"2.5-dev"*、*"2.5RC1"*、*"2.5"* あるいは *"2.5pl3"*。
それ以外の構造体の要素	これらは内部的に使用されるもので、マクロ *STANDARD_MODULE_PROPERTIES_EX* を使用して事前に設定されます。これらの要素に値を代入してはいけません。 *STANDARD_MODULE_PROPERTIES_EX* を使用するのは、 グローバルなスタートアップ関数、シャットダウン関数を使用する場合のみです。 それ以外の場合は *STANDARD_MODULE_PROPERTIES* を直接使用します。

今回の例では、この構造体を次のように実装します。

```
zend_module_entry  firstmod_module_entry  =
{
      STANDARD_MODULE_HEADER,
     "First Module",
      firstmod_functions,
     NULL, NULL, NULL, NULL, NULL,
     NO_VERSION_YET,
      STANDARD_MODULE_PROPERTIES,
};
```

これは、基本的に必要最小限の設定です。モジュール名は `First Module` ***とし、関数一覧の参照を設定し、 スタートアップ関数やシャットダウン関数はすべて未使用としています。***

参照用として、スタートアップ関数およびシャットダウン関数に関するマクロを *Zend API: PHP のコアをハックする* ***にまとめておきます。 これらは今回の例では使用しませんが、後で説明します。 スタートアップ関数やシャットダウン関数を宣言する際にはこれらのマクロを使用すべきです。 というのもこれらの関数には特別なパラメータ (*** `INIT_FUNC_ARGS` ***および*** `SHUTDOWN_FUNC_ARGS`***) を渡さなければならず、 定義済みマクロを使用することでこれらが自動的に関数宣言に組み込まれるからです。 仮にこれらの関数を (マクロを使用せずに) 手動で宣言したとしましょう。 もし*** `PHP` ***の開発者が何らかの事情で引数を変更したとすると、 それに追従するためにあなたは自分のモジュールのソースを変更しなければならなくなります。***

スタートアップ関数、シャットダウン関数を宣言するためのマクロ

マクロ	説明
ZEND_MINIT(module)	モジュールの開始時の関数を宣言します。 関数名は *zend_minit_<module>* (例えば *zend_minit_first_module*) のようになります。 *ZEND_MINIT_FUNCTION* と組み合わせて使用します。
ZEND_MSHUTDOWN(module)	モジュールのシャットダウン時の関数を宣言します。 関数名は *zend_mshutdown_<module>* (例えば *zend_mshutdown_first_module*) のようになります。 *ZEND_MSHUTDOWN_FUNCTION* と組み合わせて使用します。

ZEND_RINIT(module)	リクエストの開始時の関数を宣言します。 関数名は *zend_rinit_<module>* (例えば *zend_rinit_first_module*) のようになります。 *ZEND_RINIT_FUNCTION* と組み合わせて使用します。
ZEND_RSHUTDOWN(module)	リクエストのシャットダウン時の関数を宣言します。 関数名は *zend_rshutdown_<module>* (例えば *zend_rshutdown_first_module*) のようになります。 *ZEND_RSHUTDOWN_FUNCTION* と組み合わせて使用します。
ZEND_MINFO(module)	モジュール情報を出力する関数を宣言します。 phpinfo() がコールされた際に使用されます。 関数名は *zend_info_<module>* (例えば *zend_info_first_module*) のようになります。 *ZEND_MINFO_FUNCTION* と組み合わせて使用します。

get_module() の作成

これは特別な関数で、すべての動的読み込みモジュールで使用されます。 これを作成するため、まずは ZEND_GET_MODULE を見てみましょう。

```
#if  COMPILE_DL_FIRSTMOD
       ZEND_GET_MODULE(firstmod)
#endif
```

関数の実装が、条件付きコンパイル文で囲まれています。なぜかというと、 get_module() 関数が必要となるのは あなたのモジュールが動的モジュールとしてビルドされる場合だけだからです。 コンパイラのコマンドで COMPILE_DL_FIRSTMOD を定義することにより (動的モジュールとしてビルドするための コンパイル手順については上記を参照ください)、 動的モジュールとしてビルドするのか組み込みモジュールとしてビルドするのかを 指示することができます。組み込みモジュールを作成する場合は、 get_module() の実装は単純に取り除かれます。

get_module() は、 Zend がモジュールを読み込む際にコールされます。 例えば、スクリプト内で dl() がコールされた場合などです。この関数の目的は、モジュール情報ブロックを Zend に返し、モジュールの内容をエンジンに教えることです。

動的モジュールで get_module() 関数を実装しなかった場合、 Zend がそのモジュールにアクセスしようとした際にエラーとなります。

エクスポートするすべての関数の実装

あとは、エクスポートする関数を実装すれば終わりです。 first_module では、例としてこのような関数を使用します。

```
ZEND_FUNCTION(first_module)
{
     long parameter;

      if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &amp;parameter) == FAILURE) {
         return;
    }

      RETURN_LONG(parameter);
}
```

関数の宣言には ZEND_FUNCTION を使用します。 これは、関数エントリテーブルにおける ZEND_FE に対応しています (こちらについては先ほど説明しました)。

関数宣言の後には、引数のチェックおよびその内容の取得、 引数の変換、そして返り値の作成などのコードが続きます (これらの詳細は後述します)。

まとめ

基本的に、これですべてです。PHP モジュールを構成するのに、 これ以上のものは必要ありません。組み込みのモジュールについても その構造は動的モジュールと同じです。 ここまでに説明した内容を身に着けておけば、今後 PHP モジュールのソースファイルを読んでいく際につまづくこともなくなるでしょう。

さあ、これ以降の節で PHP の内部構造について学び、 強力な拡張モジュールを作っていきましょう。

引数の扱い

言語を拡張する際の最も重要な問題のひとつは、 引数として渡されるデータの扱いです。たいていの拡張モジュールは、 何らかの入力データを扱う (あるいはパラメータを受け取って 特定の動作を行う) ように作られています。そして、 PHP と C 言語の間でデータをやり取りするための唯一の方法が 関数の引数となります。 事前に定義したグローバル値を使用してデータを交換することももちろん可能ですが (この方法についても後述します)、いろんな意味でこの方法は避けるべきです。 これはまったくお勧めできない手段です。

PHP では、正式に関数を宣言 (declare) することはありません。 そのため、関数コールの構文は完全に動的なものとなり、 エラーチェックは行われません。 コール方法が正しいかどうかを調べるのは、ユーザが書くコードの役割となります。 例えば、ある関数に対して引数をひとつだけ指定してコールした後、 同じ関数に対して 4 つの引数を指定してコールすることも可能です。 どちらのコールについても、文法的にはまったく正しいものとなります。

引数の数の定義

PHP が正式な関数宣言を行わないためにコール時の文法チェックが行われない、 また PHP が可変引数をサポートしているなどの理由で、 実際にその関数にいくつの引数が渡されたのかを知らなければならないこともあるでしょう。 そのような場合には ZEND_NUM_ARGS マクロを使用します。 以前のバージョンの PHP では、コール時の引数の数を取得するために このマクロは関数のハッシュテーブルのエントリ ht を使用していました。このエントリは INTERNAL_FUNCTION_PARAMETERS のリストから渡されました。 関数に渡された引数の数は今では ht 自体に含まれているので、 ZEND_NUM_ARGS はダミーのマクロとなっています (実際の定義は zend_API.h を参照ください)。 しかし、今後のことを考えると、 呼び出しインターフェイスが変わっても互換性を保ち続けられるように このマクロを使用しておくことをお勧めします。 注意: 昔の PHP では、このマクロと同等の働きをするのは ARG_COUNT マクロでした。

引数の数が正しいかどうかを調べるコードは次のようになります。

```
if(ZEND_NUM_ARGS()  !=  2)  WRONG_PARAM_COUNT;
```

関数コール時の引数の数が 2 つでなかった場合、 これはエラーメッセージを表示して終了します。 上のコードは、WRONG_PARAM_COUNT マクロの使用例でもあります。 これは、以下のような標準的なエラーメッセージを生成するために使用するものです。

```
"Warning:  Wrong  parameter  count  for  firstmodule()  in  /home/www/htdocs/firstmod.php  on  line  5"
```

このマクロはデフォルトのエラーメッセージを表示し、呼び出し元に制御を戻します。 このマクロの定義もまた zend_API.h にあります。このような内容です。

```
ZEND_API  void  wrong_param_count(void);

#define  WRONG_PARAM_COUNT  {  wrong_param_count();  return;  }
```

ご覧の通り、このマクロは **wrong_param_count()** という名前の内部関数をコールしています。 この関数が警告を表示しています。独自のエラーメッセージを作成するための方法については、 後半の節「情報の表示」を参照ください。

引数の取得

> **注意**: **パラメータのパース用の新しい API** この章では、Andrei Zmievski による新しい Zend パラメータパース用 API を説明します。この API は PHP 4.0.6 から PHP 4.1.0 の間の開発中に導入されました。

パラメータのパースはあまりにもありふれた操作であり、少し退屈に感じることもあるでしょう。 また、エラーチェックやエラーメッセージは標準化されていたほうがいいでしょう。 PHP 4.1.0 以降では、パラメータのパース用の新しい API を使用することでこれが実現できます。 この API はパラメータの受け取りを劇的に単純化していますが、 可変引数を受け取る関数には使用できないという弱点があります。 しかし、大半の関数では、引数の数は固定です。そのため、 新しい標準として、このパース用 API の使用を推奨します。

パラメータのパース用関数のプロトタイプは、このようになります。

```
int  zend_parse_parameters(int  num_args  TSRMLS_DC,  char  *type_spec,  ...);
```

最初の引数に、あなたの関数が受け取るパラメータの数を指定します。 ここでは ZEND_NUM_ARGS() が使用できます。 2 番目のパラメータは、常に TSRMLS_CC マクロでなければなりません。3 番目の引数は、 あなたの関数が受け取る引数の数および型を表す文字列です。 これは、printf の書式指定文字列によって出力内容を指定するのに似ています。 そして最後に、パラメータの値を受け取る変数へのポインタを指定します。

常に希望通りの型でデータを受け取ることができるよう、 **zend_parse_parameters()** は可能な範囲で型変換を行います。 あらゆるスカラー型は別のスカラー方に変換することが可能です。 しかし、複雑な型 (配列、オブジェクトあるいはリソース) とスカラー型の間の型変換はできません。

パラメータの取得に成功し、かつ型変換でエラーが発生しなかった場合は この関数は SUCCESS を返します。それ以外の場合は FAILURE を返します。また、 「パラメータの数が一致しない」「型変換ができなかった」 などの情報を含むエラーメッセージを出力します。

出力されるエラーメッセージは、例えば次のようなものになります。

```
Warning  -  ini_get_all()  requires  at  most  1  parameter,  2  given
Warning  -  wddx_deserialize()  expects  parameter  1  to  be  string,  array  given
```

もちろん、これらのメッセージにはエラーの発生したファイル名や行番号も含まれています。

型を指定する文字の一覧をここにまとめます。

- l - long
- d - double
- s - string (null バイトの可能性もあり) およびその長さ
- b - boolean
- r - zval* に保存されたリソース
- a - zval* に保存された配列
- o - zval* に保存された (あらゆるクラスの) オブジェクト
- O - zval* に保存された (クラスエントリで指定されているクラスの) オブジェクト
- z - zval* 自体

以下の文字は、型指定文字列の中で特別な意味を持ちます。

- | - 残りのパラメータがオプションであることを表します。 これらのパラメータに対応する保存用変数は、 拡張モジュール自身によってデフォルト値で初期化されなければなりません。 なぜなら、パラメータが渡されていなければパース関数を通過しないからです。
- / - パラメータの後にこの文字を続けると、 そのパラメータに対して **SEPARATE_ZVAL_IF_NOT_REF()** をコールします。これにより、そのパラメータが参照でなければパラメータのコピーが作成されます。
- ! - パラメータの後にこの文字を続けると、 そのパラメータは指定した型あるいは NULL となります (a、o、O、r および z についてのみ適用可能)。 NULL 値が渡された場合は、 保存ポインタの値が NULL に設定されます。

この関数の使用法について説明するには、 実際の例を見ていただくのがいちばんでしょう。

```
/* long、文字列とその長さ、そして zval を受け取ります。*/
long  l;
char  *s;
```

```
int  s_len;
zval  *param;
if  (zend_parse_parameters(ZEND_NUM_ARGS()  TSRMLS_CC,
                                "lsz", &l, &s, &s_len, &param) == FAILURE) {
     return;
}

/* my_ce で指定するクラスのオブジェクト、そしてオプションで double 値を受け取ります。*/
zval  *obj;
double  d  =  0.5;
if  (zend_parse_parameters(ZEND_NUM_ARGS()  TSRMLS_CC,
                                "O|d", &obj, my_ce, &d) == FAILURE) {
     return;
}

/* オブジェクト (あるいは null)、そして配列を受け取ります。
    オブジェクトに null が渡された場合、obj は NULL となります。*/
zval  *obj;
zval  *arr;
if  (zend_parse_parameters(ZEND_NUM_ARGS()  TSRMLS_CC,  "O!a",  &obj,  &arr)  ==  FAILURE)  {
     return;
}

/* 配列のコピーを受け取ります。*/
zval  *arr;
if  (zend_parse_parameters(ZEND_NUM_ARGS()  TSRMLS_CC,  "a/",  &arr)  ==  FAILURE)  {
     return;
}

/* 最初の 3 つのパラメータのみを受け取ります (可変引数の関数の場合に有用です)。*/
zval  *z;
zend_bool  b;
zval  *r;
if  (zend_parse_parameters(3,  "zbr!",  &z,  &b,  &r)  ==  FAILURE)  {
     return;
}
```

最後の例で、**ZEND_NUM_ARGS()** を使用せずにパラメータ数を 3 としていることに注目しましょう。 こうすることで、可変引数の関数に対して最低限必要な引数の数を指定することができます。 もちろん、もし残りのパラメータについても処理したいのなら **zend_get_parameters_array_ex()** を使用してそれを取得しなければなりません。

拡張版のパース関数も存在します。 これは、追加のフラグを引数に指定することでその動きを制御します。

```
int  zend_parse_parameters_ex(int  flags,  int  num_args  TSRMLS_DC,  char  *type_spec,  ...);
```

現在は、渡すことのできるフラグは ZEND_PARSE_PARAMS_QUIET だけです。これを指定すると、処理中に発生したエラーメッセージを出力しないようになります。 これは、その関数がさまざまな形式の引数を受け取ることを想定しており、 独自のエラーメッセージで対応したい場合などに有用です。

例えば、3 つの long 値あるいは 3 つの文字列を受け取る関数は、 このようになります。

```
long  l1,  l2,  l3;
char  *s;
if  (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                ZEND_NUM_ARGS() TSRMLS_CC,
                                "lll", &l1, &l2, &l3) == SUCCESS) {
     /* long の場合 */
}  else  if  (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                        ZEND_NUM_ARGS(), "s", &s, &s_len) == SUCCESS) {
     /* 文字列の場合 */
} else {
      php_error(E_WARNING,  "%s()  takes  either  three  long  values  or  a  string  as  argument",
                    get_active_function_name(TSRMLS_C));
     return;
}
```

上で説明したいずれの場合についても、パラメータの取得処理は慎重に行いましょう。 これ以外の例については、PHP に同梱されている拡張モジュールのソースを 参考にしてください。便利な使用法がいろいろ発見できるでしょう。

引数の取得の古い方法 (廃止予定)

> **注意**: **廃止予定のパラメータパース用 API** この API は非推奨です。現在は新しい ZEND パラメータパース API が使用されています。

引数の数をチェックし終えたら、次は引数そのものにアクセスしなければなりません。 そのためには **zend_get_parameters_ex()** の助けを借りることになります。

```
zval  **parameter;

if(zend_get_parameters_ex(1,  &parameter)  !=  SUCCESS)
    WRONG_PARAM_COUNT;
```

すべての引数は zval コンテナに格納され、 二重にポイントされる必要があります。 上のコードは、引数を取得してそれをポインタ parameter でアクセスできるようにしています。

zend_get_parameters_ex() は、少なくとも 2 つの引数を受け付けます。 最初の引数は、取得する引数の数です (これは、 関数がコールされた際の引数の数と同じでなければなりません。 そのため、呼び出し構文をきちんとチェックすることが大切になります)。 2 番目 (およびそれ以降) の引数は、zval へのポインタへのポインタへのポインタ (なんてややこしいことでしょう!) です。 これが必要になるのは、私たちが作成した関数内のローカル **zval を管理するために Zend は内部で **zval を使用しており、 **zend_get_parameters_ex()** はそれに対するポインタを必要とするからです。

zend_get_parameters_ex() の返り値は SUCCESS あるいは FAILURE で、それぞれ (当然のごとく) 成功したこと、あるいは引数の処理に失敗したことを示します。 もっともありがちな失敗の原因は、パラメータの数を間違えることです。 この場合は、 WRONG_PARAM_COUNT で関数を抜けなければなりません。

複数の引数を受け取るには、このようにします。

```
zval  **param1,  **param2,  **param3,  **param4;

if(zend_get_parameters_ex(4,  &param1,  &param2,  &param3,  &param4)  !=  SUCCESS)
      WRONG_PARAM_COUNT;
```

zend_get_parameters_ex() がエラーとなるのは、 実際の数より多くのパラメータを取得しようとした場合のみです。 もし実際の関数が 5 つの引数でコールされたのに **zend_get_parameters_ex()** では 3 つしか取得しなかった場合、 何もエラーは発生せず、最初の 3 つのパラメータが取得されます。 続けて **zend_get_parameters_ex()** をコールしたとしても、 残りの引数が取得できるわけではなく最初と同じものが得られるだけです。

可変引数 / オプションパラメータの扱い

もしあなたの関数が可変引数を受け付けるのなら、 さきほどの例が次善の策となることもあるでしょう。ただ、 取りうる可能性のあるすべての引数の数に対して、それぞれ **zend_get_parameters_ex()** コールしなければならず、 不満が残ります。

このような場合には、 **zend_get_parameters_array_ex()** 関数を使うことができます。 これは、引数の数を指定してデータを取得し、それを配列に格納します。

```
zval  **parameter_array[4];

/* 引数の数を取得します */
argument_count  =  ZEND_NUM_ARGS();

/* 引数の数の範囲 (最低 2 個、最大 4 個) */
/* を満たしているかどうかを調べます       */
if(argument_count  <  2  ||  argument_count  >  4)
      WRONG_PARAM_COUNT;

/* 引数の数が正しかったので、その内容を取得します */
if(zend_get_parameters_array_ex(argument_count,  parameter_array)  !=  SUCCESS)
      WRONG_PARAM_COUNT;
```

まず、引数の数を調べてそれが予期する範囲内にあることを確かめます。 それから **zend_get_parameters_array_ex()** を使用し、 引数の値へのポインタを parameter_array に格納します。

これを非常にうまく実装したのが、PHP の fsockopen() を処理するコードです。このコードは ext/standard/fsock.c にあり、 Zend API: PHP のコアをハックする で見ることができます。 このソースの中に知らない関数が含まれていたとしても、現時点では問題ありません。 すぐにわかるようになります。

Example#6 fsockopen() における、PHP の可変引数処理の実装

```
pval  **args[5];
int  *sock=emalloc(sizeof(int));
int  *sockp;
int  arg_count=ARG_COUNT(ht);
int socketd = -1;
unsigned char udp = 0;
struct timeval timeout = { 60, 0 };
unsigned short portno;
unsigned long conv;
char *key = NULL;
FLS_FETCH();

if (arg_count  >  5  ||  arg_count  <  2  ||  zend_get_parameters_array_ex(arg_count,args)==FAILURE)  {
      CLOSE_SOCK(1);
      WRONG_PARAM_COUNT;
}

switch(arg_count)  {
     case 5:
           convert_to_double_ex(args[4]);
           conv = (unsigned long) (Z_DVAL_PP(args[4]) * 1000000.0);
          timeout.tv_sec = conv / 1000000;
          timeout.tv_usec = conv % 1000000;
         /* fall-through */
     case 4:
          if (!PZVAL_IS_REF(*args[3])) {
                php_error(E_WARNING,"error string argument to fsockopen not passed by reference");
        }
           pval_copy_constructor(*args[3]);
           ZVAL_EMPTY_STRING(*args[3]);
         /* fall-through */
     case 3:
          if (!PZVAL_IS_REF(*args[2])) {
                php_error(E_WARNING,"error argument to fsockopen not passed by reference");
             return;
        }
          ZVAL_LONG(*args[2], 0);
         break;
}

convert_to_string_ex(args[0]);
convert_to_long_ex(args[1]);
portno  =  (unsigned  short)  Z_LVAL_P(args[1]);

key  =  emalloc(Z_STRLEN_P(args[0])  +  10);
```

fsockopen() が受け付ける引数の数は、 2、3、4、あるいは 5 です。お決まりの変数宣言の後に、 この関数は引数の数が範囲内にあるかどうかを調べます。 それから、switch() 文の下に流れていく性質を利用して、 すべての引数を処理します。switch() 文は、 まず引数の数が最大 (5 個) であった場合の処理から始まります。 その後、引数の数が 4 個であった場合の処理、3 個であった場合の処理、 と順にたどっていきます。これは、キーワード break の記述を省略しているからです。最後の処理を実行した後で switch() 文は終了し、 関数の引数が 2 個だけであった場合でも、必要最小限の処理が実行されます。

このように複数ステージの処理を段階的に実行するようにすると、 可変引数の処理がやりやすくなります。

引数へのアクセス

引数にアクセスするには、すべての引数の型がきちんと定義されていなければなりません。 何度も言いますが、PHP は究極の動的言語なので、時に問題が起こることがあります。 PHP は一切の型チェックを行わないので、 どんな型のデータでも関数に渡すことができてしまいます。 本当は整数値を期待しているのに配列が渡されるかもしれないし、 その逆だってありえます。PHP は、こんな場合でも一切なにも通知しません。

これを防ぐには、これらの API を使用して引数の方を強制的に変換しなければなりません (Zend API: PHP のコアをハックする を参照ください)。

注意: すべての変換関数のパラメータは **zval です。

引数の変換関数

関数	説明
convert_to_boolean_ex()	Boolean 型への強制的な変換を行います。Boolean 値が渡された場合は何もしません。Long、double および *0* を含む文字列、そして NULL 値は Boolean *0* (FALSE) となります。配列やオブジェクトは、その元になっているエントリやプロパティの数を基準にして変換されます。空の配列や空のオブジェクトは FALSE、それ以外は TRUE となります。その他の値は、すべて Boolean *1* (TRUE) になります。
convert_to_long_ex()	long 型 (デフォルトの整数型) への強制的な変換を行います。NULL 値、Boolean、リソースおよび long はそのまま何

	もしません。 double は切り詰められます。整数値を含む文字列は対応する数値表現に変換され、 それ以外の文字列は *0* になります。 配列やオブジェクトは、中身が空の場合に *0*、それ以外の場合に *1* となります。
convert_to_double_ex()	double 型 (デフォルトの浮動小数点数値型) への強制的な変換を行います。 NULL 値、Boolean、リソース、 long、そしてもちろん double はそのまま何もしません。数値を含む文字列は対応する数値表現に変換され、 それ以外の文字列は *0.0* になります。 配列やオブジェクトは、中身が空の場合に *0.0*、 それ以外の場合に *1.0* とな

	ります。
convert_to_string_ex()	文字列への強制的な変換を行います。文字列が渡された場合は何もしません。NULL 値は空の文字列に変換されます。Boolean TRUE は *"1"*、それ以外の Boolean は空の文字列となります。long および double はそれぞれ対応する文字列表現に変換されます。 配列は *"Array"*、オブジェクトは *"Object"* という文字列に変換されます。
convert_to_array_ex(value)	配列への強制的な変換を行います。配列が渡された場合は何もしません。 オブジェクトは、すべてのプロパティが配列に変換されます。 プロパティ名が配列のキー、プロパティの内容が配列の値となりま

	す。 NULL 値は空の配列に変換されます。それ以外のすべての値は、 キー *0* に対応する値として元の値を格納した配列となります。
convert_to_object_ex(value)	オブジェクトへの強制的な変換を行います。オブジェクトが渡された場合は何もしません。 NULL 値は空のオブジェクトに変換されます。配列は、そのキーをプロパティに、 その値をプロパティの内容として保持するオブジェクトに変換されます。その他の型は、すべてプロパティ *scalar* をもつオブジェクトに変換されます。このプロパティの内容は、変換前の値となります。
convert_to_null_ex(value)	NULL 値、つまり空白になるように変換しま

す。

注意: これらの振る舞いのデモが、付録 CD-ROM の cross_conversion.php で見られます。

PHP の型変換の挙動

引数に対してこれらの関数を使用することで、 渡されたデータの型の安全性を確保できます。 渡された型が要求と異なった場合、PHP は空の値（空の文字列、配列、オブジェクトや 数値の 0、Boolean の FALSE など）を結果として返します。

これは、先ほど説明したサンプルモジュールのコードの一部を引用したものです。 ここで実際に変換関数を使用しています。

```
zval  **parameter;

if((ZEND_NUM_ARGS()  !=  1)  ||  (zend_get_parameters_ex(1,  &parameter)  !=  SUCCESS))
{
      WRONG_PARAM_COUNT;
}

convert_to_long_ex(parameter);

RETURN_LONG(Z_LVAL_P(parameter));
```

パラメータへのポインタを取得した後に、パラメータの値が long 型（整数値）に変換されます。そしてそれがこの関数の返り値となります。 この値の中身にアクセスするには、zval 型についての知識が必要です。その定義を Zend API: PHP のコアをハックする に示します。

Example#7 PHP/Zend zval 型定義

```
typedef  pval  zval;

typedef  struct  _zval_struct  zval;

typedef  union  _zvalue_value  {
     long lval;                    /* long 値 */
     double dval;                  /* double 値 */
     struct {
          char *val;
          int len;
     } str;
     HashTable *ht;               /* ハッシュテーブル値 */
     struct {
           zend_class_entry *ce;
           HashTable *properties;
     } obj;
}  zvalue_value;

struct  _zval_struct  {
     /* 変数の情報 */
     zvalue_value value;          /* 値 */
     unsigned char type;          /* アクティブな型 */
      unsigned char is_ref;
      short refcount;
};
```

実際のところは pval（php.h で定義）は単なる zval（zend.h で定義）のエイリアスであり、これは _zval_struct をさしています。 そこで、この構造体に注目してみましょう。 _zval_struct は「マスタ」構造体であり、 その中には値情報、型情報、参照情報が含まれています。 中に含まれている zvalue_value は共用体であり、 変数の値がそこに含まれます。変数の型に応じて、共用体の中の適切なメンバーにアクセスする必要があります。 それぞれの構造体については Zend API: PHP のコアをハックする、 Zend API: PHP のコアをハックする および Zend API: PHP のコアをハックする を参照ください。

Zend zval 構造体

項目	説明
value	この変数の内容を含む共用体。詳細は Zend API: PHP のコアをハックする を参照ください。

type	変数の型を含みます。使用可能な型については Zend API: PHP のコアをハックする を参照ください。
is_ref	この変数が参照ではない場合に 0、他の変数への参照である場合に 1 となります。
refcount	この変数に対する参照の数。この変数に格納されている値への新しい参照が作成されるたび

に、カウンタが 1 増加します。参照が解除されるたびに、カウンタが 1 減少します。参照カウンタが 0 になった段階で、この値はどこからも参照されていないことになります。この時点で、自動的にメモリが解放されます。

Zend zvalue_value 構造体

項目	説明
lval	変数の型が *IS_LONG*、*IS_BOOLEAN* あるいは *IS_RESOURCE* である場合にこのプロパティを使用します。
dval	変数の型が *IS_DOUBLE* である場合にこのプロパティを使用します。
str	型が *IS_STRING* の変数にアクセスする際に、この構造体を使用します。 len には文字列の長さが含まれ、 val が文字列へのポインタとなります。Zend は C の文字列を使用しているので、文字列の長さには、最後の *0x00* のぶんも含まれます。
ht	変数が配列である場合に、この項目は配列のハッシュテーブルエントリへのポインタとなります。
obj	変数の型が *IS_OBJECT* である場合にこのプロパティを使用します。

Zend 変数の型を表す定数

定数	説明
IS_NULL	NULL (空白) 値を表します。
IS_LONG	long (整数) 値。
IS_DOUBLE	double (浮動小数点) 値。
IS_STRING	文字列。

IS_ARRAY	配列を表します。
IS_OBJECT	オブジェクト。
IS_BOOL	Boolean 値。
IS_RESOURCE	リソース (リソースについては、以下の適切なセクションを参照ください)。
IS_CONSTANT	定数 (定義済み) 値。

long 値にアクセスするには zval.value.lval、 double 値にアクセスするには zval.value.dval、といったように使用します。 すべての値は共用体に保存されるので、不適切なメンバーにアクセスすると、 無意味な結果を得ることになります。

配列やオブジェクトへのアクセスは少し複雑なので、後で説明します。

参照渡しされた引数の扱い

参照渡しの引数を受け取って関数内部でそれを変更しようとする場合は、 少々注意が必要です。

敢えて説明しませんでしたが、ここで示している状況では、 関数のパラメータとして受け取った zval に対する書き込み権限はありません。 もちろん関数内で独自に作成した zval を変更することは可能ですが、 Zend の内部データを参照している zval は、 決して変更してはいけません。

これまで説明してきたのは、いわゆる ***_ex()** 系の API ばかりです。お気づきかもしれませんが、これまで使用してきた API 関数は **zend_get_parameters()** ではなくて **zend_get_parameters_ex()** でしたし、また **convert_to_long()** ではなくて **convert_to_long_ex()** でした。これらの ***_ex()** 系の関数は、いわゆる「拡張」Zend API と呼ばれるものです。 これらは、以前の API に比べて速度が少し向上しているのですが、 それと引き換えに読み込み専用のアクセスしかできないようになっています。

Zend は内部的には参照を使用しているので、 さまざまな変数が同じ値を参照することもありえます。 zval コンテナへの書き込みアクセスをするためには、 このコンテナが保持する値が他と完全に分離していること、 つまり他のコンテナから参照されていないことが必要です。 zval コンテナが他のコンテナから参照されている場合に その zval を変更すると、この zval を参照しているその他のコンテナの内容も変わってしまいます (それらも変更後の値を指すようになるからです)。

zend_get_parameters_ex() は、単に zval コンテナへのポインタを返します。そこに参照が含まれているかどうかは考慮しません。 一方、これに対応する伝統的な API である **zend_get_parameters()** は、参照をチェックします。 参照が見つかった場合には、独立した zval コンテナを新しく作成し、参照先のデータをそこにコピーし、 その新しく作成した (他とは分離している) コンテナへのポインタを返します。

この操作のことを zval separation (zval の分離) (あるいは pval separation) と言います。***_ex()** API は zval の分離を行わないません。そのために大幅に高速化されましたが、 その代償として書き込みアクセスができなくなっています。

とは言うものの、パラメータを変更するには書き込みアクセスをしなければなりません。 このような場合のために、Zend には特別な方法が用意されています。 関数のパラメータが参照渡しされた場合は、自動的に zval の分離が行われるのです。 つまり、PHP で下のように関数をコールすると、$parameter が独立した値として渡されることを Zend が自動的に保証してくれるのです。 これにより、書き込みが可能となります。

```
my_function(&amp;$parameter);
```

しかし、これは値渡しのパラメータには適用されません! 参照渡し以外で渡されたパラメータ以外は、読み込み専用となります。

これらの性質により、パラメータを扱う際にはそれが参照なのかそうでないのかをしっかり見極める必要があります。 さもないと、思ってもいない結果を引き起こすことになります。 パラメータが参照渡しされたかどうかを調べるためには、マクロ PZVAL_IS_REF を使用します。このマクロは zval* を受け取り、それが参照かそうでないかを返します。 Zend API: PHP のコアをハックする に実際の使用例があります。

Example#8 パラメータが参照で渡されたかどうかを調べる

```
zval  *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
     return;

/* パラメータが参照で渡されたかどうかを調べます */
if (!PZVAL_IS_REF(parameter)) {
{
     zend_error(E_WARNING, "パラメータは参照渡しされませんでした");
     RETURN_NULL();
}

/* パラメータを変更します */
ZVAL_LONG(parameter, 10);
```

その他のパラメータの書き込みについての安全性の保障

zend_get_parameters_ex() で取得したパラメータのうち、 参照渡しではないものについても値を変更したくなる場合があるかもしれません。 そんなときには、マクロ SEPARATE_ZVAL を使用します。 これは、指定したコンテナについて zval の分離を行います。 新しく作成された zval は内部のデータとは分離されており、 ローカルスコープでしか使用できません。つまり、 スクリプト全体のコンテキストに影響を与えることなく データを変更したり破壊したりできるようになるのです。

```
zval  **parameter;

/* パラメータを取得します */
zend_get_parameters_ex(1,  &parameter);

/* この時点では、<parameter> はまだ            */
/* Zend の内部データバッファと紐付いています */

/* <parameter> を書き込み可能にします */
SEPARATE_ZVAL(parameter);

/* この時点で、<parameter> が変更できるようになります。*/
/* グローバルに影響を与えることはありません            */
```

SEPARATE_ZVAL は、emalloc() を使用して新しい zval コンテナを確保します。 つまり、このメモリの開放を忘れてしまったとしても、 スクリプトの終了時に自動的に破棄されるということです。 しかし、コンテナを開放せずにこのマクロをコールし続けると、 RAM が散らかってしまいます。

注意: 書き込み権限の問題については、 「伝統的な」API（zend_get_parameters() やその他） を使用すれば簡単に回避できます。しかし、この API は非推奨のようなので、今後この章では深入りしません。

変数の作成

あなたの作成する拡張モジュールが PHP スクリプトとの間でデータ交換を行うにあたって、 もっとも重要な問題は変数の作成です。この節では、PHP がサポートする変数の型を扱う方法を示します。

概要

実行中のスクリプトによって「外部から」見える変数を新しく作成するには、 まず新しい zval コンテナを確保してそこに値を格納し、 それを Zend の内部シンボルテーブルに登録しなければなりません。 これは、変数を作成する際のお決まりの手順です。

```
zval  *new_variable;

/* 新しいコンテナを確保して初期化します */
MAKE_STD_ZVAL(new_variable);

/* 型や値をここで設定します。これ以降の節を参照ください */

/* この変数を "new_variable_name" という名前でシンボルテーブルに登録します */
ZEND_SET_SYMBOL(EG(active_symbol_table),  "new_variable_name",  new_variable);

/* これで、$new_variable_name という名前でスクリプトからアクセスできるようになります */
```

マクロ MAKE_STD_ZVAL は、ALLOC_ZVAL を使用して新しい zval コンテナを確保し、 INIT_ZVAL でそれを初期化します。 これを書いている時点の Zend の実装では、初期化 というのは参照カウンタを 1 にして is_ref フラグをクリアすることを指します。 しかし、これは今後拡張される可能性があります。そのため、 ALLOC_ZVAL を直接使用するのではなく MAKE_STD_ZVAL を用いるようにしておくことが大切です。 実行速度を最適化したい場合（そして、ここで明示的に zval コンテナを初期化する必要がない場合）は ALLOC_ZVAL を使用することも可能です。 しかし、データの整合性を保証できなくなるため、 この方法は推奨されません。

ZEND_SET_SYMBOL の役割は、新しい変数を Zend のシンボルテーブルに登録することです。このマクロは、 その値が既にシンボルテーブルに存在するかどうかを調べ、 もし存在すれば新しいシンボルを参照に変換します（古い zval コンテナが使用していたメモリは、 自動的に開放されます)。速度を気にする必要がない場合には この方法がお勧めです。メモリの使用量を抑えることができます。

ZEND_SET_SYMBOL は、マクロ EG 経由で Zend executor のグローバル変数を使用することに注意しましょう。 EG(active_symbol_table) を指定することで、 現在アクティブなシンボルテーブルを、アクティブなローカルスコープで扱えるようになります。 ローカルスコープは、その関数が関数内でコールされたかどうかによって異なります。

とにかく速度を最適化したい、メモリの消費量はあまり気にしない という場合には、同じ値が既存の変数に登録されているかどうかの チェックを省略することができます。その代わりに、 zend_hash_update() を使用して 強制的にシンボルテーブルに挿入します。

```
zval  *new_variable;

/* 新しいコンテナを確保して初期化します */
MAKE_STD_ZVAL(new_variable);

/* 型や値をここで設定します。これ以降の節を参照ください */

/* この変数を "new_variable_name" という名前でシンボルテーブルに登録します */
zend_hash_update(
      EG(active_symbol_table),
      "new_variable_name",
      strlen("new_variable_name") + 1,
      &amp;new_variable,
      sizeof(zval *),
     NULL
);
```

これは、ほとんどのモジュールで使用されている標準的な方法です。

上の例で作成された変数は、常にローカルスコープにあります。 つまり、その関数がコールされたコンテキスト内に存在するということです。 新しい変数をグローバルスコープに作成するにも同じメソッドを使用しますが、 別のシンボルテーブルを参照します。

```
zval  *new_variable;

// 新しいコンテナを確保して初期化します
MAKE_STD_ZVAL(new_variable);

//
// 型や値をここで設定します
```

```
//

// この変数を "new_variable_name" という名前でグローバルシンボルテーブルに登録します
ZEND_SET_SYMBOL(&amp;EG(symbol_table), "new_variable_name", new_variable);
```

今回は、EG(symbol_table) によって参照される本体のグローバルシンボルテーブルを使用して ZEND_SET_SYMBOL マクロがコールされています。

注意: 変数 active_symbol_table はポインタですが、symbol_table はそうではありません。 これは、 EG(active_symbol_table) および &EG(symbol_table) を ZEND_SET_SYMBOL へのパラメータとして使用する必要があるからです - ここにはポインタが必要です。

同様に、より効率的なバージョンとして、 シンボルテーブルの更新をハードコーディングすることもできます。

```
zval *new_variable;

// 新しいコンテナを確保して初期化します
MAKE_STD_ZVAL(new_variable);

//
// 型や値をここで設定します
//

// この変数を "new_variable_name" という名前でグローバルシンボルテーブルに登録します
zend_hash_update(
      &amp;EG(symbol_table),
      "new_variable_name",
      strlen("new_variable_name") + 1,
      &amp;new_variable,
     sizeof(zval *),
     NULL
);
```

Zend API: PHP のコアをハックする は、ふたつの変数を作成する例です。 local_variable はローカルスコープ、そして global_variable はグローバルスコープとなります (図 9.7 を参照ください)。完全な例は CD-ROM にあります。

注意: グローバル変数は、実際には関数内からアクセスできないことにお気づきでしょう。 これは、PHP ソース内で global $global_variable; のようにローカルスコープにインポートしていないからです。

Example#9 スコープが異なる変数の作成

```
ZEND_FUNCTION(variable_creation)
{
      zval *new_var1, *new_var2;

       MAKE_STD_ZVAL(new_var1);
       MAKE_STD_ZVAL(new_var2);

       ZVAL_LONG(new_var1, 10);
       ZVAL_LONG(new_var2, 5);

        ZEND_SET_SYMBOL(EG(active_symbol_table), "local_variable", new_var1);
        ZEND_SET_SYMBOL(&amp;EG(symbol_table), "global_variable", new_var2);

      RETURN_NULL();

}
```

Long (整数型)

さあ、それではデータを変数に代入していきましょう。まずは long 型からです。long は PHP の整数型で、非常にシンプルな形式で保存されます。この章の最初のほうで説明した zval.value コンテナの 構造を見てみましょう。long 型のデータは、共用体の lval フィールドに直接格納されることがおわかりでしょう。 long 型に対応する type の値は IS_LONG です (Zend API: PHP のコアをハックする を参照ください)。

Example#10 long の作成

```
zval *new_long;

MAKE_STD_ZVAL(new_long);

new_long-&gt;type = IS_LONG;
new_long-&gt;value.lval = 10;
```

あるいは、マクロ ZVAL_LONG を使用することもできます。

```
zval *new_long;

MAKE_STD_ZVAL(new_long);
ZVAL_LONG(new_long, 10);
```

Double (浮動小数点型)

double 型は PHP の浮動小数点型で、long 型と同様に簡単に代入できます。 この値もまた共用体に直接格納されるからです。 zval.value コンテナで対応するメンバは dval です。また、対応する type は IS_DOUBLE となります。

```
zval *new_double;

MAKE_STD_ZVAL(new_double);

new_double-&gt;type = IS_DOUBLE;
new_double-&gt;value.dval = 3.45;
```

あるいは、マクロ ZVAL_DOUBLE を使用することもできます。

```
zval *new_double;

MAKE_STD_ZVAL(new_double);
ZVAL_DOUBLE(new_double, 3.45);
```

文字列

文字列についてはもう少し手間を掛けなければなりません。 先に説明したように、Zend の内部データ構造に関連付けられる すべての文字列は、Zend 自身のメモリ管理関数を使用して管理しなければなりません。 静的な文字列を参照したり、標準関数を使用して割り当てた文字列を使用することはできません。 文字列を代入するには、zval.value コンテナの構造体 str にアクセスしなければなりません。 対応する type は IS_STRING です。

```
zval  *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);

new_string-&gt;type = IS_STRING;
new_string-&gt;value.str.len = strlen(string_contents);
new_string-&gt;value.str.val = estrdup(string_contents);
      </programlisting>
      ここでの、Zend の <function>estrdup</function> の使用法に注意しましょう。
      もちろん、定義済みのマクロ <literal>ZVAL_STRING</literal> を使用することも可能です。
      <programlisting>
zval  *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);
ZVAL_STRING(new_string, string_contents, 1);
```

ZVAL_STRING は、3 番目のパラメータをとることができます。 これは、指定した文字列が（**estrdup()** を使用して）複製されるべきかどうかを指定します。このパラメータに 1 を指定すると、文字列が複製されます。 0 の場合は、渡されたポインタを そのまま変数の内容として使用します。これは、すでに Zend の内部メモリに割り当てられている文字列を参照する変数を 作成する場合に便利です。

ある場所で文字列を切り捨てたい場合、 あるいは文字列の長さが事前にわかっている場合は、ZVAL_STRINGL(zval, string, length, duplicate) を使用して 新しい文字列の長さを明示的に指定することができます。 このマクロは ZVAL_STRING より高速に動作し、 バイナリセーフです。

空の文字列を作成するには、文字列の長さを 0 にしたうえで、中身に empty_string を使用します。

```
new_string-&gt;type = IS_STRING;
new_string-&gt;value.str.len = 0;
new_string-&gt;value.str.val = empty_string;
```

もちろん、これを行うためのマクロもあります (ZVAL_EMPTY_STRING)。

```
MAKE_STD_ZVAL(new_string);
ZVAL_EMPTY_STRING(new_string);
```

論理型

論理型の作り方は long 型と同じです。ただ、type は IS_BOOL となります。 lval に指定できる値は 0 および 1 です。

```
zval  *new_bool;

MAKE_STD_ZVAL(new_bool);

new_bool-&gt;type = IS_BOOL;
new_bool-&gt;value.lval = 1;
```

この型に対応するマクロは ZVAL_BOOL (値を指定できます)、そして ZVAL_TRUE および ZVAL_FALSE (それぞれ、値を明示的に TRUE および FALSE に設定します) です。

配列

配列は、Zend の内部ハッシュテーブルに格納されます。 このハッシュテーブルにアクセスするには **zend_hash_*()** API を使用します。配列を作成するたびに、新しいハッシュテーブルの ハンドルが必要となります。これは、zval.value コンテナのメンバ ht に格納されます。

単に配列を作成するためだけの API があります。これは非常に便利です。 新しい配列を開始するには、**array_init()** をコールします。

```
zval  *new_array;

MAKE_STD_ZVAL(new_array);

array_init(new_array);
```

array_init() は、常に SUCCESS を返します。

配列に新しい要素を追加するには、 やりたいことに応じてさまざまな関数を使用できます。 Zend API: PHP のコアをハックする、 Zend API: PHP のコアをハックする および Zend API: PHP のコアをハックする で、これらの関数について説明しています。これらの関数はすべて、 失敗した場合に FAILURE、 成功した場合に SUCCESS を返します。

連想配列用の Zend の API

関数	説明
add_assoc_long(zval *array, char *key, long n);()	*long* 型の要素を追加します。
add_assoc_unset(zval *array, char *key);()	未設定の要素を追加します。
add_assoc_bool(zval *array, char *key, int b);()	Boolean 要素を追加します。
add_assoc_resource(zval *array, char *key, int r);()	リソースを配列に追加します。
add_assoc_double(zval *array, char *key, double d);()	浮動小数点値を追加します。
add_assoc_string(zval *array, char *key, char *str, int duplicate);()	文字列を配列に追加します。フラグ duplicate は、 文字列の内容を Zend の内部メモリにコピーする必要があるかどうかを指定します。
add_assoc_stringl(zval *array, char *key, char *str, uint length, int duplicate); ()	長さ length の文字列を配列に追加します。 それ以外は **add_assoc_string()** と同じです。
add_assoc_zval(zval *array, char *key, zval *value);()	zval を配列に追加します。別の配列やオブジェクト、ストリームなどを追加する際に便利です。

数値添字配列用の Zend の API その 1

関数	説明
add_index_long(zval *array, uint idx, long n);()	*long* 型の要素を追加します。
add_index_unset(zval *array, uint idx);()	未設定の要素を追加します。
add_index_bool(zval *array, uint idx, int b);()	Boolean 要素を追加します。
add_index_resource(zval *array, uint idx, int r);()	リソースを配列に追加します。
add_index_double(zval *array, uint idx, double d);()	浮動小数点値を追加します。
add_index_string(zval *array, uint idx, char *str, int duplicate);()	文字列を配列に追加します。フラグ duplicate は、 文字列の内容を Zend の内部メモリにコピーする必要があるかどうかを指定します。
add_index_stringl(zval *array, uint idx, char *str, uint length, int duplicate);()	長さ length の文字列を配列に追加します。 この関数は高速でバイナリセーフです。 それ以外は **add_index_string()** と同じです。
add_index_zval(zval *array, uint idx, zval *value);()	zval を配列に追加します。別の配列やオブジェクト、ストリームなどを追加する際に便利です。

数値添字配列用の Zend の API その 2

関数	説明
add_next_index_long(zval *array, long n);()	*long* 型の要素を追加します。
add_next_index_unset(zval *array);()	未設定の要素を追加します。
add_next_index_bool(zval *array, int b);()	Boolean 要素を追加します。
add_next_index_resource(zval *array, int r);()	リソースを配列に追加します。
add_next_index_double(zval *array, double d);()	浮動小数点値を追加します。
add_next_index_string(zval *array, char *str, int duplicate);()	文字列を配列に追加します。フラグ duplicate は、 文字列の内容を Zend の内部メモリにコピーする必要があるかどうかを指定します。
add_next_index_stringl(zval *array, char *str, uint length, int duplicate);()	長さ length の文字列を配列に追加します。 この関数は高速でバイナリセーフです。 それ以外は **add_index_string()** と同じです。
add_next_index_zval(zval *array, zval *value);()	zval を配列に追加します。別の配列やオブジェクト、ストリームなどを追加する際に便利です。

これらの関数はすべて、Zend 内部のハッシュ API を抽象化して使用しやすくしたものです。 もちろん、ハッシュ関数を直接使用することも可能です。例えば、すでに割り当て済みの zval コンテナを配列に挿入する場合などが考えられます。 これを行うには、連想配列の場合は ***zend_hash_update()*** *(Zend API: PHP のコアをハックする を参照ください)、 数値添字配列の場合は* ***zend_hash_index_update()*** *(Zend API: PHP のコアをハックする を参照ください) を使用します。*

Example#11 連想配列への要素の追加

```
zval  *new_array,  *new_element;
char  *key  =  "element_key";

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

array_init(new_array);

ZVAL_LONG(new_element,  10);

if(zend_hash_update(new_array-&gt;value.ht,  key,  strlen(key)  +  1,  (void  *)&amp;new_element,  sizeof(zval  *),  NULL)  ==  FAI
{
    // エラー処理をここで行います
}
```

Example#12 数値添字配列への要素の追加

```
zval  *new_array,  *new_element;
int key = 2;

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

array_init(new_array);

ZVAL_LONG(new_element,  10);

if(zend_hash_index_update(new_array-&gt;value.ht,  key,  (void  *)&amp;new_element,  sizeof(zval  *),  NULL)  ==  FAILURE)
{
    // エラー処理をここで行います
}
```

add_next_index_*() の機能をエミュレートするには、 これを使用します。

```
zend_hash_next_index_insert(ht,  zval  **new_element,  sizeof(zval  *),  NULL)
```

注意: 関数から配列を返すには、まず **array_init()** を使用し、それ以降の操作は定義済み変数 return_value で行います (エクスポートする関数への引数として渡します。 呼び出しインターフェイスについての先ほどの議論を参照ください)。 ここで MAKE_STD_ZVAL を使用してはいけません。

Tip: 毎回 new_array->value.ht を書く手間を省くためには HASH_OF(new_array) を使用します。 これは、互換性やコーディングスタイルの観点からもお勧めです。

オブジェクト

オブジェクトは配列に変換できる (その逆も可能) ので、 PHP の配列と多くの共通点があるであろうことはお気づきでしょう。 オブジェクトを処理するには、同じようなハッシュ関数を使用しますが、 オブジェクトを作成する際には異なる API を使用します。

オブジェクトを初期化するには、関数 **object_init()** を使用します。

```
zval  *new_object;

MAKE_STD_ZVAL(new_object);

if(object_init(new_object)  !=  SUCCESS)
{
    // エラー処理をここで行います
}
```

Zend API: PHP のコアをハックする で説明している関数を使用すると、 オブジェクトにメンバを追加することができます。

オブジェクトを作成するための Zend の API

関数	説明
add_property_long(zval *object, char *key, long l);()	long 型をオブジェクトに追加します。
add_property_unset(zval *object, char *key);()	未設定のプロパティをオブジェクトに追加します。
add_property_bool(zval *object, char *key, int b);()	Boolean をオブジェクトに追加します。
add_property_resource(zval *object, char *key, long r);()	リソースをオブジェクトに追加します。
add_property_double(zval *object, char *key, double d);()	double 型をオブジェクトに追加します。
add_property_string(zval *object, char *key, char *str, int duplicate);()	文字列をオブジェクトに追加します。
add_property_stringl(zval *object, char *key, char *str, uint length, int duplicate);()	指定した長さの文字列をオブジェクトに追加します。この関数は add_property_string() より高速で、バイナリセーフです。
add_property_zval(zval *obect, char *key, zval *container):()	*zval* コンテナをオブジェクトに追加します。これは、整数値や文字列のような単純なものではなく 配列やその他のオブジェクトなどをプロパティとして追加する際に有用です。

リソース

リソースは、PHP における特殊なデータ型です。 リソース (resources) は何らかの特定の型を表すわけではなく、 さまざまな情報を扱うための抽象化した方法を表しています。 リソースは、Zend の内部では特別なリストの中に保持されます。 リストの各エントリは、そのリソースが指すデータを表す型定義を持っています。 Zend は、内部でリソースを扱う際には、常にこれを使用します。 リソースに直接アクセスすることはできず、提供されている API を通じてアクセスしなければなりません。特定のリソースに対する 参照がすべて失われると、対応するシャットダウン関数がすぐにコールされます。

例えば、リソースは、 データベースへのリンクやファイル記述子を格納するために使用されます。 事実上の標準実装となっているのは MySQL モジュールです。しかし、例えば Oracle モジュールのような その他のモジュールでもリソースを使用しています。

> **注意**: 実際のところ、あなたが関数内で処理したい任意のデータへのポインタ (例: 構造体へのポインタ) を、リソースとして使用することができます。 ユーザがこのデータにアクセスするには、ひとつのリソース変数を 関数に渡すだけでいいようになります。

新しいリソースを作成するには、そのリソースを開放するためのハンドラを 登録しなければなりません。リソースにはあらゆる種類のデータを保存できるので、 Zend は、それが不要になった際にどのように開放するのかを知っておく必要があるのです。 これを実現するために、自分が作成したリソース開放ハンドラを Zend に登録します。これは、(手動・自動にかかわらず) リソースを開放することになった際に、Zend によってコールされます。 リソースハンドラを Zend に登録すると、そのリソースについての リソース型ハンドルが Zend から返されます。 このハンドルは、後でこの型のリソースにアクセスする際には常に必要となり、 たいていは拡張モジュール内のグローバルな静的変数として保存されます。 スレッドセーフであるかどうかについて心配する必要はありません。 なぜなら、リソースハンドラの登録は モジュールの初期化時に一度行うだけだからです。

リソースハンドラを登録するための Zend の関数は、次のように宣言されています。

```
ZEND_API int zend_register_list_destructors_ex(rsrc_dtor_func_t ld, rsrc_dtor_func_t pld, char *type_name, int module_nu
```

この関数には、二種類の異なるリソース破壊ハンドラを渡すことが可能です。 それぞれ、通常のリソース用のものと持続的なリソース用のものになります。 持続的なリソースとは、例えばデータベース接続などに使用されるものです。 リソースを登録する際には、これらのどちらかのハンドラを必ず指定しなければなりません。 もう一方のほうには、単に NULL を渡すようにします。

zend_register_list_destructors_ex() は、次のパラメータを受け取ります。

ld	通常のリソース用のリソース破壊ハンドラコールバック。
pld	持続的なリソース用のリソース破壊ハンドラコールバック。
type_name	そのリソースの名前を表す文字列。 PHP 内で一意になるような名前をつけるよう心がけましょう。 ユーザが例えば *var_dump($resource);* をコールした際に、この名前が表示されます。
module_number	*module_number* は、*PHP_MINIT_FUNCTION* 関数の中で自動的に使用可能となります。 そのため、単純にそれを渡すだけです。

返り値は、作成した リソース型 の一意な ID となる整数値です。

リソース破壊ハンドラ (通常版および持続的リソース版のどちらも) のプロトタイプは次のようになります。

```
void resource_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC);
```

rsrc に渡されるのは、次のような構造体へのポインタです。

```
typedef struct _zend_rsrc_list_entry {

    void *ptr;
    int type;
    int refcount;

} zend_rsrc_list_entry;
```

メンバ void *ptr が、 リソースへの実際のポインタとなります。

これでやるべきことがわかりました。Zend に登録したいリソースを、 実際に定義してみましょう。 ここでは、ふたつの整数型メンバからなる単純な構造体を考えます。

```
typedef struct {

    int resource_link;
    int resource_type;

} my_resource;
```

このリソースを破壊するハンドラは、おそらく次のようなものとなるでしょう。

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    // たいていは void ポインタを構造体の型にキャストすることになるでしょう。

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;

    // ここで、そのリソースに対して必要な作業を行います。たとえば
    // ファイルやソケットを閉じたり、内部で新たに確保したメモリを開放したりなどです。
    // もちろん、このリソース自身が使っているメモリを開放することも忘れないようにしましょう。
```

```
        do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}
```

注意: 注意すべき点: もしそのリソースが複雑な構造体であり、 その内部で実行時に確保したメモリへのポインタを含んでいる場合は、 リソース自身のメモリを開放する 前に 実行時に確保したメモリを開放しなければなりません。

ここまでで、

1. どんなリソースなのか
2. そのリソースを破壊する際のハンドラ

の定義が終了しました。それでは残りの作業を進めましょう。

1. 拡張モジュール内のグローバル変数を作成し、 リソース ID を保持させる。必要に応じて、 これは全ての関数からアクセス可能となる
2. リソース名を定義する
3. リソース破壊ハンドラを記述する
4. そしてそのハンドラを登録する

```
// 拡張モジュール内のどこかで、登録したリソース用の変数を定義します。
 // 'le' って何のことだって? 単に 'list entry' を略しただけです。
  static int le_myresource;

// リソース名もどこかで定義しておくとよいでしょう。
  #define le_myresource_name   "My type of resource"

 [...]

// 実際にリソース破壊ハンドラを登録します。
  void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

       my_resource *my_rsrc = (my_resource *) rsrc->ptr;
         do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}

 [...]

  PHP_MINIT_FUNCTION(my_extension) {

       // 'module_number' は、PHP_MINIT_FUNCTION() 関数の
    // 定義で既に提供されていることに注意しましょう。

        le_myresource = zend_register_list_destructors_ex(my_destruction_handler, NULL, le_myresource_name, module_numbe

    // 別のリソースを登録したり、グローバル変数や定数を
    // 初期化したりします。
}
```

実際に新しいリソースを登録するには、 **zend_register_resource()** 関数あるいは **ZEND_REGISTER_RESOURE()** マクロのいずれかを使用します。 これらはどちらも zend_list.h で定義されています。 それぞれの引数は一対一対応しているので、 将来の互換性を考えると常にマクロを使用するようにすることをお勧めします。

```
int ZEND_REGISTER_RESOURCE(zval *rsrc_result, void *rsrc_pointer, int rsrc_type);
```

rsrc_result	これは、事前に初期化済みの zval * コンテナです。
rsrc_pointer	保存したいリソースへのポインタ。
rsrc_type	リソース破壊ハンドラを登録した際に受け取る型。 命名規約に従うなら、これは le_myresource となります。

返り値は、そのリソースに対応する一意な整数値になります。

新しいリソースを登録する際に実際には何が行われているのかというと、 まずそれが Zend の内部リストに挿入されます。そして、 結果は単に与えられた zval * コンテナに保存されます。

```
      rsrc_id = zend_list_insert(rsrc_pointer, rsrc_type);

     if (rsrc_result) {
          rsrc_result->value.lval = rsrc_id;
          rsrc_result->type = IS_RESOURCE;
    }

     return rsrc_id;
```

返される rsrc_id は、 新しく登録されたリソースを表す一意な識別子となります。 マクロ RETURN_RESOURE を使用して、 これを利用者に返すことができます。

```
      RETURN_RESOURCE(rsrc_id)
```

注意: そのリソースをすぐに利用者に返したいのなら、 return_value を zval * コンテナに設定するのが一般的な方法です。

Zend はこれ以降、このリソースに対するすべての参照を管理できるようになります。 リソースへの参照がすべて失われると、リソースに対して事前に定義した デストラクタがすぐにコールされます。この設定の利点は、 あなたのモジュール内で発生するメモリリークを気にしなくてもすむということです。 呼び出し元スクリプトで割り当てているすべてのメモリはリソースを参照しています。 メモリが必要なくなったとスクリプトが判断した時点で、 Zend はすぐにそれを検出して通知してくれます。

さて、利用者がリソースを取得したあとで、 それをまたあなたの作成した関数に渡してきたとしましょう。 zval * コンテナ内の value.lval にはあなたのリソースのキーが含まれているので、 それを使用して次のマクロでリソースを取得することができます。 ZEND_FETCH_RESOURCE:

ZEND_FETCH_RESOURCE(rsrc, rsrc_type, rsrc_id, default_rsrc_id, resource_type_name, resource_type)

rsrc	これは、あなたが事前に登録したリソースを指すポインタです。
rsrc_type	これは、ポインタに対する型キャスト引数、例えば myresource * などです。
rsrc_id	これは、利用者があなたの関数に渡した zval * コンテナのアドレスです。例えば zval *z_resource が指定された場合には &z_resource となります。
default_rsrc_id	これは、リソースが取得できなかった場合は -1 が返された場合などのための デフォルトのリソース ID です。
resource_type_name	これは、要求されたリソースの名前です。 リソースが見つからなかった場合や無効なリソースだった場合に、 意味のあるエラーメッセージを作成するためにこの文字列を使用します。
resource_type	リソース破壊ハンドラを登録した際に返すリソース型です。今回の例では、これは le_myresource でした。

このマクロは何も値を返しません。これは開発者の利便性のためで、 TSRMLS 引数のことを考慮したものです。リソースを読み込む段階でチェックが行われます。 リソースの取得時にエラーが発生した場合は、このマクロは 警告メッセージをスローし、PHP 関数は NULL を返します。

リソースをリストから強制的に削除するには、関数 **zend_list_delete()** を使用します。 また事前に割り当てた値への新たな参照を作成した場合 (例えば、デフォルトのデータベースリンクを再利用する場合) などは、強制的に参照カウンタを増加させることができます。 このような場合は、関数 **zend_list_addref()** を使用します。事前に割り当てたリソースエントリを探すには **zend_list_find()** を使用します。 これらの完全な API は、zend_list.h で確認できます。

自動グローバル変数の作成のためのマクロ

先ほど説明したマクロに加え、 単純なグローバル変数を簡単に作成するためのマクロがあります。 これらは、例えばグローバルなフラグなどを作成する際に知っておくと便利です。 あまり行儀のよいやり方ではありませんが、 Zend API: PHP のコアをハックする で説明するマクロはまさにこの作業を行うためのものです。 これらは zval を確保する必要がありません。 単純に、変数名と値を渡すだけでいいのです。

グローバル変数の作成のためのマクロ

マクロ	説明
SET_VAR_STRING(name, value)	新しい文字列を作成します。
SET_VAR_STRINGL(name, value, length)	新しい文字列を、指定した長さで作成します。このマクロは SET_VAR_STRING より高速で、バイナリセーフです。
SET_VAR_LONG(name, value)	新しい long を作成します。
SET_VAR_DOUBLE(name, value)	新しい double を作成します。

定数の作成

Zend は（通常の変数ではなく）真の定数の作成もサポートしています。 定数へのアクセスにはドル記号（$）の接頭辞が不要で、すべてのスコープで使用可能です。 例としては TRUE や FALSE などがあります。

独自の定数を作成するには、 Zend API: PHP のコアをハックする のマクロを使用します。 これらのマクロは、指定した名前と値の定数を作成します。

各定数に対して、フラグを指定することもできます。

- CONST_CS - この定数名は、大文字小文字を区別して扱われます。
- CONST_PERSISTENT - この定数は持続的に扱われ、この定数を保持しているプロセスが 終了した後も「忘れられる」ことはありません。

これらのフラグは、OR で組み合わせて使用します。

```
// "long" 型の新しい定数を登録します。
      REGISTER_LONG_CONSTANT("NEW_MEANINGFUL_CONSTANT", 324, CONST_CS |
    CONST_PERSISTENT);
```

マクロには、ふたつの形式があります。 REGISTER_*_CONSTANT および REGISTER_MAIN_*_CONSTANT です。 最初の形式は、定数を現在のモジュールにバインドします。 これらの定数は、そのモジュールがメモリから開放されるとすぐに、 シンボルテーブルから削除されます。二番目の形式が作成する定数は、 モジュールとは独立してシンボルテーブルに残り続けます。

定数を作成するためのマクロ

マクロ	説明
REGISTER_LONG_CONSTANT(name, value, flags) REGISTER_MAIN_LONG_CONSTANT(name, value, flags)	long 型の新しい定数を登録します。
REGISTER_DOUBLE_CONSTANT(name, value, flags) REGISTER_MAIN_DOUBLE_CONSTANT(name, value, flags)	double 型の新しい定数を登録します。
REGISTER_STRING_CONSTANT(name, value, flags) REGISTER_MAIN_STRING_CONSTANT(name, value, flags)	文字列型の新しい定数を登録します。指定した文字列は、Zend の内部メモリ上に存在しなければなりません。

REGISTER_STRINGL_CONSTANT(name, value, length, flags) *REGISTER_MAIN_STRINGL_CONSTANT(name, value, length, flags)*	文字列型の新しい定数を登録します。文字列の長さを、明示的にlengthと指定します。指定した文字列は、Zendの内部メモリ上に存在しなければなりません。

変数の内容の複製: コピーコンストラクタ

遅かれ早かれ、zval コンテナの内容を別のコンテナに 代入したい場面が訪れるでしょう。これは口で言うほど簡単ではありません。 というのも zval コンテナには、 型情報だけではなく Zend の内部データへの参照も含まれているからです。 例えば、配列やオブジェクトの場合は、その大きさに応じて 多くのハッシュテーブルエントリが入れ子構造になっています。 ある zval を別のコンテナに代入すると、 ハッシュテーブルのエントリを複製するのではなく（単に）その参照が複製されるだけになります。

このような複雑な形式のデータをコピーするには、 コピーコンストラクタ を使用します。 コピーコンストラクタは、一般的には 演算子のオーバーロードをサポートしている言語で定義されており、 複雑な型をコピーするために使用されています。 そのような言語でオブジェクトを定義すると、 "=" 演算子をオーバーロードできるようになります。この演算子の一般的な役割は、 rvalue（演算子の右側を評価した結果）の内容を lvalue（同じく左側の結果）に代入することです。

オーバーロード とはこの演算子に別の意味を割り当てることです。 通常は、演算子に対して関数コールを割り当てるために用いられます。 プログラム内でそのようなオブジェクトに対してこの演算子が使用されると、 lvalue と rvalue をパラメータとして関数がコールされます。 この関数内で、パラメータの情報を使用して "=" 演算子にさせたい動作を行います（通常は、コピーを拡張した動作になるでしょう）。

PHP の zval コンテナについても、この 「コピーを拡張した動作」が必要になります。 配列の場合は、この拡張コピー動作によって 配列に関連するすべてのハッシュテーブルの内容を新しく作成するようにします。 また文字列の場合は適切なメモリの確保が必要になります。

*Zend では、これを行うための関数として **zend_copy_ctor()** を提供しています（以前の PHP では、同じ機能を持つ **pval_copy_constructor()** という関数がありました）。*

一番わかりやすい例として、 「複雑な型を引数として受け取ってそれを変更し、変更した内容を返す」 という関数を考えてみましょう。

```
zval  *parameter;

if  (zend_parse_parameters(ZEND_NUM_ARGS()  TSRMLS_CC,  "z",  &amp;parameter)  ==  FAILURE)
    return;
}

// ここでパラメータに何らかの変更を加えます

// 変更した後のコンテナを返します
*return_value  =  *parameter;
zval_copy_ctor(return_value);
```

この関数の最初の部分は、単に引数を取得しているだけのごく普通の処理です。 しかし、（省略している部分で）変更を加えた後が面白いところです。 parameter のコンテナが、（事前に定義済みの）return_value コンテナに代入されます。 ここで、コンテナの中身をうまく複製するためにコピーコンストラクタがコールされます。 コピーコンストラクタは、与えられた引数に対して直接動作します。 標準的な返り値は、 失敗した場合に FAILURE、 成功した場合に SUCCESS となります。

この例でコピーコンストラクタのコールを省略した場合、 parameter と return_value の両方が同じ内部データを指すことになり、return_value は同じデータ構造への無効な参照ということになってしまいます。 parameter が指すデータに変更を加えると、それが return_value にも影響を及ぼしてしまいます。 つまり、別の複製を作るためにはコピーコンストラクタの使用が必須であるということです。

*Zend API において、コピーコンストラクタに対応するデストラクタは **zval_dtor()** です。これはコンストラクタと反対の動作をします。*

値を返す

あなたの関数から PHP に値を返す方法については、既に簡単に説明しました。 ここではより詳細に説明します。返り値は return_value 変数を用いて渡されます。この変数は関数の引数として渡されたものです。 return_value 引数は zval コンテナで構成されており（前の章の呼び出しインターフェイスのところを参照ください）、 自由に変更することができます。コンテナ自身はすでに割り当て済みなので、 改めて MAKE_STD_ZVAL を実行する必要はありません。 その内容に直接アクセスすることが可能です。

関数から返す値を作成しやすくするため、そして zval コンテナの内部構造にアクセスしやすくするために、 いくつかのマクロが定義されています。 Zend API: PHP のコアをハックする および Zend API: PHP のコアをハックする にまとめたこれらのマクロは、対応する型と値を自動的に設定します。

注意: Zend API: PHP のコアをハックする のマクロは、自動的に関数から値を return します。一方 Zend API: PHP のコアをハックする のマクロは単に値を 設定する だけです。関数から値を返すことはありません。

関数から値を返すための、定義済みマクロ

マクロ	説明
RETURN_RESOURCE(resource)	リソースを返します。
RETURN_BOOL(bool)	Boolean を返します。
RETURN_NULL()	何も返しません (NULL 値を返します)。
RETURN_LONG(long)	long を返します。
RETURN_DOUBLE(double)	double を返します。
RETURN_STRING(string, duplicate)	文字列を返します。duplicate フラグでは、 **estrdup()** を使用して文字列を複製するかどうかを指定します。
RETURN_STRINGL(string, length, duplicate)	指定した長さの文字列を返します。それ以外の振る舞いは *RETURN_STRING* と同じです。 しかし、このマクロは高速でバイナリセーフです。
RETURN_EMPTY_STRING()	空の文字列を返します。
RETURN_FALSE	false を返します。
RETURN_TRUE	true を返します。

関数の返り値を設定するための、定義済みマクロ

マクロ	説明
RETVAL_RESOURCE(resource)	指定したリソースを返り値に設定します。
RETVAL_BOOL(bool)	指定した Boolean 値を返り値に設定します。
RETVAL_NULL	NULL を返り値に設定します。
RETVAL_LONG(long)	指定した long 値に返り値を設定します。
RETVAL_DOUBLE(double)	指定した double 値に返り値を設定します。
RETVAL_STRING(string, duplicate)	指定した文字列に返り値を設定します。必要に応じて Zend 内部メモリに文字列をコピーします (*RETURN_STRING* も参照ください)。
RETVAL_STRINGL(string, length, duplicate)	指定した文字列を返り値に指定し、その長さが length となるようにします (*RETVAL_STRING* も参照ください)。 このマクロは高速でバイナリセーフです。 文字列の長さがわかっている場合は常にこのマクロを使用すべきです。
RETVAL_EMPTY_STRING	空の文字列を返り値に指定します。
RETVAL_FALSE	false を返り値に指定します。
RETVAL_TRUE	true を返り値に指定します。

配列やオブジェクトのような複雑な型を返すには **array_init()** および **object_init()** を使用し、対応するハッシュ関数を return_value に使用します。 これらの型は簡単に作成することはできないので、定義済みのマクロは存在しません。

情報の表示

スクリプト内で print() を使用したときのように、 あなたのモジュールから出力ストリームにメッセージを表示しなければならない場面は よくあることでしょう。PHP では、警告メッセージの出力や phpinfo() 用の出力の生成などの一般的なタスクのための関数を 用意しています。以下の節で、その詳細を説明します。 これらの関数の使用例については CD-ROM を参照ください。

zend_printf()

zend_printf() は、標準の printf() と同じような動作をしますが、 出力先が Zend の出力ストリームとなります。

zend_error()

zend_error() は、エラーメッセージを生成するために使用します。 この関数は 2 つの引数を受け取ります。最初の引数はエラーの型 (zend_errors.h を参照ください)、 そして 2 番目の引数がエラーメッセージとなります。

```
zend_error(E_WARNING, "This function has been called with empty arguments");
```

指定できる値を Zend API: PHP のコアをハックする にまとめます (以下 を参照ください)。 これらの値は、php.ini でも参照されます。 選んだ型によっては、エラーメッセージがログに記録されます。

Zend の定義済みエラーメッセージ

エラー	説明
E_ERROR	エラーを発生させ、その場でスクリプトの実行を停止します。
E_WARNING	一般的な警告を発生させ、そのまま続行します。
E_PARSE	パーサのエラーを発生させ、そのまま続行します。
E_NOTICE	通知を発生させ、そのまま続行します。*php.ini* のデフォルト設定では、この型のエラーメッセージは表示されないようになっていることに注意しましょう。
E_CORE_ERROR	コアによる内部エラーです。ユーザが作成したモジュールでは

ブラウザへの警告メッセージの表示

phpinfo() の出力内容

実際にモジュールを作成してみると、そのモジュールについての情報を phpinfo() に表示してみたくなるでしょう（モジュール名だけは、モジュール一覧のところにデフォルトで表示されます）。 PHP allows you to create your own section in the phpinfo() output with the ZEND_MINFO() function. この関数はモジュール記述子ブロック（先ほど説明しました）に配置しなければなりません。スクリプトが phpinfo() をコールした際には、常にこの関数がコールされます。

ZEND_MINFO 関数を指定すると、PHP は phpinfo() の出力に自動的にセクションを追加します。 ここには見出しとしてモジュール名が含められます。 それ以外のすべてはあなた自身が書式設定して表示する必要があります。

一般的には、まず **php_info_print_table_start()** を使用して HTML テーブルのヘッダを作成した後で、 標準関数 **php_info_print_table_header()** および **php_info_print_table_row()** を使用します。 これらの関数は、どちらも引数としてカラム数（整数）とカラムの内容（文字列）をとります。 Zend API: PHP のコアをハックする に、ソースの例とその出力を示します。 テーブルのフッタを表示するには、**php_info_print_table_end()** を使用します。

Example#13 ソースコードおよび phpinfo() の出力のスクリーンショット

```
php_info_print_table_start();
php_info_print_table_header(2, "First column", "Second column");
php_info_print_table_row(2, "Entry in first row", "Another entry");
php_info_print_table_row(2, "Just to fill", "another row here");
php_info_print_table_end();
```

実行時の情報

現在実行中のファイル名など、実行時の情報を表示することも可能です。 現在実行中の関数名を取得するには、関数 **get_active_function_name()** を使用します。 この関数は何も引数を受け取らず、関数名へのポインタを返します。 現在実行中のファイル名を取得するには、関数 **zend_get_executed_filename()** を使用します。 この関数は executor グローバルにアクセスします。これは TSRMLS_C マクロを使用して渡されます。 executor グローバルは、Zend から直接コールされたすべての関数（この章で先ほど説明した INTERNAL_FUNCTION_PARAMETERS の一部です）で自動的に使用可能となります。 自動的に使用可能にならないような関数で executor グローバルにアクセスしたい場合は、 その関数内でいちど TSRMLS_FETCH() マクロをコールします。 これにより、グローバルの内容がローカルスコープに読み込まれます。

最後に、現在実行中の行番号を取得するには、関数 **zend_get_executed_lineno()** を使用します。 この関数も、引数として executor グローバルを受け取ります。 これらの関数の使用例については Zend API: PHP のコアをハックする を参照ください。

Example#14 実行時の情報の表示

```
zend_printf("The name of the current function is %s&lt;br&gt;", get_active_function_name(TSRMLS_C));
zend_printf("The file currently executed is %s&lt;br&gt;", zend_get_executed_filename(TSRMLS_C));
zend_printf("The current line being executed is %i&lt;br&gt;", zend_get_executed_lineno(TSRMLS_C));
```

スタートアップ関数およびシャットダウン関数

スタートアップ関数およびシャットダウン関数は、モジュールの 一回限りの初期化/後処理のために使用します。 この章で以前に説明したように (Zend モジュール記述子の ブロックを参照ください)、モジュール単位およびリクエスト単位の スタートアップ/シャットダウンイベントが存在します。

モジュールのスタートアップ関数およびシャットダウン関数は、 モジュールが読み込まれ、初期化が必要な場合にコールされます。 リクエストのスタートアップ関数およびシャットダウン関数は、 リクエストが処理されるたび（つまり、ファイルが実行されるたび）に毎回コールされます。

動的拡張モジュールでは、モジュールおよびリクエストの スタートアップ/シャットダウンイベントが同時に発生します。

これらの関数の宣言および実装は、マクロを使用して行います。 詳細は、さきほどの節 "Zend モジュールブロックの宣言" を参照ください。

ユーザ関数のコール

モジュール内からユーザ関数をコールすることができます。 これは、コールバックを実装する場合などにとても便利です。 例えば配列の順次処理や検索、 あるいはイベントベースのプログラムに使用できます。

ユーザ関数をコールするには、**call_user_function_ex()** を使用します。この関数に渡す内容は、アクセスしたい関数テーブルについてのハッシュ、 オブジェクトへのポインタ（メソッドをコールする場合）、関数名、 返り値、引数の数、引数の配列、そして zval を分割するかどうかを示すフラグです。

```
ZEND_API int call_user_function_ex(HashTable *function_table, zval *object,
zval *function_name, zval **retval_ptr_ptr,
int param_count, zval **params[],
int no_separation);
```

function_table および object は、両方とも指定する必要はなく、どちらか一方だけでよいことに注意しましょう。 メソッドをコールしたい場合は、そのメソッドを含むオブジェクトを 指定しなければなりません。このときに、 **call_user_function()** は自動的にこのオブジェクトの関数テーブルを設定します。 メソッドをコールするのではない場合は、 function_table だけを設定して object は NULL とします。

通常は、デフォルトの関数テーブルは "root" であり、 このテーブルにすべての関数のエントリが含まれます。 この関数テーブルはコンパイラ全体から CG マクロを使用してアクセス可能です。 コンパイラグローバルの内容を関数内で使用するには、 TSRMLS_FETCH マクロを一度コールします。

関数名は zval コンテナで指定します。 最初は少し戸惑うかもしれませんが、これはきわめて論理的なことです。 なぜなら、スクリプト内からコールされる関数でパラメータとして受け取った 関数名は、たいていの場合は再び zval コンテナに保存されることになるからです。 そのため、この関数に引数を渡すだけでよくなります。ここで使用する zval は IS_STRING 型でなければなりません。

その次の引数には、返り値へのポインタを含めます。 このコンテナ用のメモリを確保する必要はありません。 関数が自分自身でメモリを確保します。しかし、 その後には（**zval_dtor()** を使用して）コンテナを破棄しなければなりません!

その次はパラメータの数を表す整数値、 それからすべての必要なパラメータを含む配列となります。 最後の引数では、この関数が zval の分割を行うかどうかを指定します。 この引数は常に 0 にしておくべきです。 1 にすると関数のメモリ消費量を抑えられますが、 分割が必要なパラメータがひとつでもあった場合に関数が実行できなくなります。

Zend API: PHP のコアをハックする は、ユーザ関数をコールする簡単な例です。 このコードは、引数として渡された関数をコールし、 コールした関数の返り値をそのまま自分自身の返り値として使用します。 最後のほうのコンストラクタおよびデストラクタの使用に注目しましょう。 ここではこれらの使用は必須ではありませんが（値を分割しており、代入は安全に行われる）、念のために記述しています。

Example#15 ユーザ関数のコール

```
zval  **function_name;
zval  *retval;

if((ZEND_NUM_ARGS()  !=  1)  ||  (zend_get_parameters_ex(1,  &function_name)  !=  SUCCESS))
{
      WRONG_PARAM_COUNT;
}

if((*function_name)->type  !=  IS_STRING)
{
       zend_error(E_ERROR,  "Function  requires  string  argument");
}

TSRMSLS_FETCH();

if(call_user_function_ex(CG(function_table),  NULL,  *function_name,  &retval,  0,  NULL,  0)  !=  SUCCESS)
{
      zend_error(E_ERROR,  "Function  call  failed");
}

zend_printf("We  have  %i  as  type¥n",  retval->type);

*return_value  =  *retval;
zval_copy_ctor(return_value);
zval_ptr_dtor(&retval);

<?php

dl("call_userland.so");

function  test_function()
{
     echo "We are in the test function!¥n";
     return 'hello';
}

$return_value  =  call_userland("test_function");

echo  "Return  value:  '$return_value'";
?>
```

上の例の出力は以下となります。

```
We  are  in  the  test  function!
We  have  3  as  type
Return  value:  'hello'
```

ini ファイルのサポート

PHP 4 では ini ファイルのサポートが設計しなおされています。 デフォルトの初期エントリをコード中で直接指定したり、 それらの値を実行時に読み込んで変更したり、 変更があったことを通知するためのメッセージハンドラを作成したりすることができます。

あなたのモジュール用の .ini セクションを作成するには、 PHP_INI_BEGIN() マクロでセクションの開始位置を指定、 PHP_INI_END() マクロでセクションを終了位置を指定して、 その中に PHP_INI_ENTRY() を使用してエントリを作成します。

```
PHP_INI_BEGIN()
PHP_INI_ENTRY("first_ini_entry",    "has_string_value",  PHP_INI_ALL,  NULL)
PHP_INI_ENTRY("second_ini_entry",  "2",                    PHP_INI_SYSTEM,  OnChangeSecond)
PHP_INI_ENTRY("third_ini_entry",   "xyz",                 PHP_INI_USER,  NULL)
PHP_INI_END()
```

PHP_INI_ENTRY() マクロは 4 つのパラメータを受け取ります。 パラメータの内容は、それぞれエントリ名、エントリの値、変更の権限、 そして変更通知ハンドラへのポインタとなります。 エントリ名とエントリの値は、たとえ実際の値が整数であったとしても 文字列で指定しなければなりません。

権限は、3 種類に分けられます。 PHP_INI_SYSTEM は php.ini ファイルでしか変更できません。PHP_INI_USER は、実行時にユーザが設定を上書きすることができます。上書きするには .htaccess のような別の設定ファイルを使用します。 そして PHP_INI_ALL の場合は無制限に値を変更することができます。 4 種類目として PHP_INI_PERDIR というものもありますが、 この振る舞いについては検証することができません。 yet.

4 番目のパラメータには、変更通知ハンドラへのポインタを指定します。 これらの ini エントリが変更された際に、ハンドラがコールされます。 ハンドラを宣言するには、PHP_INI_MH マクロを使用します。

```
PHP_INI_MH(OnChangeSecond);                  // ini エントリ "second_ini_entry" 用のハンドラ

// ここで ini エントリを指定します

PHP_INI_MH(OnChangeSecond)
{

       zend_printf("Message  caught,  our  ini  entry  has  been  changed  to  %s&lt;br&gt;",  new_value);

       return(SUCCESS);

}
```

変更ハンドラの変数 new_value に、 新しい値がに文字列として渡されます。 PHP_INI_MH の定義を見ると、 それ以外にもいくつかのパラメータがあることがわかるでしょう。

```
#define  PHP_INI_MH(name)  int  name(php_ini_entry  *entry,  char  *new_value,
                                    uint new_value_length, void *mh_arg1,
                                   void *mh_arg2, void *mh_arg3)
```

これらの定義は php_ini.h で確認できます。 メッセージハンドラの内部では、エントリ全体を含む構造体・ 新しい値・その長さ・そして 3 つのオプションの引数にアクセスすることができます。 3 つのオプションの引数を指定するには、 PHP_INI_ENTRY1 (追加の引数を 1 つ指定する)、

PHP_INI_ENTRY2 (追加の引数を 2 つ指定する) そして PHP_INI_ENTRY3 (追加の引数を 3 つ指定する) のそれぞれのマクロを使用します。

変更通知ハンドラを使用すべき場面としては以下のようなものがあります。 高速にアクセスするために ini エントリをローカルにキャッシュしたり、 値が変更された際に何らかのタスクを実行したりといった場合です。 例えば、特定のホストと常に接続を確率している拡張モジュールにおいてホスト名が変更された場合に、今の接続を自動的に終了させて 新しい接続を確率するなどといったものが考えられます。

Zend API: PHP のコアをハックする に示すマクロを使用することによっても、 ini エントリへのアクセスは可能です。

PHP の ini エントリにアクセスするためのマクロ

マクロ	説明
INI_INT(name)	エントリ *name* の現在の値を integer (long) で返します。
INI_FLT(name)	エントリ *name* の現在の値を float (double) で返します。
INI_STR(name)	エントリ *name* の現在の値を文字列で返します。 *注意:* 文字列は複製されず、 内部データへのポインタが返されます。後でこのデータにアクセスするには、ローカルメモリに複製しなければなりません。
INI_BOOL(name)	エントリ *name* の現在の値を Boolean (zend_bool で定義されるもので、その実体は、現在は unsigned char) で返します。
INI_ORIG_INT(name)	エントリ *name* の元の値を integer (long) で返します。
INI_ORIG_FLT(name)	エントリ *name* の元の値を float (double) で返します。
INI_ORIG_STR(name)	エントリ *name* の元の値を文字

最後に、あなたが作成した ini エントリについて PHP に教えてあげなければなりません。 モジュールのスタートアップ関数およびシャットダウン関数の中で、マクロ REGISTER_INI_ENTRIES() および UNREGISTER_INI_ENTRIES() を使用します。

```
ZEND_MINIT_FUNCTION(mymodule)
{

      REGISTER_INI_ENTRIES();

}

ZEND_MSHUTDOWN_FUNCTION(mymodule)
{

      UNREGISTER_INI_ENTRIES();

}
```

次に進むべき道

ここでは PHP について多くのことを学びました。あなたは、今や 動的ロード可能なモジュールや静的にリンクされた拡張モジュールの 作成方法を身につけています。PHP や Zend が変数を内部でどのように 管理しているのか、そして変数を作成したりそれにアクセスするには どうすればいいのかもわかるようになりました。 各種情報の表示・シンボルテーブルへの変数の自動登録などの お決まりの手続きを行う関数についても知っています。

この章の中には「リファレンス的な」部分も多々ありましたが、 あなたが独自の拡張モジュールを書き始めるための手助けとなることを 望んでいます。 ところで、スペースの都合上 省略せざるを得なかった内容が多くあります。 ぜひ、時間をとってヘッダファイルや既存のモジュールについて 勉強してみることをお勧めします（特に ext/standard ディレクトリのモジュールや MySQL モジュールがお勧めです。これらのモジュールには 一般的な機能がすべて実装されています）。 これらを勉強することで、（ここで取り上げられなかった機能も含めた）API 関数の使用方法を身につけることができるでしょう。

リファレンス: 設定マクロ

config.m4

config.m4 は buildconf によって使用されます。 configure 時に実行する内容についてはすべてここで指示しておかなければなりません。 例えば、行うべきテストや必要な外部ファイル（ヘッダファイルなど）についての指示などがあてはまります。このために使用できるマクロが PHP で定義されています。その中でも最も有用なものを Zend API: PHP のコアをハックする にまとめます。

config.m4 で使用する M4 マクロ

マクロ	説明
AC_MSG_CHECKING(message)	*configure* 時に "checking <message>" というテキストを表示します。
AC_MSG_RESULT(value)	*AC_MSG_CHECKING* の結果を指定します。 value には、 *yes* あるいは *no* のいずれかを指定しなければなりません。
AC_MSG_ERROR(message)	*configure* 時に エラーメッセージ message を表示し、 スクリプトを終了させます。
AC_DEFINE(name,value,description)	*php_config.h* に *#define* を追加し、値を value に設定して description をコメントに加えます (モジュールで条件付きコンパイルを行う場合に有用です)。
AC_ADD_INCLUDE(path)	コンパイラのインクルードパスを追加します。 ヘッダファイルを探すためのサーチパスを追加する場合などに使用します。
AC_ADD_LIBRARY_WITH_PATH(libraryname,librarypath)	リンクする外部ライブラリを指定します。
AC_ARG_WITH(modulename,description,unconditionaltest,conditionaltest)	非常に強力なマクロであり、 *configure --help* の出力内容に description という説明とともにモジュールを追加します。 *configure* スクリプトにオプション *--with-<modulename>* が指定されているかどうかを PHP がチェックし、もし指定されていれば スクリプト *unconditionaltest* を実行します (例えば *--with-myext=yes*)。 このとき、オプションの値は変数 $withval に格納されます。オプションが指定されなかった場合は、 *conditionaltest* を実行します。
PHP_EXTENSION(modulename, [shared])	このマクロは、PHP があなたの拡張モジュールを設定するために 必ずコールしなければならないものです。 モジュール名に加えて 2 番目の引数を指定することができ、 ここでは共有モジュールとしてコンパイルを行うかどうかを指示します。 この内容は、ソースをコンパイルする際の *COMPILE_DL_<modulename>* の定義として使用されます。

API マクロ

Zend API では、zval コンテナへのアクセスを単純化する マクロ群が定義されています (Zend API: PHP のコアをハックする を参照ください)。

zval コンテナへのための API マクロ

マクロ	参照先
Z_LVAL(zval)	(zval).value.lval
Z_DVAL(zval)	(zval).value.dval
Z_STRVAL(zval)	(zval).value.str.val

Z_STRLEN(zval)	(zval).value.str.len
Z_ARRVAL(zval)	(zval).value.ht
Z_LVAL_P(zval)	(*zval).value.lval
Z_DVAL_P(zval)	(*zval).value.dval
Z_STRVAL_P(zval_p)	(*zval).value.str.val
Z_STRLEN_P(zval_p)	(*zval).value.str.len
Z_ARRVAL_P(zval_p)	(*zval).value.ht
Z_LVAL_PP(zval_pp)	(**zval).value.lval
Z_DVAL_PP(zval_pp)	(**zval).value.dval
Z_STRVAL_PP(zval_pp)	(**zval).value.str.val
Z_STRLEN_PP(zval_pp)	(**zval).value.str.len
Z_ARRVAL_PP(zval_pp)	(**zval).value.ht

TSRM API

将来の展望: PHP 6 および Zend Engine 3

FAQ: よくある質問

目次

一般的な情報

このセクションでは PHP に関するもっとも基本的な情報 (PHP とは何か? PHP は何をするものか? 等)を扱います。

1. PHPã¨ã¯?
2. PHP ã¨ã¯ä½ã®ç¥ã§ãã?
3. åãã¾ã¨ã§ã³ãã¤ãã®é¢ä¿ã¯?
4. ç°ãªãã¾ã¨ã§ã³ãã® PHP ãåæã«å®è¡ãã¾ã¨ãã¾ãã?
5. PHP 3 ã¨ PHP 4 ã®éãã¯ä½ã§ãã?
6. ãã°ãè¦ã¤ãã¾ãã! è¤ãã«ä½ãã°ãã§ãã?

PHPとは?

マニュアルの序文より:

PHP は "PHP: Hypertext Preprocessor" を意味する、HTML 埋め込み型の スクリプト言語です。 PHP の多くの構文は C、Java、Perl 言語から転 用したものですが、PHP 特有の機能を実装するために独自の構文を導入 しています。PHP の目的は、Web 開発者が動的に生成されるページの 作成を速やかに行うことができるようにすることです。

PHP とは何の略ですか?

PHP は PHP: Hypertext Preprocessor の略です。 最初の文字が「頭字語の頭字語」になっているため多くの人は混乱します。 この種の頭字語は「再帰的頭字語」と呼ばれます。興味がある方は、» Free On-Line Dictionary of Computing で再帰的頭字語についてもっと詳しく

知ることが出来ます。

各バージョン間の関係は?

PHP/FI 2.0 は PHP の初期の、 そしてもうすでにサポートされていないバージョンです。 PHP 3 は PHP/FI 2.0 の後継バージョンで、PHP/FI 2.0 に比べて大きく進歩しています。 PHP 4 は PHP の最新バージョンで、内部に » Zend Engine を使用しています。 PHP 5 は Zend engine 2 を使用し、 この他の機能と共に、多くのOOP 機能が追加されています。

異なるバージョンの PHP を同時に実行することができますか?

はい。PHP 4 のソースコードディストリビューションに含まれる INSTALL ファイルを見てください。また、 付録 の関連個所にも目を通してください。

PHP 3 と PHP 4 の違いは何ですか?

ここには、非常に重要な新機能の一覧を挙げておきます。

- 拡張 API モジュール
- UNIX でのビルドの一般化
- マルチスレッドサーバにも対応した一般化されたウェブサー バインターフェース
- 進化したシンタックスハイライター
- ネイティブ HTTP セッションのサポート
- 出力のバッファリングサポート
- 強力な configuration システム
- 参照カウンタのサポート

上記機能の詳しい説明については» What's New in PHP 4 overview を参 照してください。もし PHP 3 から PHP 4 へ移行する場合には 付録 の関連箇所にも目を通してください。

バグを見つけました! 誰に伝えればよいですか?

まず、PHP Bug Database でそれが既知のバグかどうかを調べてください。 もしデータベースで見つけることができなければ、「reporting form」 を使ってバグを報告してください。メーリングリスト等ではなく Bug Database に報告することは非常に重要です。 なぜなら個々のバグには「Tracking Number」がつけられるため、 後からバグの状態をチェックすることが可能だからです。 Bug Database は » http://bugs.php.net/ にあります。

メーリングリスト

このセクションではPHPのコミュニティへの参加に関する質問を扱います。 最良の方法はメーリングリストに参加することです。

1. PHPã®ãƒãƒ¼ã‚¸ãƒ§ãƒ³ã¯ã„ãã¤ãŒãƒªãƒªãƒ¼ã‚¹ã•ã‚Œã¦ã„ã¾ã™ã‹?
2. ä»¥ä¸‹ã®ãƒ¡ãƒ¼ãƒ«ã‚¢ãƒ‰ãƒ¬ã‚¹ã¯ä½¿ç”¨ã§ãã¾ã™ã‹?
3. Help! ãƒ¡ãƒ¼ãƒªãƒ³ã‚°ãƒªã‚¹ãƒˆã«å‚ åŠ /è±é€ã§ãã¾ã›ã‚“!
4. ã©ãŠã„ãƒãƒ¼ã‚¸ãƒ§ãƒ³ãƒ®ã¢ã¼ãƒ«(éå»ãƒãƒ¼)ã¯ä¿ãŠã¾ã™ã‹?
5. ãƒãƒ¼ã‚¸ãƒ§ãƒ³ãƒªãƒ¼ã‚¹ã¯ã©ã®ãã‚‰ã„ã®é »åº¦ã§è¡Œã‚ã‚Œã¾ã™ã‹?
6. ãƒãƒ¼ã‚¸ãƒ§ãƒ³ãƒªãƒ¼ã‚¹æ¸¿ã¿ãƒãƒ¼ã‚¸ãƒ§ãƒ³ã¯ã©ã®æ[ã]å ±ãå«ãŠãªã§ãã¾ã™ã‹?

PHPのメーリングリストはありますか?

もちろんです! 様々なトピックに関するたくさんのメーリングリストが あります。メーリングリストの一覧が私たちの » サポートページ にあります。

最も一般的なメーリングリストが php-general です。 参加するには » php-general-subscribe@lists.php.net にメールを送ってください。 Subject や本文には何も記述する必要はありません。 脱退するには » php-general-unsubscribe@lists.php.net にメールを送信してください。

また、私たちの » サポートページ を通じて参加/脱退を行うこともできます。

他にコミュニティはありますか?

世界中に数え切れないほどあります。» サポートページ には例としていくつかの IRC サーバや他の国のメーリングリストへのリンクがあります。

Help! メーリングリストに参加/脱退できません!

もし、メーリングリストへの参加/脱退で問題が起きているなら、 それはおそらくシステムがあなたのメールアドレスを 正しく取得できていないからだと思われます。 仮にあなたのメールアドレスを joeblow@example.com とすると、参加のリクエストは php-general-subscribe-joeblow=example.com@lists.php.net、 脱退のリクエストは php-general-unsubscribe-joeblow=example.com@lists.php.net へメールを送信することで行うことができます。 他のメーリングリストに関しても同様です。

どこかにメーリングリストのアーカイブ(過去ログ)はありますか?

はい。» サポートページ にアーカイブサイトの一覧があります。 また、メーリングリストの記事は news メッセージとしても記録されています。 news クライアントで » news://news.php.net/ へアクセスすることができます。さらに、 » http://news.php.net/ に実験的な news サーバへの Web インターフェースがあります。

メーリングリストではどんな質問をすることができるのですか?

PHP が日進月歩で有名になっている影響で、php-general メーリングリストの流量も増加しています (今のところ、1日あたり150〜200通の流量があります)。 なので、参加者全員の利益のために、メーリングリストは 他のどこを当たっても駄目だった場合の最後の手段にしてください。

メーリングリストに投稿する前に、 この FAQ とマニュアルを見て手がかりを探してください。 もし駄目なら次にメーリングリストのアーカイブをチェックしてください (上記参照)。もし、PHP のインストール、 もしくは設定で問題が起こっている場合は (ソースアーカイブに含まれる)すべてのドキュメントと README ファイルに目を通してください。 それでも問題が解決できない場合に、 メーリングリストを使用することは大歓迎です。

メーリングリストに投稿するときには、どんな情報を含めればよいのですか?

「PHP が動作しません。助けて! 何がおかしいのですか?」 といった投稿は間違いなく誰にとっても無意味なものです。 もしPHPが動作しないのであれば、 あなたが使用しているオペレーティングシステム、PHPのバージョン、 またそれをどのように手に入れたか(コンパイル済みだった、 CVS から手に入れた、RPM で提供されたものだった、等) 動作させようとしてあなたはどんなことをしたのか、 どこであなたは行き詰まって、どんなエラーメッセージが出るのか、 といった情報を記述する必要があります。

これは他のどんな問題にも当てはまることです。あなたは、何をしたのか、 どこで行き詰まっているのか、何をしようとしているのか、 もしできるなら正確なエラーメッセージを記述してください。 もしあなたが書いた PHP のプログラムで問題が起こった場合は、 動作しない個所のソースコードを含める必要があります。 でも、必要以上のソースコードは含めないでください! さもないと、 あなたの投稿は非常に読みにくいものとなり、 多くのひとはこうした投稿を読み飛ばしてしまいます。 もし、どれくらいの情報を含めればいいのか分からない場合には、 できるだけの情報を記述してください(多いに越したことはありません)。

もう一つ重要な点があります。 それは Subject であなたの問題を端的に表すことです。 「助けてーー!」とか「何かおかしいのでしょうか?」といった Subject のついたメールはほとんど人に無視されてしまいます。

そして最後に、 » How To Ask Questions The Smart Way にあるペーパーを読むことをお薦めします。 このペーパーはあらゆる人、 特にあなた自身にとって多いに参考になるはずです。

PHP を手に入れるには

このセクションでは、PHP をダウンロードする詳しい場所や OS に関する話題を扱います。

PHPはどこで手に入れることができますか?

PHPネットワークのどのメンバーのサイトからでもダウンロードすることができ ます。これらは» http://www.php.net/ から見つけることができます。また匿名 CVSを使って最新のソースコードを手に入れることもできます。詳しい情報は、 » http://www.php.net/anoncvs.php を参照してください。

コンパイル済みのバージョンはありますか?

私たちはWindows版のみコンパイル済みのバイナリを配布しています。というの もPHPをすべてのLinux/Unixのプラットホーム用にすべての拡張モジュールの組 み合わせでコンパイルするのは不可能だからです。ただ、最近の多くのLinuxディ ストリビューションには予めPHPがインストールされています。Windows用のバイ ナリは» ダウンロードページ からダウンロードすることができます。Linux用のバイナリについてはあなたが 使用しているディストリビューションのウェブサイトを訪れてください。

一部のPHP拡張モジュールのコンパイルに必要なライブラリはどこで手に入りま すか?

> **注意**: *印がついているものはスレッドセーフなライブラリではないため、Windows用 のマルチスレッドなウェブサーバ(IIS, Netscape)のサーバモジュールとして コンパイルされたPHPで使用してはいけません。現在のところUnix環境では関 係ありません。

- » LDAP (Unix).
- » LDAP (Unix/Win) : Mozilla Directory (LDAP) SDK
- » free LDAP server.
- » Berkeley DB2 (Unix/Win) : http://www.sleepycat.com/.
- » SNMP* (Unix): .
- » GD* (Unix/Win).
- » mSQL* (Unix).
- » PostgreSQL (Unix).
- » IMAP* (Win/Unix).
- » Sybase-CT* (Linux, libc5) : Available locally.
- » FreeType (libttf):.
- » ZLib (Unix/Win32).
- » expat XML parser (Unix/Win32).
- » PDFLib.
- » mcrypt.
- » mhash.
- » t1lib.
- » dmalloc.
- » aspell.
- » readline.

上記のライブラリはどのように使うのですか?

個々のライブラリで提供される情報に従ってください。いくつかのライブラリ (GD等)は、PHPのconfigureスクリプト実行時に自動的に検出されます。その他の ライブラリに関してはconfigureスクリプトのオプションに '--with-EXTENSION'オプションを追加すること使用できます。 'configure --help'を実行するとこれらのオプション一覧が 表示されます。

CVSリポジトリから最新のPHPソースコードを手に入れました。Windows上でコン パイルするにはどうしたらよいですか?

まず、Microsoft Visual C++ v6が必要です(v5でも問題ないかもしれませんが、 v6を使ってください)。加えて、いくつかのファイルが必要です。マニュアルの ソースからの構築を参照してください。

ブラウザキャパビリティファイルはどこで手に入りますか？

browscap.iniファイルは » http://browsers.garykeith.com/downloads.aspで 手に入れることができます。

データベースに関する問題

このセクションでは PHP とデータベースとの関係に関する一般的な質問を 扱います。なんと！ PHP は事実上あらゆるデータベースにアクセスすることが できます。

1. PHP ã¯ Microsoft SQL Server ã«ã¢ã¯ã»ã¹ãã¨èãã¾ãããã©ãã° ãã®ãã§ãã¾ãï¼
2. Microsoft Access ãã¾ã¿ãã¾ã¹ã«ã¢ã¯ã»ã¹ãã¾ãï¼
3. PHP 4 ã«ã¢ãã°ã¬ã¼ãããã¨mysql ãµã¼ãã "Warning: MySQL: Unable to save result set in ..." ã¨ãè¦åãã¤ã¥ãã¾ãã¤ãè¨·ãã£ã¦ãã®ã§ãã?
4. PHP 5 ãçã¯ MySQL ã¯ã©ã¢ã¤ã¢ã³ãã©ã¤ãã©ãªãã¯ãã¯ã¯ãã¯ã¯ c§ã«ã¨ã£ã¦ã©ãã®æå³ãã¾ãã¾ãã? PHP ã MySQL ã¨ãã«ãä½¿ç¨ãã¾ãã¾ãã®ãçã? MySQL ãä½¿ç¨ãã¾ãã¨ãã¾ã "function undefined" ã¨ã©ã¼ãåºãã®ãçãã¾ãã©ãã®ãã§ãã?
5. å[ﾊ]±æ MySQL ãµãã¼ãã¤ã¹ã«ãã¾ãlibphp4 ãã¾ã¾ãã¾ã¨ åæ« Apache ãã³ãã³ãã¾ãã¾ãç´ãã¾ãï¼
6. "Warning: 0 is not a MySQL result index in <file> on line <x>" ãã¾ã¯ "Warning: Supplied argument is not a valid MySQL result resource in <file> on line <x>" ã®ãã¾ãªã¨ã©ã¼ãåºã®ã¯ãªãçãã§ãã?

PHP は Microsoft SQL Server にアクセスできると聞きました。どうすれば よいのでしょうか？

Windows マシン上では、ODBC サポートと適切な ODBC ドライバを 使用すればよいだけです。

Unix マシン上では、Sybase-CT ドライバを使って Microsoft SQL Server にアクセスすることができます。なぜなら(ほとんど完全に)プロトコル 互換だからです。Sybase は » 必要な Linux 用ライブラリのフリーな実装 を作成しました。他の Unix システムでは適切なライブラリを手に入れるために Sybase と連絡を取る必要があります。 次の質問に対する回答も参照してください。

Microsoft Access データベースにアクセスできますか？

はい。もし全て (PHP と Microsoft Access) を Windows9x/Me/NT/2000 上で動作させるのであり、ODBC と Microsoft が提供する Microsoft Access 用 ODBC ドライバ が使用できる状態ならば、 すべての必要なツールは揃っています。

Unix で PHP を動作させて Windows マシンで動作する MS Access に接続したい場合には Unix ODBC ドライバが必要です。» OpenLink Software が Unix ベースの ODBC ドライバを提供しています。

他には、Microsoft SQL Server に ODBC ドライバを使用してデータを 保存するという手段もあります。これによって、Microsoft Access (ODBC を使用します) と PHP (組み込まれているドライバを使用します) でデータに アクセスすることができます。また、Microsoft Access と PHP 両者共に 解釈可能な中間ファイル(例えば単層からなるファイルや、dBase データベース等)を使用する手もあります。これに関しては OpenLink Software の Tim Hayes が以下のように述べています。

> PHP からじかに ODBC ドライバが使用できる場合(つまり OpenLink の ドライバを使用している場合)には、他のデータベースを中継手段として 用いることはよいアイデアとは言えない。どうしても中間ファイルが 必要な場合のために、OpenLink は NT、Linux、そして他の Unix 用の Virtuoso(仮想データベースエンジン)をリリースした。私たちの » ウェブサイトを 訪れてもらえれば、無料でダウンロードできる。

うまくいくことが分かってるもう一つの方法は、MySQL と Windows 用の MyODBC ドライバを利用してデータベースを同期する方法です。Steve Lawrence が言うには、

- MySQL を説明に従って任意のプラットフォームにインストールします。 最新バージョンは » http://www.mysql.com/ にあります。データベースを作成するとき、そしてユーザアカウントを を設定するときに、ホストフィールドに % か MySQL を使って アクセスする Windows マシンのホスト名を入力する、ということ以外には 特別な設定は必要ありません。 使用するサーバ名、ユーザ名、パスワードを書きとめておいてください。
- MySQL サイトから Windows 用 MyODBC ドライバをダウンロードします。 それを Windows マシンにインストールします。 アーカイブに含まれるユーティリティプログラムで、 テストを行うことができます。
- コントロールパネルの ODBC administrator を使用して、ユーザ もしくはシステム DSN を作成します。DSN 名を決定して、Step1 で MySQL に設定したホスト名、ユーザ名、パスワード、ポート等を入力します。
- Access をフルインストールでインストールします。これは適切な アドインをインストールするためです。少なくとも ODBC サポートとリンクテーブルマネージャが必要です。
- ここからがお楽しみです。新規データベースを作成しましょう。 テーブルウィンドウで右クリックしてリンクテーブルを選択します。 もしくはファイルメニューから外部データの取り込みを選び リンクテーブルを選択します。ファイルブラウザが表示されたら ファイルタイプから ODBC を選択します。次にシステム DSN で 選択肢 STEP3 で作成した DSN 名を選択します。リンクするテーブルを 選んで OK ボタンを押しましょう。MySQL サーバでテーブルを オープンできるようになっていて、データの追加/削除/編集ができるように なっています。さらに、クエリの構築、テーブルのインポート/エクスポート、 フォームやレポートの構築等が可能です。

Tips and Tricks(役に立つヒント):

- Access でテーブルを作成してそれを MySQL へエクスポートします。 それを再度アクセスからリンクすれば素早くテーブルを作成すること ができます。
- Access でテーブルを作成するときは、そのテーブルに書き込み権限を もたせるためにプライマリキーを設定しておく必要があります。Access とテーブルをリンクする前に MySQL にプライマリキーを設定 しておくことも忘れないでください。
- MySQL 側でテーブル(の構成)を変更した場合、再度 Access とリンクする 必要があります。ツール>アドイン>リンクテーブルマネージャから 適切な ODBC DSN を選んで再リンクが必要なテーブルを選択します。 また、ここで OK ボタンを押す前に「リンク先を更新するための プロンプトを毎回表示する」をチェックしておけば DSN ソースを 移動させることができます。

PHP 4 にアップグレードしたら、mysql サーバが "Warning: MySQL: Unable to save result set in ..." という警告をしつづけます。何が 起こっているのでしょうか?

一番に考えられるのは、PHP 4 を MySQL へのパス指定なしに --with-mysql オプションをつけてコンパイルしたという場合です。これは PHP が組み込まれた MySQL クライアントライブラリを使用するという ことを意味します。もしあなたのシステムで、PHP 3 や auth-mysql の apache モジュールといった他のバージョンの MySQL クライアントを 使用するアプリケーションが同時に動作している場合には、 各クライアント間でバージョンの競合が発生してしまいます。

PHP 4 を '--with-mysql=/your/path/to/mysql' というようにオプションにパスを記述して再コンパイルすることで、 通常の場合は問題

は解決されます。

PHP 5 では MySQL クライアントライブラリがバンドルされません。これは 私にとってどのような意味がありますか? PHP を MySQL とともに使用する ことはできるのですか? MySQL を使用しようとすると "function undefined" エラーが出るのですが、どうしたらいいのですか?

はい。どのような意味においても PHP は常に MySQL をサポートしています。 PHP 5 で変わったことといえば、クライアントライブラリをバンドルしなく なったということだけです。以下に、順不同で理由を挙げます。

- 最近のシステムにはすでにクライアントライブラリがインストールされています。
- 上と関連して、複数バージョンのライブラリを共存させると環境がおかしく なりがちです。たとえば、mod_auth_mysql と PHP をそれぞれ別のバージョンの ライブラリとリンクさせ、Apache でそれら両方を有効にすると、いとも簡単に クラッシュすることでしょう。また、バンドルされているライブラリが インストールされているサーバとうまく動作するとは限りません。もっとも ありがちな症状は、Unix ドメインソケットのファイル mysql.socket を探す場所の不一致です。
- 保守がとどこおりがちで、現在リリースされているバージョンに どんどん遅れをとってしまいます。
- 将来のバージョンのライブラリは GPL の元で配布されます。そのため、 私たちはアップグレードの手段を提供することができません。なぜなら 私たちのような BSD/Apache スタイルのライセンスを採用している プロジェクトは、GPL のライブラリをバンドルできないからです。 PHP 5 でバンドルをやめたことは最善の方法だと考えます。

このことがそんなに多くの人々に影響することはないでしょう。 Unix ユーザ、少なくとも自分が何をしているかを把握している人たちは、PHP をビルドする際には常に --with-mysql=/usr してシステムの libmyqlclient ライブラリを使用するようにしているでしょう。Windows ユーザは、php.ini で php_mysql.dll を有効にしていることでしょう。 インストール手順の詳細は、MySQL リファレンス を参照ください。また、libmysql.dll が システムの PATH が通った場所にあることを確認してください。その方法に ついては、FAQ の Windows のシステム PATH を設定する を参照ください。 libmysql.dll (やその他多くの PHP 関連ファイル) が PHP フォルダにあることから、このフォルダをシステムの PATH に追加 したくなるかもしれません。

共有 MySQL サポートをインストールしたら、libphp4 がロードされると 同時に Apache がコアダンプします。直りますか?

もしあなたのシステムの MySQL ライブラリが pthreads とリンクされて いる場合にはこの現象が発生します。ldd コマンドを使用して チェック してください。もし pthreads がリンクされている場合は、MySQL の tarball を展開してソースからコンパイルしなおしてください。もしくは SRPM の SPEC ファイルのスレッドクライアントコードの箇所を削除してコンパイル しなおしてください。いずれかの方法で問題を解決できます。その後、 PHP を新しい MySQL ライブラリでコンパイルしなおしてください。

"Warning: 0 is not a MySQL result index in <file> on line <x>" もしくは "Warning: Supplied argument is not a valid MySQL result resource in <file> on line <x>" のようなエラーが出るのはなぜでしょうか?

あなたは値が 0 である結果(result)ID を指定しようとしています。0 は あなたのクエリが何らかの理由で失敗したことを示しています。結果 (result)ID を使用する前に、クエリを送信したあとのエラーをチェック する必要があります。以下のようなコードが正しい方法です。

```
<?php

$result = mysql_query("SELECT * FROM tables_priv");
if (!$result) {
    echo mysql_error();
    exit;
}
?>
```

もしくは

```
<?php

$result = mysql_query("SELECT * FROM tables_priv")
    or die("Bad query: " . mysql_error());
?>
```

インストール

このセクションでは PHP のインストール方法に関する一般的な質問を扱います。 PHP は(OSX 以前の MacOS を除けば)、ほとんど全ての OS とウェブサーバで 利用可能です。

PHP をインストールするには、インストールと設定 の指示に従ってください。

1. ãªãâApache 2 ã®ãã«ãã¹ã¬ãã MPM ã¢ã¾ãâå®éç¨ç°å¢ã§ä¾¿ç¨ãâã¹ãã¯ãªãã®ã§ãâ?
2. Unix/Windows: php.ini ãâ¡ã¤ã«ã¯ãã©ãã«ç½®ãâãã®ãâã®ã§ãâ?
3. Unix: PHP ãâ¤ã³ãã¾ãã«ãã¾ãâãâãâãâ¡ã¤ã«ãâãã¾ãâãâãâ³ã« 'Document Contains No Data(ãâ¾ãã¸è¨ç¤ºã§ãã¾ãâ)' ã¨ãâã¡ãâ»ãâ¾ãã¸è¨ç¤ºãâãä¾¿ãâè¡¨ç¤ºãâ¦ãâ£ã¦ãâã®ã§ãâ?
4. Unix: RPM ãä½¿ã£ã¦ PHP ãâ¤ã³ãã¾ãã«ãâã®ã§ãâãâApache ã PHP ã®ãã¾ãã¸ å¦çãâ¦ãâãã¾ãâãâãä¾¿è¡¨ãâ£ã¦ãâã®ã§ãâ?
5. Unix: PHP 3 ã RPM ãä½¿ã£ã¦ã¤ã³ãã¾ãã«ãâã®ã§ãâãâãâ® RPM ã¯ ç¶åå¿è¦ã¨ãâãâã¾ãã¾ãã¾ãã®ãµãâã¾ãâãã¾ãã³ã³ãâã¤ãã«ãã®ã§ãâãâãâãâ©ãâãâ°ãâã§ãâ?
6. Unix: Apache ã« FrontPage ã¨ã¯ã¨ã¾ã§ã³ã®ãâãâãå¾ã¦ãâãçºç¶ PHP ãåä½ãâãâãã¾ãâ¦ãâãã¾ãâãâãâPHP ã¯ Apache ã® FrontPage ã¨ã¯ã¨ã¾ã³ã¾ã§ã³ã¨å¼±ãâãâãâã¨ã¯ã¨ã§ãâãã¾ãâã®ã§ãâ?
7. Unix/Windows: PHP ãâ¤ã³ãã¾ãã«ãâã®ã§ãâãâã¦ã¶ãâ PHP ã¹ã¯ãâãã«ã¢ã¯ã»ã¹ãâ¦ãä½¿è¨ç¤ºãâãâã
8. Unix/Windows: PHP ãâ¤ã³ãã¾ãã«ãâã®ã§ãâãâã¦ã¶ãâ PHP ã¹ã¯ãâãã«ã¢ã¯ã»ã¹ãâ¦ãâã¨ Internal Server Error 500 ã¨ãâã¡ãâ¦ãâã¾ãâã
9. PHP ãâ¤ã³ãã¾ãã«ãâã®ã§ãâã¯ã©ã¾ãâãâãâãâé¡ãâãé²ãâã ã®ã§ãâãâ apache ãè¡¨ç¤ºãâãâãâã¨ãâã¨ undefined symbol ã¨ãâ©ã¾ãçºçãâã¾ã [mybox:user /src/php4] root# apachectl configtest apachectl: /usr/local/apache/bin/httpd Undefined symbols: _compress _uncompress
10. Windows: PHP ãâ¤ã³ãã¾ãã«ãâã®ã§ãâãâã¦ã¶ãâ PHP ã¹ã¯ãâãã«ã¢ã¯ã»ã¹ãâ¦ãâã¨ãæ¬¡ãâ®ãâãâãâ¨ãâ©ã¾ãâºåâãâã¾ãâ cgi error: The specified CGI application misbehaved by not returning a complete set of HTTP headers. The headers it did return are:
11. Windows: å¯¨ã¦ã®è²¬æãâåã¾ãâ¦ãâã®ã« PHP ã IIS ã§ãâãâã¾ãâ!
12. IIS, PWS, OmniHTTPD, Xitamiä ã§CGIã¨ãâã¦PHPãâ®è¡¨ãâãâ¨ãâ æ¬¡ã®ãâãâãâ¨ãâ©ã¾ãâºã:Security Alert! PHP CGI cannot be

なぜ、Apache 2 のマルチスレッド MPM モードを実運用環境で使用するべきではないのですか?

PHP はグルー(糊)です。このグルーは、多くのサードパーティ製のライブラリを くっつけることによりクールな Web アプリケーションを構築するために使用され、 直観的で簡単に習得できる言語インターフェイスにより、一つの整合性のある実体として 見せることができます。 PHP の柔軟性と力は、プラットフォームの安定性と堅牢性に基づいています。 グルーによる結合をするためには、OS や Web サーバ、サードパーティ製のライブラリを 必要とします。 これらの一つの機能が停止した場合、PHP は問題を特定し、速やかに修正する 手段を必要とします。 実行スレッドを完全に分離しなかったり、 メモリセグメントを完全に分離しなかったり、 各リクエストで使用される強力なサンドボックスを有さないことで、 基本的なフレームワークをより複雑なものにした場合、 PHP のシステムに弱点が生まれます。

マルチスレッド MPM を使用する必要がある場合、 PHP が自分のメモリ空間で実行される FastCGI 設定を観てみてください。

最後に、この警告はマルチスレッド MPM の使用に対するものですが、 Windows システムではより多くのライブラリがスレッドセーフである傾向が あるため、この警告の度合はより弱くなります。

Unix/Windows: php.ini ファイルはどこに置けばよいのですか?

UNIX の場合、デフォルトでは /usr/local/lib です。コンパイル時に --with-config-file-path オプションを使用してこの場所を変えたいという人も多いでしょう。 例えばこのようにすることも出来ます:

```
--with-config-file-path=/etc
```

そしてディストリビューションの php.ini-dist を /etc/php.ini にコピーし、環境に合うように 編集してください。

```
--with-config-file-scan-dir=PATH
```

Windows の場合、php.ini のデフォルトパスは Windows ディレクトリになります。 Apache ウェブサーバを使っている場合はまず Apache がインストールされているディレクトリ (例えば c:¥program files¥apache group¥apache にある php.ini を探そうとします。このため、異なる php.ini ファイルを異なるバージョンの Apache ごとに置いておくことができます。

設定ファイルの章も参照してください。

Unix: PHP をインストールしましたが、ファイルをロードするたびに 'Document Contains No Data(ページが表示できません)' というメッセージが表示されます。何が起こっているのですか?

これはおそらく PHP に何らかの問題が起こっていてコアダンプしている という状態です。サーバのエラーログを見てこのケースかどうか チェック してください。そして小さなテストケースで問題を再現させてみてください。 もし 'gdb' の使い方が分かるならバグレポートに加えてバックトレースを 提供してもらえると開発者が問題の箇所を突き止めるのに 非常に役立ちます。もしあなたがPHPをApacheモジュールとして使用している 場合は以下のようにします:

- httpd を停止します
- gdb httpd
- Stop your httpd processes
- > run -X -f /path/to/httpd.conf
- ブラウザから問題のある URL にアクセスします
- > run -X -f /path/to/httpd.conf
- もしコアダンプが発生すると gdb が知らせてくれます
- bt とタイプします
- このバックトレースをバグレポートに含めてください。バグレポートは » http://bugs.php.net/ から送信してください。

もしそのスクリプトが正規表現関数を使用している場合 (ereg()やその類似関数)、PHP と Apache が同じ正規表現の パッケージを使用してコンパイルされているかどうかを確認してください。 PHP と Apache 1.3.x を使用している場合は常に確認が必要です。

Unix: RPM を使って PHP をインストールしたのですが、Apache が PHP のページを 処理してくれません。何が起こっているのですか?

あなたが Apache と PHP の両方を RPM でインストールしたとすると、以下に示す 内容の一部もしくは全てを httpd.conf ファイルに追加するか、コメントを 外す必要があります:

```
# Extra Modules
AddModule mod_php.c
AddModule mod_php3.c
AddModule mod_perl.c

# Extra Modules
LoadModule php_module            modules/mod_php.so
LoadModule php3_module           modules/libphp3.so      # for PHP 3
LoadModule php4_module           modules/libphp4.so      # for PHP 4
LoadModule perl_module           modules/libperl.so
```

そして以下の行:

```
AddType application/x-httpd-php3 .php3      /* for PHP 3 */
AddType application/x-httpd-php .php        /* for PHP 4 */
```

以上の内容を全体設定、もしくは PHP サポートを追加したいバーチャル ドメインの設定に加えてください。

Unix: PHP 3 を RPM を使ってインストールしたのですが、その RPM は 私が必要とするデータベースのサポートなしでコンパイルされたものでした。 どうすればよいですか?

PHP 3 のビルド方法の制約により、どの環境にも対応できる柔軟な PHP の RPM を作成するのは困難です。この問題に関しては PHP 4 で取り組んでいます。 PHP 3 ではとりあえず PHP ディストリビューションの INSTALL.REDHAT ファイルで説明されている方法を用いることをお勧めします。 もしどうしても RPM バージョンの PHP 3 を使う、という場合は続きをお読みください。

RPM のパッケージャはインストール処理を単純化するため、 そして標準の /usr/local/ ディレクトリではなく /usr/ を使用するためにデータベースサポート無しの RPM を作成しています。 データベースサポートを追加するには RPM spec ファイルにサポートする データベースの種類と最上位ディレクトリを指定する必要があります。

以下の例は広く使われている MySQL データベースサポートを追加して Apache モジュールをインストールする場合について説明しています。

もちろん以下の全ての情報は PHP がサポートするいずれのデータベース の場合でも対象箇所を適宜変更すれば対応可能です。この場合は MySQL と Apache 共に RPM のみを使ってインストールしたものと仮定しています。

- まず mod_php3 を削除します。

  ```
  rpm -e mod_php3
  ```

- ソース RPM を手に入れてインストールします。--rebuild ではありません。

  ```
  rpm -Uvh mod_php3-3.0.5-2.src.rpm
  ```

- /usr/src/redhat/SPECS/mod_php3.spec を編集します。

  %build セクションに追加するデータベースサポートとそのパスを記述します。

  MySQL の場合は以下のようになります。 --with-mysql=/usr %build セクションはこのようになります。

  ```
  ./configure --prefix=/usr ¥
  --with-apxs=/usr/sbin/apxs ¥
  --with-config-file-path=/usr/lib ¥
  --enable-debug=no ¥
  --enable-safe-mode ¥
  --with-exec-dir=/usr/bin ¥
  --with-mysql=/usr ¥
  --with-system-regex
  ```

- 変更が終了したら以下のようにしてバイナリ RPM を構築します。

  ```
  rpm -bb /usr/src/redhat/SPECS/mod_php3.spec
  ```

- インストールします。

  ```
  rpm -ivh /usr/src/redhat/RPMS/i386/mod_php3-3.0.5-2.i386.rpm
  ```

Apache を再起動するのを忘れないでください。そうすれば PHP 3 には MySQL サポートが追加されているはずです。ただ、ディストリビューションの tar ファイルから構築してそこに含まれる INSTALL.REDHAT ファイルにしたがった方が おそらくずっと簡単である、ということも忘れないでください。

Unix: Apache に FrontPage エクステンションのパッチを当てたら、突然 PHP が動作しなくなってしまいました。PHP は Apache の FrontPage エクステンションと共存することはできないのですか?

いいえ。PHP は FrontPage エクステンションと問題なく共存できます。 問題は FrontPage エクステンションのパッチが PHP が依存している Apache の 構造の一部を変更してしまうことにあります。パッチを当てた後で PHP を再コンパイル('make clean; make' としてください)すれば問題は 解決されます。

Unix/Windows: PHP をインストールしたのですがブラウザから PHP スクリプトにアクセスしても何も表示されません。

ブラウザの「ソースの表示」を実行してください。おそらく PHP の ソースコードが表示されると思います。これはウェブサーバがスクリプトを PHP に渡していないためスクリプトが実行されていない、ということを 意味します。サーバ側の設定のどこかが間違っているはずですので、 PHP インストールマニュアルに従って再度入念に設定を確認してみてください。

Unix/Windows: PHP をインストールしたのですがブラウザから PHP スクリプトにアクセスすると Internal Server Error 500 となってしまいます。

ウェブサーバが PHP を実行するときに何らかの問題が起きています。 どんなエラーが起こっているか確認するために、コマンドラインで PHP 実行ファイル(Windows では php.exe) のある ディレクトリに移動して php -i を実行してください。 もし PHP の実行時に問題があった場合は適切なエラーメッセージが表示されるので、 それを手がかりに次に何をすべきかを知ることができます。画面 一杯に HTML(phpinfo() 関数の出力)が表示された場合には PHP は問題なく動作していますので、問題はウェブサーバの設定 にあるはずです。再度入念にチェックしてみてください。

PHP をインストールするまではエラーもなく問題なく進んだのですが、 apache を起動させようとすると undefined symbol エラーが発生します。

```
[mybox:user /src/php4] root# apachectl configtest
  apachectl: /usr/local/apache/bin/httpd Undefined symbols:
    _compress
    _uncompress
```

これは、PHP とは関係なく MySQL クライアントライブラリの問題です。 このライブラリのうちある種のものは --with-zlib を必要とし、他のものは必要としません。この問題は MySQL FAQ でも扱われています。

Windows: PHP をインストールしたのですがブラウザから PHP スクリプト にアクセスすると次のようなエラーが出力されます。

```
cgi error:
  The specified CGI application misbehaved by not
```

```
returning a complete set of HTTP headers.
The headers it did return are:
```

このエラーメッセージは PHP が(何らかの理由で)何も出力できなかった ことを意味します。詳細なエラーメッセージを得るためには、 コマンドラインから PHP 実行ファイル(Windows では php.exe)のあるディレクトリに移動して **php -i** を実行してください。もし PHP の実行時に問題 があった場合は適切なエラーメッセージが表示されるのでそれを 手がかりに次に何をすべきかを知ることができます。画面一杯に HTML (phpinfo()関数の出力) が表示された場合には PHP は問題なく動作しています。

PHP がコマンドラインで動作したなら、再度ブラウザから PHP スクリプトに アクセスしてみてください。もしまだ失敗するようなら以下のいずれかの 理由によるものと思われます。

- PHP スクリプト、 php.exe、php4ts.dll、 php.ini もしくはロードしようとしている PHP 拡張モジュールの ファイルパーミッションが匿名インターネットユーザ IUSER_<machinename> にアクセスできない ものになっている。
- スクリプトが存在しない(もしくはあなたが在ると思っている場所と ウェブサーバのルートディレクトリからの相対位置がずれている)。IIS を使用している場合は、Internet Service Manager でスクリプト マッピングを設定するときに「ファイルの存在を確認する」にチェックを することでこのエラーをトラップすることができます。もしスクリプト ファイルが存在しない場合はサーバが代わりに 404 エラーを返すように なります。これにはもう一つ利点があります。それは IIS が スクリプトファイルの NTLanMan パーミッションに応じて認証を要求する ようになる、ということです。

Windows: 全ての説明に従っているのに PHP が IIS で動作しません!

PHP スクリプトを実行しようとするあらゆるユーザが php.exe の実行権限を必要としているということを 忘れないでください。IIS はインストール時に追加された匿名ユーザを使用 します。このユーザに対して php.exe の実行権限が 必要です。また、認証された全てのユーザに関しても php.exe の実行権限が必要です。IIS4 の場合はさらに PHP がスクリプトエンジンであるということを教えてやる必要があります。 この FAQ も読んでください。

IIS, PWS, OmniHTTPD, Xitami上でCGIとしてPHPを実行するとき、 次のようなエラーが出る:Security Alert! PHP CGI cannot be accessed directly.

cgi.force_redirect に 0 をセットしてください。デフォルトでは 1 にセットされていますので、そのディレクティブが ; でコメントアウトされていないことを確認してください。 他のディレクティブと同様にこれは php.ini 上でセットされます。

デフォルトは 1 なので、100% 正しく php.ini ファイルが 読み込まれているかどうかが重要です。 詳細はこの FAQ を 読んでください。

php.ini が認識され読み込まれていることをどうやって知ることが できますか? 自分の行った変更が反映されてないらしいのですが。

php.ini とそれが PHP に読み込まれているかを確認するには phpinfo() をコールして最初のほうに 表示されている Configuration File (php.ini) を見てください。これは PHP が認識している php.ini と、それが 読み込まれているか否かを示しています。ディレクトリパスだけが 表示されている場合は、読み込まれていないということなので、 そのディレクトリに php.ini を置いてください。 php.ini が PATH にある場合それが読み込まれます。

php.ini が読み込まれていてかつ PHP をモジュールとして実行 している場合、php.ini を変更した後で必ず Web サーバを 再起動してください。

Windows で PHP のディレクトリを PATH に追加するにはどうすればいいのですか?

Windows NT、2000、XP および 2003 では、

- コントロールパネルのシステムアイコンを選択します(スタート -> 設定 -> コントロールパネル -> システム、あるいは Windows XP/2003 では 単に スタート -> コントロールパネル -> システム)。
- 詳細設定タブに移動します。
- 「環境変数」ボタンを押します。
- 「システム環境変数」欄を見ます。
- Path というエントリを見つけます(スクロールする必要があるかもしれません)。
- Path のエントリをダブルクリックします。
- その最後に ';' と追加し、その後に PHP のディレクトリを追加します (例: ;C:¥php)。
- OK を押し、コンピュータを再起動します。

Windows 98/Me では、autoexec.bat を編集する必要があります。

- メモ帳を開きます(スタート -> ファイル名を指定して実行 で、 notepad と入力します)。
- C:¥autoexec.bat ファイルを開きます。
- PATH=C:\WINDOWS;C:\WINDOWS\COMMAND;..... という行を探し、 その行の最後に ;C:¥php を追加します。
- ファイルを保存し、コンピュータを再起動します。

  注意: 上の作業を行った後は、必ずコンピュータを再起動し、 PATH の変更が適用されていることを確認しましょう。

PHP マニュアルでは、これまでファイルを Windows のシステムディレクトリに コピーするよう推奨してきました。なぜなら、このディレクトリ (C:¥Windows、C:¥WINNT、など) にはデフォルトでシステムの PATH が通っていたからです。ファイルを Windows のシステムディレクトリにコピーすることはずいぶん昔に非推奨と なっており、これはさまざまな問題のもととなりえます。

Windows 上の PHP で、php.ini を使えるようにするにはどうしたらいいのですか?

いくつかの方法があります。Apache を使用しているのなら、Apache 固有のインストール手順 (Apache 1、Apache 2) を参照ください。 それ以外の場合は、環境変数 PHPRC を指定する 必要があります。

Windows NT、2000、XP および 2003 では、

- コントロールパネルのシステムアイコンを選択します(スタート -> 設定 -> コントロールパネル -> システム、あるいは Windows XP/2003 では 単に スタート -> コントロールパネル -> システム)。
- 詳細設定タブに移動します。

- 「環境変数」ボタンを押します。
- 「システム環境変数」欄を見ます。
- 「新規」ボタンを押し、変数名に 'PHPRC' 、変数値に php.ini の 存在するディレクトリ（例: C:¥php）を入力します。
- OK を押し、コンピュータを再起動します。

Windows 98/Me では、autoexec.bat を編集する必要があります。

- メモ帳を開きます（スタート -> ファイル名を指定して実行 で、 notepad と入力します）。
- C:¥autoexec.bat ファイルを開きます。
- ファイルの最後に以下の内容の行を追加します。set PHPRC=C:¥php（C:¥php を、実際に php.ini が存在するディレクトリに置き換えます）。パスには空白文字を 含められないことに注意しましょう。たとえば、もし PHP を C:¥Program Files¥PHP にインストールしたのなら、 その代わりに C:¥PROGRA~1¥PHP と入力する必要が あります。
- ファイルを保存し、コンピュータを再起動します。

Apache のコンテントネゴシエーション（MultiViews オプション）を PHP で使用することはできますか?

もしリンクする PHP ファイルが拡張子を持っているなら、すべては うまく動作します。この FAQ が対象としているのは、リンクする PHP ファイルが拡張子を持っておらず、拡張子のない URL から PHP ファイルを判別するためにコンテントネゴシエーションを使用する方法です。この場合、AddType application/x-httpd-php .php という行を以下のように変更します。

```
# PHP 4
AddHandler php-script php
AddType text/html php

# PHP 5
AddHandler php5-script php
AddType text/html php
```

この方法は、Apache 1 ではうまく動作しません。なぜなら PHP モジュールが php-script をキャッチできないからです。

PHP が処理できるリクエストメソッドは GET および POST だけなのですか?

いいえ、それ以外のいかなるメソッド（例: CONNECT）でも扱えます。 適切な応答ステータスは、header() を使用して送信可能です。 GET および POST だけを処理したいなら、Apache の設定を以下のようにします。

```
<LimitExcept GET POST>
Deny from all
</LimitExcept>
```

構築時の問題

本節では、構築時に発生する多くの一般的なエラーを集めたものです。

1. anonymous CVS ãµã¼ãã¹ã«ãæ°çã® PHP åå¨¥æãã¾ãã configure ã¹ã¯ãªãã¾ãã!
2. PHPãApacheã¨çµã¿ãã¦åä½ãã¾ãã«configureè¡ãéã«åé¡ã ãã¾ãã¾ãhttpd.hãè¦ä»ãã¾ãã¨ãã¾ãã¾ãã æå®ãã¾ãå ´æã«ãã®ãã¡ã¤ã«ã¯ãã®ã§ã!
3. PHPãconfigureãã¦ãã¾ã¨ãã« (./configure) ä»¥ä¸ã®ãã¾ãã¨ã©ã¼ãééãã¾ãã¾ã checking lex output file root... ./configure: lex: command not found configure: error: cannot find output from lex; giving up
4. Apacheãèµ·åãã¨ãã«ä»¥ä¸ã®ãã¾ãã¡ãã»ã¼ã¸ ãå¤ã: fatal: relocation error: file /path/to/libphp4.so: symbol ap_block_alarms: referenced symbol not found
5. configureå®è¡ãéãéãGDãgdbmã¾ãã¨ä»®ãã¾ãã¡ã®ãã¡ãã¤ã«ã¾ ãã¯ã©ãã©ãªãè¦¤ãã¾ãã¨ãã¾ãã¨ãã¾ãã¨è¨ãã¾ãã¾ãã
6. ãã¡ãã¤ã«language-parser.tab.cãã³ãã¤ãã«ãéãé 'yytname undeclared'ã¨ãã¨ãã¨ã©ã¼ãçãã¾ãã
7. makeå®è¡ãéãéå®è¡ãã«ã¯æåã¦ãã¾ãã¾ãã³ãã³ãã¤ãã«ãã æçµçãªã¢ãã±ã¾ã¡ãã³ãã³ãã¯ãã¨ãéã«ä¾ãã¡ãã¤ãã«ãã¨ãã¨ã©ã¼ãçãã¾ãã
8. PHP ãã³ãã¯ãéãæå®ç¾©ã®åçã¾ãã¨ãã¨ã©ã¼ãçãã¾ãã
9. PHPãApache 1.3ã¨çµã¿ãã¾ã¦æ§ç¯ãæ³ãåãã¾ãã
10. ãã¤ãã¹ãã¾ãã«æé ã«å®å¨¥ã¨ã«ãºãUnixä¸ã§Apacheã¢ã¸ã¥ã¼ã¼ãã¤ãã¹ ãã¾ãã«ãã¾ãã¾ãã©ã¦ã¡ã¶ã¢ãã¯PHPã¹ã¯ãªãã¨è¨ãã¤ãã¹ãã¾ãã¨ãã¤ãã¤ãã¤ãã«ãã¿ãã¾ãã¨ãã¾ãã¨ãã¾ãã¾ãã
11. --activate-module=src/modules/php4/libphp4.a ã¨æ¸ãã¦ãã¾ãã¾ãã¾ãã¤ãã«ãã¾ãã¾ãã¾ãã¾ãã¾ãã --activate-module=src/modules/php4/libmodphp4.aã« å¤æ´ãã¾ãã¾ãã¾ãã¯ãã ãã§ãã¼ãèµ·åãã¦ãã¾ãã¾ãã§ãã?
12. --activate-module=src/modules/php4/libphp4.a ãæå®ãPHPãéçã¢ã¸ã¥ã¼ã¾ãã¨ã¨ãçµã¿è¾¼ãã§Apacheæ§ç¯ãã ãã¨ãéã«ãã·ã¹ãã ANSIå¯¾å¿ãã§ãã¾ãã¨ãã¨ã©ã¼ãçãã¾ãã
13. --with-apxs ãæå®ãã¦ PHP ãæ§ç¯ãã¾ãéåå¥å¦ãã¤ãã¨ã©ã¼ãã¤ãã»ã¾ãã¸ãåºåãã¾ãã
14. makeã®éãmicrotimeãã¾ã³ RUSAGE_éçéãã§å¤ãã®ã¨ã©ã¼çãã¾ãã
15. MySQLãä½¿ç¨ãã§ãã¾ãã«PHPãã³ã³ãã¤ãã«ãã¾ãéãconfigureã¯æ£å¸¸ã« å®è¡ãã¾ãã¾ãmakeã®æä¸ã«ã»ä»¥ä¸ã®ãã¤ãã¨ã©ã¼ãåºåãã¾ãã: ext/mysql/libmysql/my_tempnam.o(.text+0x46): In function my_tempnam': /php4/ext/mysql/libmysql/my_tempnam.c:103: the use of tempnam' is dangerous, better use mkstemp' ä½¿ãã¾ãã®ãï¼
16. PHPãçãã°ã¬ãã¾ãã®ãçãçç¾å¨ã¤ã³ã¹ãã¼ã«ãã¦ãPHPã æ§ç¯ãã¾ãã«æå®ãã./configureã³ãã³ãã®ãªãã·ãã³ã©ãã§ãã¾ãã¨åºæ¥ãã¾ãï¼
17. GDã©ã¤ãã©ãªãªã³ãã¦ãPHPæ§ç¯ãã¾ã³ãã³ãã¤ãã«ã¨ã©ã¼çãã¾ãå®è¡æã«ã»ã°ã¤ãã¾ã¾ãã·ã³éåã«ãªãã¦ãã¾ãã¾ãã¾ãã
18. PHP ãã³ãã³ãã¤ãã«ãééã«ãã¾ãã¾ãã¾ãã¨ã©ã¼çãã¦ãã³ã°ãã ãã¾ãã¾Solaris ãä½¿ç¨ãã¦ãã¾ãå ´åã«èµ·ãã¾ãã§ãã

anonymous CVS サービスにより最新版の PHP を入手しましたが、 configure スクリプトがありません!

configure.inからconfigureスクリプトを生成するためには、GNU autoconfパッケージが必要です。CVSサーバからソースを入手した後、 最上位のディレクトリで**./buildconf**を実行して下 さい。(また、configureに --enable-maintainer-modeオプションを付けて実 行した場合以外は、configureスクリプトはconfigure.in ファイルが更新された際に自動的に再構築を行いません。このため、 configure.inが 変更された場合には忘れずに手動で再構築を行う必要があります。 再構築の際に行われることの 1 つは、configureまたは config.statusを 実行した後、Makefileの中の@VARIABLE@のような物を見つけることです。)

PHPをApacheと組みあわせて動作するようにconfigureを行う際に問題が あります。httpd.hが見付からないといわれますが、 指定した場所にこのファイルはあるのです!

configure/setupスクリプトにApacheソースツリーの最上位の場所を指 定する必要があります。これは、--with-apache=/path/to/apache を指定するのであった、--with-apache=/path/to/apache/src ではないということです。

PHPをconfigureしているときに (./configure) 以下のようなエラーに遭遇しました。

```
checking lex output file root... ./configure: lex: command not found
configure: error: cannot find output from lex; giving up
```

インストールの説明をよく読み、 PHPのコンパイルにはflexとbisonの両方が必要であることに留意してください。 そーすかあるいはRPMのようなパッケージからbisonとflexを インストールしてください。

Apacheを起動するときに以下のようなメッセージが出る:

```
fatal: relocation error: file /path/to/libphp4.so:
symbol ap_block_alarms: referenced symbol not found
```

ApacheのcoreプログラムがsharedなDSOライブラリとして コンパイルされている場合にこのエラーが出ます。 最低でも以下のフラグを使用してApacheを再configureして ください:

```
--enable-shared=max --enable-rule=SHARED_CORE
```

詳細はApacheのソースディレクトリのトップレベルにある INSTALLファイル またはApacheの » DSO manual pageをご覧ください。

configureを実行した際、GD、gdbmまたは他のパッケージのファイルま たはライブラリを見つけることができないと言われます。

Cプリプロセッサおよびリンカに次のように追加でフラグを指定するこ とにより、configureスクリプトがヘッダファイルまたはライブラリを標準以外の場所で探すことが可能となります。

```
CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

ログインシェルとしてcsh系のシェルを使用している場合、次のように なります。

```
env CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

ファイルlanguage-parser.tab.cをコンパイルする際、 'yytname undeclared'というエラーが発生します。

bisonのバージョンを更新する必要があります。最新版は、» http://www.gnu.org/software/bison/bison.htmlにあります。

makeを実行する際、実行には成功しているようですがコンパイルする 最終的なアプリケーションをリンクしようとした際に何かファイルが見つからないというエラーが発生します。

古いバージョンのmakeには、functionsディレクトリにあるファイルの コンパイルされたものを同じディレクトリに正しく入れないものがあります。**cp *.o functions**を試しに実行してから **make** を再度実行し、解決されるか確認して下さ い。解決される場合には、GNU makeの最近のバージョンに更新するべき です。

PHP をリンクする際、未定義の参照があるというエラーが複数発生します。

最後に適切なライブラリが全てインクルードされているかどうか 確認してください。よくあるのは、'-ldl' や 指定したデータベースのサポート機能に必要なライブラリの指定が欠けているというものです。

Apache 1.2.x とリンクする場合には、適当な情報を Configuration ファイルの EXTRA_LIBS の行に追加し、Apache の Configure スクリプトを 再度実行したかどうかを確認してください。より詳細な情報については、 インストールの章 を参照ください。

何名かの人から、Apache とリンクする際に 'libphp4.a' の直後に '-ldl' を追加する必要があることも報告されています。

PHPをApache 1.3と組み合わせて構築する手法が分かりません。

この手順は実際には非常に簡単です。以下の手順に注意深く従って下さ い。

- » http://www.apache.org/dist/httpd/から最新 のApache 1.3配布ファイルを取得してください。
- これをどこか、例えば、 /usr/local/src/apache-1.3で ungzip、untarしてください。
- まず、 **./configure --with-apache=/<path>/apache-1.3**を実行し、PHP をコンパイルしてください。 (<path> の部分は実際のapache-1.3ディレクトリのパスに置 き換えてください)
- **make** を実行した後、make installを実行してください。これにより、PHPが構築され、 必要なファイルがApacheのソースツリーにコピーされます。
- ディレクトリを /<path>/apache-1.3/srcに移動し、 Configurationファイルを編集してください。 このファイルに以下を追加してください。: AddModule modules/php4/libphp4.a
- **./configure** を実行した後、 makeを実行してください。
- 以上の手順によりPHP対応のhttpdバイナリが構築できました。

注意: : Apache の新しい ./configureスクリプトを使用することも可能です。 Apache配布ファイルのREADME.configureファイル の指示を参照ください。また、PHP配布ファイルの INSTALL ファイルも参照ください。

インストール手順に完全に基づきUnix上でApacheモジュール版をインス トールしましたが、ブラウザでアクセスするとPHPスクリプトが表示され、ファイルを保存しますかと尋ねられます。

これは、何らかの理由によりPHPモジュールが起動していないことを意 味しています。助けを求める質問を行う前にまず以下を確認ください。

- 実行しているhttpdバイナリが構築したばかりの新しいhttpdバイナ リであることを確認してください。確認を行うには以下のように入力します。 /path/to/binary/httpd -l mod_php4.c がリストの中にない場合、正し いバイナリが実行されていません。正しいバイナリを見つけて、イ ンストールしてください。
- Apache .conf ファイルの１つに正しいMIME型 が追加されていることを確認してください。これは以下のようになり ます。 AddType application/x-httpd-php3 .php3 (PHP 3の場合) または AddType application/x-httpd-php .php (PHP 4の場合) また、このAddTypeの行が、 <Virtualhost> または <Directory> ブロックの中に隠 されて、テスト用のスクリプトの場所に適用できていないようなこ とがないことを確認してください。
- 最後に、デフォルトのApache設定ファイルの場所はApache 1.2と Apache 1.3の間で変更されています。AddTypeの行を追加した設定ファ イルが実際に読み込まれていることを確認してください。このファイ ルが正しく読み込まれている場合には、明らかな構文エラーをhttpd.confファイルの中に書き込んでしまったり、何らかの明らか な変更があった可能性があります。

**--activate-module=src/modules/php4/libphp4.a と書いてありますが、ファイルがありません。このため、
--activate-module=src/modules/php4/libmodphp4.aに 変更しましたが、やはりだめです。何が起きているのでしょう?**

libphp4.aファイルはこの時点では存在しない、ということに気をつけてください。 このファイルは、Apacheの構築時に自動的に作成されます。

**--activate-module=src/modules/php4/libphp4.a を指定し、PHPを静的モジュールとして組み込んでApacheを構築しよう とした際に、
システムがANSI対応ではないというエラーを発生します。**

Apacheのこのエラーメッセージは紛らわしく、より新しいバージョンで は修整されています。

--with-apxs を指定して PHP を構築しようとした際、奇妙なエラーメッセージが出力されます。

ここでは、確認すべきことが3点あります。まず、何らかの理由により、 Apacheがapxs Perlスクリプトを構築する際に適当なコンパイラやフラ グ変数を付けないで構築されてしまうことが時々あります。 使用するapxsの場所を見つけたら(**which apxs**コマンドを 試してみてください。/usr/local/apache/bin/apxs または/usr/sbin/apxs等にあります)、 以下の行を確認してください。

```
my $CFG_CFLAGS_SHLIB   = ' ';             # substituted via Makefile.tmpl
my $CFG_LD_SHLIB       = ' ';             # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = ' ';              # substituted via Makefile.tmpl
```

上記のようになっている場合は問題です。これらの行は空白になっているか 'q()'のような正しくない値になっていると思います。これを以下のように 変更してください。

```
my $CFG_CFLAGS_SHLIB   = '-fpic -DSHARED_MODULE'; # substituted via Makefile.tmpl
my $CFG_LD_SHLIB       = 'gcc';                    # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = q(-shared);                # substituted via Makefile.tmpl
```

可能性のある第2の問題は、RedHat-6.1と6.2でのみ存在する問題です。 RedHatが出荷した apxs スクリプトは壊れています。以下の行を見てください。

```
my $CFG_LIBEXECDIR     = 'modules';            # substituted via APACI install
```

上の行がある場合、これを次のように変更してください。

```
my $CFG_LIBEXECDIR     = '/usr/lib/apache'; # substituted via APACI install
```

最後に、Apacheのconfigure/再インストールを行います。その際、 **./configure** と**make**の間に **make clean**を行ってください。

makeの間、microtimeおよび RUSAGE_関連で多くのエラーを発生します。

インストール時に**make**を行っている際、以下のようなエラーを発生す るとした場合、

```
microtime.c: In function `php_if_getrusage':
microtime.c:94: storage size of `usg' isn't known
microtime.c:97: `RUSAGE_SELF' undeclared (first use in this function)
microtime.c:97: (Each undeclared identifier is reported only once
microtime.c:97: for each function it appears in.)
microtime.c:103: `RUSAGE_CHILDREN' undeclared (first use in this function)
make[3]: *** [microtime.lo] Error 1
make[3]: Leaving directory `/home/master/php-4.0.1/ext/standard'
make[2]: *** [all-recursive] Error 1
make[2]: Leaving directory `/home/master/php-4.0.1/ext/standard'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/home/master/php-4.0.1/ext'
make: *** [all-recursive] Error 1
```

あなたのシステムは壊れています。使用しているglibcと同じバージョンの glibc-develパッケージをインストールして、/usr/include ファイルを修正 する必要があります。この問題は、PHPの動作とは全く関係ありません。 これを示すには、次のような簡単なテストを行ってみてください。

```
$ cat >test.c <<X
#include <sys/resource.h>
X
$ gcc -E test.c >/dev/null
```

これが、エラーが発生する場合、インクルードファイルが見つからない ことがわかります。

**MySQLを使用できるようにPHPをコンパイルする際、configureは正常に 実行されるがmakeの最中に以下のようなエラーが出る:
ext/mysql/libmysql/my_tempnam.o(.text+0x46): In function my_tempnam': /php4/ext/mysql/libmysql/my_tempnam.c:103: the use of tempnam' is dangerous, better use mkstemp' 何がおかしいのか?**

まず、これはWarningであって致命的な エラーではないと認識することが重要です。makeの 最後のほうでこの出力がしばしば見られるため致命的なエラーと 思われがちですがそうではありません。もちろん、Warningが 起きるとコンパイラが停止するような設定をしている場合は別です。 また、MySQLサポートはデフォルトで構築されることにも 留意してください。

> **注意**: PHP4.3.2以降、ビルド(make)完了後に以下のようなテキストが出る ことがあります:
>
> ```
> Build complete.
> (It is safe to ignore warnings about tempnam and tmpnam).
> ```

PHPをアップグレードしたいのですが、現在インストールされているPHPを 構築するときに指定した./configureコマンドの オプションはどこで知ることが出来ますか？

現在インストールされているPHPを構築した時のディレクトリにある `config.nice` ファイルを見るか、もしもうそのディレクトリが無い場合は

<?php phpinfo(); ?>
というスクリプトを実行すれば、最初の方にある./configure という箇所に表示されます。

GDライブラリをリンクしてPHPを構築すると、コンパイルエラーが発生したり 実行時にセグメンテーション違反になってしまいます。

リンクしたGDライブラリとPHPが同じライブラリ(例えばlibpng)に依存しているか どうかを確認してください。

PHP をコンパイルする際に、よくわからないエラーが発生してハングアップ します。Solaris を使用している場合に起こるようです。

PHP のコンパイル時に 非 GNU ツールを使用すると問題を引き起こします。 PHP を正しくコンパイルするには、GNU ツールを使用することを忘れないで ください。たとえば、Solaris で使用される SunOS BSD 互換の sed や Solaris 版の sed は正しく動作しません。しかし、GNU 版もしくは Sun POSIX (xpg4) 版の sed ならうまく 動きます。以下を参照ください。 [» GNU sed]、 [» GNU flex] および [» GNU bison] 。

PHPを使う

このセクションにはPHPスクリプトを書くにあたってよく問題となる事柄が 集められています。

あらゆるフォームから送信されたデータを扱うことができる汎用的な PHPスクリプトを書きたいのですが、POSTメソッドでどのようなデータ が送信されたかを知るにはどうするのですか？

PHPは$_POSTのような [定義済みの変数] を沢山提供しています。$_POSTを連想配列として ループすることでPOSTされた全ての値にアクセスできます。例えば、 [foreach]で 単純にループして[empty()]で値をチェックし、 結果を出力します。

```
<?php
$empty = $post = array();
foreach ($_POST as $varname => $varvalue) {
    if (empty($varvalue)) {
        $empty[$varname] = $varvalue;
    } else {
        $post[$varname] = $varvalue;
    }
}

print "<pre>";
if (empty($empty)) {
    print "None of the POSTed values are empty, posted:¥n";
    var_dump($post);
} else {
    print "We have " . count($empty) . " empty values¥n";
    print "Posted:¥n"; var_dump($post);
    print "Empty:¥n";  var_dump($empty);
    exit;
}
```

```
?>
```

注意: **スーパーグローバル: 使用可能なバージョンに関する注意**PHP 4.1.0 以降、 $_GET, $_POST, $_SERVER 等のスーパーグローバル配列が使用可能となっています。詳細な情報については、マニュアルの superglobals のセクションを参照してください。

シングルクオート(')をバックスラッシュでエスケープされた シングルクオート(\')に変換しなければならないのですが、 正規表現を用いてこれを行うにはどの様にするのですか? 同様に " を \" に、\ を \\ に変換したいのです。

addslashes()がこれを行ってくれます。 mysql_escape_string()も見てください。 stripslashes()によって バックスラッシュを除去することもできます。

注意: **ディレクティブに関する注意: magic_quotes_gpc**PHP ディレクティブmagic_quotes_gpc のデフォルトは on です。この場合、すべての GET, POST, COOKIE データについて addslashes() が実行されます。これらを取り除くため stripslashes() を使用することができます。

" は \" に、また ' は \' に全て変換されているのですが、 これら無用なバックスラッシュを除去するにはどうしたらよいですか? どうやって、そしてなぜ、こんなことになっているのでしょう?

stripslashes()関数によってstringから バックスラッシュを取り除くことができます。 このような奇妙なバックスラッシュは、ほとんどの場合、PHPの magic_quotes_gpc ディレクティブがオンになっていることによって付加されています。

注意: **ディレクティブに関する注意: magic_quotes_gpc**PHP ディレクティブmagic_quotes_gpc のデフォルトは on です。この場合、すべての GET, POST, COOKIE データについて addslashes() が実行されます。これらを取り除くため stripslashes() を使用することができます。

次のようなコードを実行すると、思った通りの順番で出力が表示されま せん。

```
function  myfunc($argument)
{
     echo  $argument + 10;
}
$variable  =  10;
echo  "myfunc($variable)  =  "  .  myfunc($variable);
```

なぜですか?

式の中で関数の実行結果を使用する(例えば上の例の様に他の文字列と 連結する)ためには、echo()するのではなく、その 値をreturnしなければいけません。

改行されないのですが?

```
<pre>
<?php  echo  "これは1行目"; ?>
<?php  echo  "この行は改行に続いて出力されるはず"; ?>
</pre>
```

PHPでは、"?>"か"?>\n"(\nは改行を表します)をPHPのコードブロッ クの終端と見なします。このため、コードブロック終端の改行記号は省略され、表示される文は1行になります。つまり、改行をさせるために は、PHPのコードブロックの終端の後にもう1つ改行を挿入する必要があるということです。

なぜPHPはこのようなことをするのでしょうか?なぜならHTMLを出力する 場合にはこの方が都合のよいことが多いからです。もしとても長い1行を 出力しなければならない場合に、改行が解釈されてしまうとしたらどう でしょう。ソースコードの1行もとても読めないくらい長いものになって しまいます。

'Warning: Cannot send session cookie - headers already sent...'や 'Cannot add header information - headers already set...'といった メッセージが出力されるのですが。 sent...'.

header(), **set_cookie()**や セッション関数は出力ストリームに ヘッダを付加する関数で、ヘッダを送信できるのは本文の出力を 開始する前のみです。headers_sent()を使って 既にヘッダが送信済みでないかチェックすることができます。 出力制御関数もご覧ください。

リクエストヘッダに直接アクセスしたいのですが、どうすればよいですか?

もしPHPがApacheモジュールとして動作しているなら、 getallheaders()を使えば全てのヘッダを取得する ことができます。下のちょっとしたコードで全てのリクエストヘッダを 表示することができます。

```
$headers  =  getallheaders();
foreach  ($headers  as  $name  =>  $content)  {
      echo  "headers[$name]  =  $content<br />¥n";
}
```

apache_lookup_uri(), apache_response_headers(), fsockopen()も参照してください。

IISで認証を行おうとすると'No Input file specified'というエラーが 発生します。

これはIISのセキュリティモデルの欠点で、IISで動作するCGIに共通する 問題です。これを回避策するには、認証のかかったディレクトリに(PHP が解釈しない)HTMLファイルを作成します。そしてMETAタグを使ってPHP を使用したページにリダイレクトするか、リンクを張ります。こうすれ ばPHPは認証済みかどうかを正しく認識することが出来ます。ISAPIモジュー ルの場合はこの問題は起きません。また、これは他のNT ウェブサーバに は影響ありません。詳しくは» http://support.microsoft.com/kb/q160422/ とHTTP 認証を 参照してください。

Windows: 他のコンピュータと共有しているファイルに、IIS でアクセスできません。

Go to Internet Information Services を変更する必要があります。PHP ファイルを選択して プロパティを開き、セキュリティ タブに移動し、 Edit -< Anonymous access and authentication control 。

この問題を解決するには Anonymous Access のチェックをはずして Integrated Window Authentication をチェックしたままにしておくか、 あるいは Anonymous Access をチェックしてアクセスできないユーザを別途指定します。

私が書いたPHPスクリプトはIEとLynxでは動作するのですが、Netscapeを 使うと出力の一部が失われてしまいます。"ソースの表示"をするとIEには あるがNetscapeにはない内容があります。

NetscapeはHTMLタグの扱いがIEに比べて厳密になっています(テーブル等)。 スクリプトが出力したHTMLを» validator.w3.orgなどのHTMLバリデータに掛けてみると良いでしょう。 例えば</table>が無いとこのような現象が発生します。

また、IEとLynxは共にHTMLストリーム中のNULL文字(¥0)を 無視しますがNetscapeは無視しません。この問題をチェックする一番の方法 はコマンドライン版の PHP(つまりCGIバージョン)をコンパイルして、コマンドラインからその スクリプトを実行することです。*nix上では、その出力を od -c にパイプして¥0があるかどうかを チェックしてください。もしWindowsを使っている場合は、バイナリファ イルを扱えるエディタかそれに類するソフトウェアが必要です。IEや Lynxと異なりNetscapeはNULL文字を見つけるとその行の文字を一切出力 しません。

どうすればXMLとPHPは共存することが出来るのですか？XMLの <?xml>タグがPHPでエラーになります。

<?xml を直接PHPコードに埋め込むには、 PHPディレクティブのshort_tags を0に設定しなければなりません。 このディレクティブは ini_set()でセットすることは できません。 short_open_tags のオン/オフに関わらず、次のようにもできます: <?php echo '<?xml'; ?> このディレクティブはデフォルトでオンです。

FrontPageやその他のHTMLエディタを使うと、書いたPHPのコードが勝手 にあちこちに移動されてしまいます。どうしたらいいですか？

最も簡単なのはASPタグを使う方法です。こうすればASPの様に<%と %>をコード区切りとして使用することが出来ます。有名なHTMLエディ タは(今のところ)ASPタグを賢く扱ってくれます。ASPスタイルのタグを 有効にするにはphp.iniファイルでasp_tagsを設定するか、適切なApache ディレ クティブを使用します。

あらかじめ設定されている変数を全て網羅したリストはどこにあるので すか？なぜPHPのドキュメントにはその一覧がないのですか？

マニュアルの 定義済みの変数 のページを読んでください。スクリプト上で有効な定義済み変数のリストの 一部があります。有効な変数の完全なリスト（とその詳しい情報）は phpinfo()をコールすることで見ることができます。 マニュアルの PHPの外部から来る変数 のセクションも読んでください。HTMLフォームやCookie、URL等から 来る外部変数に関するシナリオが説明されています。

> **注意**: **register_globals: 重要な注意**PHP 4.2.0 以降、PHP ディレクティブ register_globals のデフォルト値は off となっています。また、このディレクティブは PHP 6.0.0 で完全に削除される予定です。 PHP コミュニティは、ユーザがこのディレクティブの設定に依存せず、 superglobals のような他の手段を使用することを推奨します。

フリーではない商用ライブラリである ClibPDFや PDFLibを使わずに PDFファイルを生成するにはどうしたらよいでしょうか？ フリーのもので、外部のPDFライブラリを必要としない ものがいいのです。

PHPで書かれている代替手段がいくつかあります。 » http://www.ros.co.nz/pdf/, » http://www.fpdf.org/, » http://www.gnuvox.com/pdf4php/, » http://www.potentialtech.com/ppl.php等です。 » Pandaモジュールもあります。

ユーザ定義関数の中で、標準的な($DOCUMENT_ROOTや$HTTP_REFERER等の) CGI変数にアクセスしたいのですが、PHPはそれらの変数を見つけること ができていないようです。何がおかしいのですか？

PHPディレクティブの register_globals がサーバと環境変数に与える影響を認識することは重要なことです。 register_globals = off (PHP4.2.0以降デフォルトでoff)の場合、 $DOCUMENT_ROOTは存在しません。代わりに $_SERVER['DOCUMENT_ROOT']を使用してください。 register_globals = on の場合は $DOCUMENT_ROOTも $GLOBALS['DOCUMENT_ROOT']も存在します。

> **注意**: **スーパーグローバル: 使用可能なバージョンに関する注意**PHP 4.1.0 以降、 $_GET, $_POST, $_SERVER 等のスーパーグローバル配列が使用可能となっています。詳細な情報については、マニュアルの superglobals のセクションを参照してください。

いくつかの PHP ディレクティブでは、バイト値を integer ではなく省略形で設定できます。この省略形で使えるオプションを教えてください。 また、これは php.ini 以外でも使えるのですか ？

有効なオプションは K（キロバイト）、M（メガバイト）および G（ギガバイト: PHP 5.1.0 以降で有効）です。大文字小文字は区別しません。これ以外の文字は バイト値と判断されます。 1M は、1 メガバイトあるいは 1048576 バイトと等しくなります。1K は、1 キロバイトあるいは 1024 バイトです。この表記法を php.ini 以外では使用 しないでください。かわりに integer のバイト値を設定します。 これらの値を数値に変換する例は ini_get() のドキュメントを 参照ください。

PHP と HTML

PHP と HTML は深く関係しています。PHP は HTML を生成し、HTML には PHP に 送信される情報が記述されています。 以下の FAQ を読む前に、どうやって PHP の外部から来る変数を取得するかを読んでおくことは重要です。 このマニュアルにはこのトピックに関するよい例があります。 register_globals の意味するところにも 注意を払ってください。

1. ãã©ã¼ã ãâãâãâã¯ URL ãâå¤æ´ï¼åå ´å«ã¯ã©ãâãâ£ã ã¨ã³ã³ã¼ãâ/ãâ³ã¼ãâåâ¿[ï¼è¦ï¼ãªã®ã§ãâï¼
2. <input type="image"> ã¿ãªãâ¿ãâã¨ãâ¦ãâã®ã§ãâåâ¤æ° $foo.x ã¨ $foo.y ãâ¿ãâ¾ãâãâ©ãâãâªãâã®ã§ãâ?
3. HTML ãã©ã¼ã ã§é[ï¼]åâãâ¿ç¨ãâã«ã¯ã©ãâãâªãâã§ãâï¼
4. "select multiple" ã¿ãªã®ã§é æâãâå[ï¼]¨ã¦ã®ç¢ºèªæâåâ¾ãâã«ã¯ ã©ãâãâªãâã§ãâ?
5. Javascript ãâ PHP ã«å¤æ´æâ¦ãâã«ã¯?

フォームから、もしくは URL から値を渡す場合にはどういった エンコード/デコードが必要なのですか？

エンコードが重要になる場面はいくつかあります。 string $data という エンコードされていない文字列データを渡す場合について考えてみると、

- HTML を通じて渡す場合： 文字列にはどのような値が含まれるか分からないので、 データは必ず htmlspecialchars を行い、 ダブルクオートで囲まなければなりません。
- URL を通じて渡す場合: URL はいくつかのパーツから成り立ちます。 このデータをそのパーツのうちの一つであると解釈させたいならば、 urlencode() でエンコード しなければなりません。

Example#1 HTML の hidden 要素

```
<?php
    echo "<input type='hidden' value='" . htmlspecialchars($data) . "' />¥n";
?>
```

> **注意**: $data を urlencode() をしては いけません。なぜなら、その作業はブラウザに任されているからです。 一般に普及して

いる全てのブラウザは正しくこの処理を行ってくれます。 ただ、この処理はメソッド(GET や POST)が何であるかにかかわらずに行われるということに気をつけてください。この処理に気づくのは GET リクエストのときだけになるでしょう。なぜなら POST リクエストの内容は通常目に触れることは無いからです。

Example#2 ユーザによって編集するデータ

```
<?php
    echo "<textarea  name='mydata'>¥n";
    echo htmlspecialchars($data)."¥n";
    echo "</textarea>";
?>
```

注意: ブラウザはエスケープされたシンボルを解釈するので、data は 意図したとおりに表示されます。 フォームの内容を送信するとき、GET か POST かにかかわらず data は ブラウザによって URL エンコードされ、PHP によって URL デコードされます。 要は、URL エンコード/デコードを自分で行う必要はなく、これらの処理は すべて自動的に行われると言うことです。

Example#3 URL 中の場合

```
<?php
    echo "<a href='" . htmlspecialchars("/nextpage.php?stage=23&data="  .
        urlencode($data)) . "'>¥n";
?>
```

注意: この例では、実は GET リクエストを模擬しています。このため、data を手動で urlencode() する必要があります。

注意: 全ての URL を htmlspecialchars() する必要があります。 なぜなら、この URL は HTML の value 属性として扱われるからです。 この場合は、ブラウザはまず htmlspecialchars() された データを元に戻し、それから URL を渡します。URL は urlencode() されているので、PHP はこれを正しく 解釈することができます。 URL 中の & が & に置き換えられていることに気づくでしょう。もしあなたがこれを忘れても ほとんどのブラウザは元に戻してくれますが、必ずそうしてくれるとは 限りませんので、URL が動的に変更されるものでなくても URL は htmlspecialchars() されるべき です。

<input type="image"> タグを使おうとしているのですが、変数 $foo.x と $foo.y が使えません。どうすればよいのですか?

以下のようなタグを使えば、標準のボタンの代わりに画像を使用して フォームを送信することができます。

```
<input  type="image"  src="image.gif"  name="foo"  />
```

ユーザが画像のどこかをクリックすると、そのフォームの内容に foo.x と foo.y という 2 つの変数が追加され、サーバに送信されます。

PHP では $foo.x と $foo.y という名前は変数名として正しくないので、 自動的に $foo_x と $foo_y という名前に変換されます。要は、ピリオドが アンダースコアに置き換えられる、と言うことです。そのため、これらの 変数にアクセスする際には PHP の外部から来る変数 の取得についてのセクションで記述されているのと同様な方法を とります。たとえば $_GET['foo_x'] などです。

注意: リクエスト変数名の中のスペースは、アンダースコアに置き換えられます。

HTML フォームで配列を使用するにはどうすればよいですか?

フォームの内容を PHP スクリプトで配列として受け取るには、 <input>、<select> あるいは <textarea> といった要素の name を以下のように指定します:

```
<input  name="MyArray[]"  />
<input  name="MyArray[]"  />
<input  name="MyArray[]"  />
<input  name="MyArray[]"  />
```

変数名の最後にある括弧に注意してください。これにより、フォー ムの内容が配列として扱われます。異なる要素に同じ名前をつけること で要素を配列にグループ分けすることができます。

```
<input  name="MyArray[]"  />
<input  name="MyArray[]"  />
<input  name="MyOtherArray[]"  />
<input  name="MyOtherArray[]"  />
```

上記の HTML の場合、MyArray と MyOtherArray という 2 つの配列が生成され、 PHP スクリプトに送信されます。また、配列に特定のキーを設定する こともできます。

```
<input  name="AnotherArray[]"  />
<input  name="AnotherArray[]"  />
<input  name="AnotherArray[email]"  />
<input  name="AnotherArray[phone]"  />
```

この場合、配列 AnotherArray のキーは 0、1、email そして phone となります。

注意: HTML に配列のキーを指定するかどうかは自由です。キーを指定しなかった 場合はフォームに現れる順番に番号がつけられます。最初の例だと、 キーは 0、1、2、3 となります。

配列関数と PHP の外部から来る変数 も参照ください。

"select multiple" タグで選択された全ての結果を取得するには どうすればよいですか?

"select multiple" タグを使うと、ユーザはリストから複数の項目を 選択することができるようになります。選択された項目はフォームの action で指定されたハンドラに渡されます。問題は、これらの値が全て 同じ名前で渡されることです。つまり、

```
<select  name="var"  multiple="yes">
```

選択されたそれぞれの項目は action のハンドラに次のように渡されます：

```
var=option1
var=option2
var=option3
```

それぞれの項目は前の変数 $var の値を上書きして しまいます。この問題を解決するには、PHPの "フォームの値を配列にする" 機能を使います。以下のようにするとよいでしょう。

```
<select  name="var[]"  multiple="yes">
```

こうすれば PHP に $var を配列として扱うように 知らせることができ、各項目の value の値は配列の要素として var[] に 追加されます。最初の項目は $var[0] になり、 次の項目は $var[1]... というようになります。 count() 関数を使えば選択された項目の数を知る ことができます。またもし必要なら sort() 関数を 使ってソートを行うこともできます。

JavaScript を使っている場合、フォーム要素に要素名を使って(訳注： document.myform.myelement.value 等の様に)アクセスしようとすると、 要素名に含まれる [] が問題となることがあるので 気をつけてください。この場合は、数字で表されるフォーム要素の ID を 使用するか、シングルクオートで要素名を囲んでフォーム要素の配列の インデックスとしてアクセスしてください。例えば、以下のようにします：

```
variable  =  documents.forms[0].elements['var[]'];
```

Javascript から PHP に変数を渡すには?

Javascript は（普通は）クライアントサイド技術であり、一方 PHP は（普通は） サーバーサイド技術です。また HTTP は"ステートレスな" プロトコルです。 そのため、この二つの言語はダイレクトに変数を共有することができません。

しかしながら、この二つの言語の間で変数を渡すことは可能です。 一つの方法は PHP と一緒に Javascript のコードを生成し、 ブラウザに自動的にリフレッシュ（再ロード）させることです。 以下の例はまさにそれで、PHP に画面の高さと幅を認識させています。 これは通常はクライアントサイドでしかできないことです。

```
<?php
if (isset($_GET['width']) AND isset($_GET['height'])) {
  // ジオメトリ値を出力する
  echo "画面の幅:  ". $_GET['width'] ."<br />¥n";
  echo "画面の高さ:  ". $_GET['height'] ."<br />¥n";
} else {
  // ジオメトリ変数を渡す
  // （元のクエリ文字列を保持する
  //    -- POST 変数は別の方法で扱う必要がある）

  echo "<script  language='javascript'>¥n";
  echo "    location.href=¥"${_SERVER['SCRIPT_NAME']}?${_SERVER['QUERY_STRING']}"
          . "&width=¥"  +  screen.width  +  ¥"&height=¥"  +  screen.height;¥n";
  echo "</script>¥n";
   exit();
}
?>
```

PHP と COM

PHP は Win32 プラットフォーム上で COM と DCOM オブジェクトに アクセスすることができます。

1. ã¨ãâè¨ç®âèıã DLL âäı¼æâã¾âãâãâã PHP ã§å®è¡âãâæ¹æ³ã¯ âãâ¾âã?
2. 'Unsupported variant type: xxxx (0xxxxx)'ã¨ã¯ã©ãâãæå³ã§ã§âã?
3. PHP ã§âã¸ã¥ã¢ãªâã¸ã§ã¯ãäœ¿ãã§âãâ¾âã?
4. COM ãªâã¸ã§ã¯ãã»ãâãã»ãã·ã§ã³æ[X]å ±ã¨ãã¦ä¿å§âã¾âã?
5. COM ã®å¤åãâã¨ã©ı¼ãâã©ãâã§ã¾âã?
6. PHP ã¹ã¯ãªãâãâã DLL âäı¼æâãâã¨ã¯ã§ã¾âã? Perl ã§ã¯ ã§âãâã®ã§âãâ
7. 'Unable to obtain IDispatch interface for CLSID {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}'ã¨ã¯ã©ãâãæå³ã§ã§âã?
8. COM ãªâã¸ã§ã¯ãã¨ãâãâãµı¼âãâãå®è¡ãâã«ã¯ãâãâã°ãâã§ã§âã?
9. 'DCOM is disabled in C:\path...\scriptname.php on line 6' ã¨ãâã ã¨ã©ã¼ç¤âã¾âãâã©ãâãâã°ãâã§ãâï¼
10. PHP ãäı¿ç¨âãâã¾ã ã§ ActiveX ãªâã¸ã§ã¯ãâã¾/æäı¼âãâã¨ã¯ ã§âã¾âã?
11. å®è¡äã¨ã®ã³ã³ãâã¾ãâã®ã¤ã³ã¹ã¿ã³ã¹åå¾âãâã¨ã¯ã§âã¾âã?
12. COM ãªâã¸ã§ã¯ãã§ãâãâã¦ãâã³ãâæ±æ¹æ³ã¯ãâã¾âãâï¼
13. 2 ã¤ä»¥ä¸ã®ã¤ã³ã¿ãã§ã¾âã§ã¾ã¹ãâã¦ãâ¬éâã¦ãâ COM ãªâã¸ã§ã¯ãã¨ã¾âã¾ãâã¦ãâå°ã£ã¦ãâã¾âãâã©ãâãâã°ãâã§ã§âã?
14. PHP ã¯ COM ãæ±âãâ¨ãâãâ¨ã¯åâãâã¾âãâã§ã¯ COM+ ã«ã¤ãâã¦ ã©ãâã§ã§âã?
15. PHP ã COM ãªâã¸ã§ã¯ãã§ã¯âãâãâ¨ãâã§ãâãâ¨ã¯ãâãMTS ã ä½¿ã£ã¦ã¦ã³ã³ãâã¾ãâã¹ã¾ã¹ã¹ãæ±ãâãâ¨èâ¦ãâãâ¨ãâãâ¨ã§ãâï¼

とある計算を行う DLL を作成しました。これを PHP で実行させる方法は ありますか?

もしそれが普通の DLL なら、現在のところ PHP からそれを実行する手段は ありません。DLL が COM サーバを含んでいて IDispatch インターフェースが 実装されている場合、PHP からアクセスすることができます。

'Unsupported variant type: xxxx (0xxxxx)'とはどういう意味ですか?

多くの VARIANT タイプとその組み合わせがあり、そのほとんどは サポートされていますが、残りのいくつかは未だ実装されていません。 また配列も完全にサポートされているわけではなく、PHP と COM の間で 受け渡しが可能なのは 1 次元の添字付配列のみです。もしこの他に サポートされていないタイプを見つけた場合は(既知でなければ)、 できる限りの情報を添えてバグとして報告してください。

PHP でビジュアルオブジェクトを操作できますか?

基本的には可能です。しかし PHP はほとんどの場合ウェブスクリプティングの 言語として使用されウェブサーバ上で実行されるため、ビジュアルオブジェクトが サーバのデスクトップに現れることはありません。もし PHP を アプリケーション作成に使用する、すなわち PHP-GTK と合わせて 使用する場合にはアクセスに制限はありません。COM を通じて ビジュアルオブジェクトを操作することができます。

COM オブジェクトをセッション情報として保存できますか?

できません。COM インスタンスはリソースとして扱われるため、1 つの スクリプトを実行している間のみ使用可能です。

COM の出力するエラーをトラップできますか?

PHP 5 では、COM 拡張モジュールは com_exception 例外をスローします。これをキャッチし、code メンバを調べることで次にすべきことを決定できます。

PHP 4 では、PHP が提供する方法(@, track_errors, ...)を使用して COM のエラーをトラップすることはできません。

PHP スクリプトから DLL を作成することはできますか? Perl では できるのですが。

いいえ、残念ながら PHP にはそのようなツールはありません。

'Unable to obtain IDispatch interface for CLSID {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}'とはどういう意味ですか?

このエラーが発生する理由はいくつかあります。

- CLSID が正しくない
- 要求された DLL が無い
- 要求されたコンポーネントが IDispatch インターフェースを実装していない

COM オブジェクトをリモートサーバから実行するにはどうすればよいですか?

ローカルオブジェクトを実行するのと全く同様です。ただ、COM コンストラクタの 2 番目のパラメータにリモートマシンの IP アドレスを 渡すだけです。

php.iniに COM=TRUE をセットするのを忘れないでください。

'DCOM is disabled in C:\path...\scriptname.php on line 6' という エラーが発生しました。どうすればよいですか?

php.ini を編集して COM=TRUE としてください。

PHP を使用したページで ActiveX オブジェクトをロード/操作することは できますか?

これは PHP とは関係ありません。ActiveX オブジェクトは、HTML ドキュメントから要求があった場合にクライアント側でロードされます。PHP スクリプトとは関係が無く、従ってサーバ側と直接やりとりすることも できません。

実行中のコンポーネントのインスタンスを取得することはできますか?

モニカーを使用すれば可能です。同じ Word インスタンスに対して複数の 参照を取得したい場合は、以下のようにしてインスタンスを生成します。

```
<?php
$word = new COM("C:¥docs¥word.doc");
?>
```

こうすれば、実行中のインスタンスが無い、もしくはインスタンスの 取得が不可能だった場合には新規に生成され、インスタンスが取得できた場合にはそのハンドルを返します。

COM オブジェクトから送られてくるイベントを扱う方法はありますか?

イベントシンクを定義し、com_event_sink() を使用してそれをバインドすることが可能です。PHP でイベントシンククラスの 雛形を作成するために com_print_typeinfo() を 使用することが可能です。

2 つ以上のインターフェースを公開している COM オブジェクトのメソッドを 呼び出そうとして困っています。どうすればよいですか?

この問題に対する答えは、簡単ですが残念なものです。正確には わかりませんが、おそらく打つ手はありません。もしこの問題に関する 具体的な情報があったら» 私 に教えてください。

PHP は COM を扱えるということは分かりました。では COM+ については どうですか?

COM+ とは MTS(Microsoft Transaction Server) と MSMQ(Microsfot Message Queue Service) を通じてコンポーネントを操作するフレームワークによって COM を拡張したものです。が、PHP がそういった コンポーネントをサポートするにあたって特別に必要なことはありません。

PHP が COM オブジェクトを操作することができるということは、MTS を 使ってコンポーネントリソースを扱えると考えてもよいということですか?

PHP それ自体はまだトランザクションをサポートしていません。従って エラーが発生してもロールバック処理は行われません。もし トランザクションをサポートするコンポーネントを使用する場合は、 自分でトランザクション処理を実装する必要があります。

PHP と他の言語

PHP はウェブプログラミングをするには最高の言語です。が、 他の言語はどうなのでしょうか?

PHP vs ASP?

ASP それ自体は言語ではなく、Active Server Pages の頭文字です。ASP を作成するのに使用される実際の言語は Visual Basic Script か JScript です。 ASP の最大の欠点は、それが Microsoft Internet Information Server (IIS) でのみ動作するメーカー独自のシステムだという点です。これにより、ASP は Win32 ベースのサーバ上でしか動作しません。ASP を他のプラットフォーム、 ウェブサーバで動作させようといういくつかのプロジェクトが進行中です。 » Halcyon の » InstantASP(商用)、» Chili!Soft の Chili!Soft ASP(商用)、そして » ActiveScripting.org の OpenASP です。 ASP は PHP に比べて遅く、面倒で安定性も低いといわれています。ASP のプロフェッショナルの人たちの中には、ASP では主に VBScript が使用されているので Visual Basic でのプログラミングに慣れている人に とっては覚えるのが比較的簡単だという人もいます。また、IIS サーバでは ASP サポートがデフォルトで有効になっているので設定が簡単という利点もあります。 ASP に組み込まれているコンポーネントは非常に限られていて、もし FTP サーバと協調するといった「進んだ」機能を使おうと思ったら 別途コンポーネントを購入しなければなりません。

ASP から PHP へのコンバータはありますか?

はい。サーバーサイドの » asp2php は » そのクライアントサイド オプションと同じくらい有名です。

PHP vs Cold Fusion?

世間では PHP は Cold Fusion に比べて速く、複雑なプログラミングを 効率的に行うことができ、新しいアイデアを意欲的に取り込んでいると 言われています。また、安定していてより少ないリソースで動作するというのが 一般的な意見です。Cold Fusion はエラー処理、データベースの抽象化、 日付の扱いにおいて優れていますが、データベースの抽象化については PHP 4 で行われています。Cold Fusion の利点として挙げられることの一つに、 その優れたサーチエンジンがありますが、サーチエンジンはウェブスクリプト言語に 含まれるべき機能ではないのではないかという意見もあります。PHP は ほとんど全てのプラットフォームで動作しますが Cold Fusionが 動作するのは Win32, Solaris, Linux, HP/UX だけです。PHP は初期段階から多少の プログラミングに関する知識を必要としますが、Cold Fusion にはすばらしい IDE があって、一般にとっつきやすいと言われています。Cold Fusion は プログラマではない人たちも意識してデザインされていますが、PHP は プログラマを主なターゲットとしています。

Michael J Sheldon によるこのトピックに関するすばらしいサマリが PHP のメーリングリストに投稿されました。» http://marc.theaimsgroup.com/?l=php-general&m=956021674125 42&w=1 でその内容を読むことができます。

PHP vs Perl?

PHP の Perl に対して最も有利な点は、Perl が何もかもやろうとしてその結果 複雑になってきているのに対して、PHP はウェブスクリプティングを念頭において デザインされていることです。Perl の柔軟さ/複雑さは、他人にとって非常に 読みにくいコードが簡単に書けてしまうということでもあります。PHPは、柔軟さを 失うことなくそれほどややこしくもなく厳密でもない構成になっています。 PHP は Perl に比べ、簡単に既存の HTML に統合することができます。PHP には Perl で使用されている非常に多くの「良い」機能があります。構成や文法等です。 そしてこれらが Perl ほど複雑になることなく提供されます。Perl は 80 年代後半から 在る十分に枯れた真の言語ですが、PHP は急速に成長しています。

PHP 2からPHP 3への移行

PHPにはすでに長い歴史があります。あのPHP 1.0, PHP/FI, PHP 3.0 そして PHP 4.0です。

PHP 2からPHP 3への移行に関して

PHP/FI 2.0はもうサポートされていません。PHP/FI 2.0からの移行に関しては 移行に関する付録の該当セクションを参 照してください。

もしあなたがまだPHP 2を使用している場合、私たちは(PHP 3ではなく)PHP 4へ のアップグレードを強くお勧めします。

PHP 3 から PHP 4 への移行

PHPにはすでに長い歴史があります。あの PHP 1.0、PHP/FI、PHP 3.0 そして PHP 4.0です。

PHP 3 から PHP 4 への移行に関して

PHP 4 は以前のバージョンとできる限りの互換性を保つようにデザインされていて、 失われた相関性は極わずかです。 もし互換性に不安がある場合は PHP 4 をテスト環境にインストールしてみてスクリプトを実行してみてください。

また、マニュアルの 移行に関する付録の適切な箇所 も参照してください。

セションは PHP 3 で動きますか。

ネイティブでのセッションサポート は PHP 3 に存在しませんでしたが、 セッション機能を提供した(そして、静かに行う)、 第三者アプリケーションがあります。 最も一般的な方法は » PHPLIB の使用です。

関数の互換性は?

PHP 4 は基本的に PHP のエンジン部分を完全に書き直してものであるため、 変更された関数は極わずかです。

PHP 4 から PHP 5 への移行

この章は、PHP 4 から PHP 5 への移行の助けになることでしょう。

PHP 4 から PHP 5 への移行

PHP 5 では多くの新機能を提供してますが、以前のバージョンの PHP との互換性を可能な限り保っています。しかし、いくつかの機能については互換性が失われています。

マニュアルの PHP 5 への移行に関する付録 の適切な箇所を参照してください。PHP 5 への移行に関するさらに 重要な情報が得られます。

MySQL は PHP 5 で動きますか？ 見当たらないようなのですが。

MySQL はサポートされています。 たったひとつこれまでと変わったところは、PHP 5 では MySQL のサポートが 標準では 有効になっていないということです。つまり、 configure オプションに --with-mysql が含まれていないので、 PHP をコンパイルする際に手動でこのオプションを設定する必要があります。 Windows ユーザは、php.ini を編集して php_mysql.dll DLL を有効にします。PHP 4 では このような DLL は存在しませんでした。この機能は PHP バイナリの中に 直接埋め込まれていたのです。

また、MySQL クライアントライブラリは PHP には同梱されていません。 詳細についてはこの FAQ に 記述されています。また MySQL のインストール方法については MySQL 関数 を参照してください。 configure 設定の例は --with-mysql=/usr のようになります。Windows ユーザは、 libmySQL.dll が システムに存在する必要があります。

PHP 5 ではまったく新しいオブジェクト指向モデルが採用されていると 聞きましたが、既存のオブジェクト指向なコードは動きますか？ 新しい オブジェクト指向モデルについての情報はどこで得られますか？

PHP 5 での主要な変更点は新しいオブジェクト指向モデルであり、それは Zend Engine 2.0 を用いています。 zend.ze1_compatibility_mode ディレクティブを設定することで、 Zend Engine 1.0 (PHP 4) との互換性を保つことができます。

新しいオブジェクト指向モデルについては、 クラスとオブジェクト と 新しい オブジェクトモデル の章に記述されています。

新しいオブジェクト指向モデル以外に、PHP 5 で何が変わったのですか？ また、PHP 5 に特化した PHP のマニュアルはありますか？

それ以外の変更点はほとんどありません。詳細は PHP 4 から PHP 5への移行 を参照して ください。PHP 5 に特化されたマニュアルはありません。というのも、PHP の 大半の部分はこれまでと同じだからです。

その他の質問

ここでは他のカテゴリに分類することができないような質問を扱います。

bz2 で圧縮された Windows マニュアルはどのように扱えばいいのですか？

bz2 ファイルを処理することができるアーカイバを持っていない場合、 RedHat からコマンドラインツールを » ダウンロードしてください (詳細は下記の情報を参照)。

コマンドラインツールを使いたくない場合は、» Stuffit Expander、 » UltimateZip、 » 7-Zip、 » Quick Zip のようなフリーのツールを使用することも可能です。 » WinRAR または » Power Archiver のようなツールを持っている場合、簡単に bz2 ファイルを解凍できます。 Total Commander(以前は Windows Commander)を使用している場合は、 bz2 用のプラグインを » Total Commander のサイトから無料で手に入れることが出来ます。

Redhat から入手可能な bzip2 コマンドラインツール:

Windows 2000 SP2 のユーザは バージョン 1.0.2 を、他の全ての Windows ユーザはバージョン 1.00 を使用してください。 ダウンロードしたら実行ファイルの名前を bzip2.exe へ変更してください。 そしてそれをパスの通ったディレクトリ、例えば C:\Windows(ドライブ名は貴方のシステムにあわせて適宜変更してください) へコピーしてください。

注意: lang は使用する言語を意味し、 pdf のように x は指定した形式を表します。 php_manual_lang.x.bz2 を解凍するには以下のようにします。

- コマンドプロンプトウィンドウを開きます
- ダウンロードした php_manual_lang.x.bz2 を保存したディレクトリに cd コマンドを使用して移動します。
- bzip2 -d php_manual_lang.x.bz2 を実行すると php_manual_lang.x というファイルが同じディレクトリに生成されます。

分割 HTML 版アーカイブ(php_manual_lang.tar.bz2) をダウンロードした場合でも手順は同じです。 唯一の違いは、bzip2 コマンドの実行後に php_manual_lang.tar が生成されることです。 tar 形式のファイルは » WinZip といったアーカイバで扱うことができます。

asort() の定義のように、引数の横に & が ついているのはどういう意味ですか?

その引数が 参照渡し であり、 ドキュメントに書かれているとおり、関数内でそれが変更される可能性が あることを示します。変数のみをこの方法で渡すことができ、関数を コールする際に & を渡す必要はありません（これは 廃止予定 の方法です）。

私はどのように register_globals を扱うべきですか?

register_globals とセキュリティの関連については、 Using register_globals を参照ください。

register_globals が On であることに依存するよりも、 スーパーグローバル の使用を推奨します。

register_globals が Off になっている共有ホスト上で 古いアプリケーションを動かす必要があり、アプリケーションがこの設定を On にすることを要求している場合、 あるいはこの機能が On になっているホスティングサーバ上でセキュリティの リスクを抑えたい場合など、PHP でこれらの設定を逆転させる必要が出てきます。 いちばんいい方法は PHP の設定を変更できないか確認してみることですが、 もしそれができない場合には、このような方法を使用することができます。

Example#1 Register Globals をエミュレートする

これは register_globals On をエミュレートします。 variables_order ディレクティブを変更している場合は、それに応じて $superglobals を変更してください。

```
<?php
// register_globals on をエミュレートする
if (!ini_get('register_globals')) {
    $superglobals = array($_SERVER, $_ENV,
        $_FILES, $_COOKIE, $_POST, $_GET);
    if (isset($_SESSION)) {
        array_unshift($superglobals, $_SESSION);
    }
    foreach ($superglobals as $superglobal) {
        extract($superglobal, EXTR_SKIP);
    }
}
?>
```

これは register_globals Off をエミュレートします。 注意してほしいのは、このコードはスクリプト内のいちばん最初に コールされる必要があるということです。ただし、 セッションを開始するために session_start() を使用する場合は、その後にコールする必要があります。

```
<?php
// register_globals off をエミュレートする
function unregister_GLOBALS()
{
    if (!ini_get('register_globals')) {
        return;
    }

    // よりよいエラーメッセージのために、これを変更するとよいでしょう
    if (isset($_REQUEST['GLOBALS']) || isset($_FILES['GLOBALS'])) {
        die('GLOBALS overwrite attempt detected');
    }

    // unset すべきでない変数群
    $noUnset = array('GLOBALS',  '_GET',
                     '_POST',    '_COOKIE',
                     '_REQUEST', '_SERVER',
                     '_ENV',     '_FILES');

    $input = array_merge($_GET,    $_POST,
                         $_COOKIE, $_SERVER,
                         $_ENV,    $_FILES,
                         isset($_SESSION) && is_array($_SESSION) ? $_SESSION : array());

    foreach ($input as $k => $v) {
        if (!in_array($k, $noUnset) && isset($GLOBALS[$k])) {
            unset($GLOBALS[$k]);
        }
    }
}

unregister_GLOBALS();

?>
```

付録

目次

PHP の歴史と関連するプロジェクト

目次

PHP は最近数年で大きな進歩を遂げました。強力なサーバサイド の言語として最も有名なものの一つになる、ということは簡単な ことではありませんでした。PHP がどのように今日に到ったかに 興味がある人は読み進めてみてください。古い PHP のリリースについては » PHP Museum をご覧ください。

PHPの歴史

PHP/FI

PHPはPHP/FIというソフトウェアを継承したものです。PHP/FI は1995年にRasmus Lerdorfによって作成されました。当初は オンラインに置いてある彼のレジュメへのアクセスを解析するための Perlスクリプトの単純な組み合わせでした。彼はこのスクリプト に 'Personal Home Page Tools' という名前を付けました。 さらに多くの機能が要求されるようになると、Rasmusはデータ ベースとの連携や、簡単な動的ウェブアプリケーションを作成 できるようなものをC言語で書き直しました。RasmusはPHP/FIの ソースコードを皆が見られるように» 公開 する、という選択を したため、誰もがこれを使い、またバグを直したり改良したり することが出来ました。

PHP/FI(Personal Home Page / Forms Interpreter)には現在のPHPの 基本的な機能となっているものが含まれていました。Perlライクな 変数、フォームの値を自動で解釈する機能、そしてHTMLに埋め込まれた 文法です。文法そのものはPerlのそれに類似していますが、それよりは 制限が多く単純でいくらかの矛盾を抱えたものでした。

1997年までに、再度Cで書き直されたPHP/FI 2.0は(おそらく)世界で 数千の熱狂的なユーザを持ち、Internet上の1%程度となるおよそ50,000の ドメインにインストールされていたと報告されています。数人が ちょっとしたコードを提供していたとはいえ、それはまだ大きな 一人のプロジェクトでした。

PHP/FI 2.0は、その期間の多くをβ版として過ごした後、1997年の秋に 公式にリリースされました。その後すぐにPHP 3.0のα版が登場しました。

PHP 3

PHP 3.0は今日私たちが知っているPHPに非常に近いものとなった 最初のバージョンです。これはAndi GutmansとZeev Suraskiが PHP/FI 2.0でeコマースのアプリケーションを開発しようとしたときに その力不足に気づき、1997年に完全に書き直されて作成されました。 PHP/FIの既存の資産を失うことなく共存しようという努力の結果、 Andi、RasmusそしてZeevはPHP 3.0をPHP/FI 2.0の公式な後継 バージョンとしてアナウンスし、PHP/FI 2.0の開発はほぼ完全に 中止されました。

PHP 3.0の最も強力な点は、その拡張性でした。エンドユーザに 対して、多くのデータベースへの安定したアクセス機構に加えて プロトコルやAPIを提供することでPHP 3.0の拡張機能は 多くの開発者を惹きつけ、開発に加わったり新しい拡張モジュール が提供されるようになりました。間違いなくこれがPHP 3.0の 非常におおきな成功の鍵でした。PHP 3.0の重要な他の機能としては オブジェクト指向な文法や、強力で一貫性のある文法が挙げられます。

この完全に新しい言語は、PHP/FI 2.0が持っていた個人使用に制限されている という含意を取り除いた新しい名前でリリースされました。それは シンプルな 'PHP' という名前で、PHP: Hypertext Preprocessor という 再帰的な頭字語となっています。

1998年の終わりには、インストールベースで数万のユーザがおり、 数十万のウェブサイトがPHPをインストールしたと報告していました。 ピーク時にはPHP 3.0は世界のウェブサーバのおよそ10%にインストール されていました。

9ヶ月の公開テストを経てPHP 3.0は1998年6月に公式にリリース されました。

PHP 4

PHP 3.0が公式にリリースされて間もない1998年の冬、Andi Gutmans とZeev SuraskiはPHPの核となる部分を書き直し始めました。この 目的は、複雑なアプリケーションにおけるパフォーマンスの改善と PHPコードのモジュールとしての独立性を高めることでした。 そういったアプリケーションはPHP 3.0の新機能や第三者による 多くのデータベースやAPIのサポートを使用することで開発可能 でしたが、PHP 3.0はそういった複雑なアプリケーションを効率的に 扱うようにはデザインされていませんでした。

'Zend Engine'と呼ばれる新しいエンジン(開発者であるZeevとAndi の名前の組み合わせ)は、彼らの目的を十分に果たすものでした。 そしてそれは1999年中ごろに初めて紹介されました。このエンジンを 使用し、いくつかの広範囲にわたる新機能を追加したPHP 4.0は PHP 3.0のおよそ2年後となる2000年5月にリリースされました。 PHP 4.0はパフォーマンスが大幅に改善されたのに加え、 さらに多くのウェブサーバのサポート、HTTPセッション、 出力のバッファリング、ユーザの入力のさらに安全な取得方法の 提供、いくつかの新しい言語構造の提供といった特徴があります。

今日ではPHPには数十万の開発者がおり、数百万のサイトにインストール されています。これはInternetの20%以上のドメインにあたります。

PHPの開発チームには、数十人の開発者に加えて、PHPに関連するプロジェクト 、例えばPEARやドキュメンテーションに従事する別の数十人の開発者がいます。

PHP 5

PHP 5は長期にわたる開発とプレリリースを経て2004年7月に公開されました。 改良の目玉は、コアであるZend Engine 2.0で新しいオブジェクトモデルが サポートされたことで、その他にも多くの新しい機能が追加されています。

PHP関連のプロジェクトの歴史

PEAR

» PEAR、 PHP Extension and Application Repository(元は PHP Extension and Add-on Repositoryでした)は、 PHPで書かれた基本的なクラスの集まりで、将来的には PHPの拡張モジュールを開発者に配布する中心的な 方法になっていくでしょう。

PEARは2000年1月にTel Avivで開かれたPHP Developers' Meeting (PDM)で行われた議論の中から生まれたものです。これはStig S. Bakkenによって作成され、彼の初めて娘であるMalin Bakken に捧げられています。

2000年はじめの頃から、PEARは多くの開発者が一般的で再利用可能な 機能をPHPコミュニティ全体のために実装しました。これにより、 PEARは大きな、そして重要なプロジェクトへと成長しました。 今日PEARにはデータベースへのアクセス、キャッシュ、数学計算、 eCommerce等のバラエティに富んだクラスが含まれるようになって います。

PEARに関する詳細については、 » PEARのマニュアルを 参照してください。

PHP Quality Assurance Initiative(PHPの品質保証機関)

» PHP Quality Assurance Initiativeは、PHPのリリースが製品として様々な環境で 動作するべく十分にテストされていないという批判への対応策 として2000年の夏に開始されました。品質保証チームは現在のところ、PHPの ソースコードについて十分に理解している開発の中心となって いる人々によって構成されています。こうした開発者はPHPの ローカライズやバグフィクスに多くの時間を費やしています。 この他に、開発者の行った修正を個々の環境でテストして フィードバックを提供する多くのメンバーがいます。

PHP-GTK

» PHP-GTK は、 クライアントとして動作するGUIアプリケーションを構築するための PHPソリューションです。PHP-GTKの設計と実装は主として Andrei Zmievskiによります:

> GUIプログラミングにはずっと興味を持ちつづけていて、あるとき Gtk+というものが、Cでコーディングするにはあまりに面倒なものだということを除けば非常にすばらしいものだと言うことを知りました。 PyGtkやGTK-Perlの実装を見て、PHPで(必要最小限の部分だけでも) Gtkインターフェースを作成できないものだろうか、と考え始めたのです。 そして2000年の8月にちょっとした時間が出来たので実験を始めました。 私が主として参考にしたのはPyGtkの実装でした。というのもPyGtkは ほぼ全ての機能を網羅していて、且つオブジェクト指向のすばらしい インターフェースを持っていたからです。こうした最初の段階で 私はPyGtkの作者であるJmaes Henstridgeの有用なアドバイスを得ることが 出来ました。
>
> Gtk+関数へのインターフェースを全て自作するのは問題外だったので コードジェネレータというアイデアに飛びつきました。これは PyGtkと 同様の方法です。コードジェネレータとはGtk+のクラス、定数、そして メソッドが記述されている.defsファイルを読み込み PHPへのインターフェース となるCのコードを生成するPHPのプログラムです。自動的に生成することが 出来ない部分に関しては.overridesファイルに手動で追加していきました。
>
> 2000年の秋の間はあまり時間が取れなかったため、コードジェネレータ やインフラの構築に長い期間がかかってしまいました。しかしPHP-GTKをFrank kromannに見せると、彼は大いに興味を示して コードジェネレータとWin32版の実装を手伝い始めてくれました。 最初のプログラムHello Worldを書いて、それが動作したとき 私たちはとても興奮しました。このプロジェクトが公開するに 足るものになるにはさらに数ヶ月が必要でしたが2001年3月1日に 最初のバージョンをリリースすることが出来ました。このニュースは すぐにSlashDotにも載りました。
>
> PHP-GTKはさらに拡張していくだろうと考え、私は専用のメーリング リストとCVSリポジトリ、そしてCollin Viebrockの助けを得てウェブサイト(gtk.php.net)を用意しました。ドキュメンテーションは 目下進行中でJames Mooreが手伝ってくれています。
>
> PHP-GTKはリリース以来徐々に広がっています。ドキュメンテーション チームもは日々マニュアルを更新していますし、PHP-GTK用の拡張 モジュール、そしてPHP-GTKを使った興味深いアプリケーションを 作成する人も増えてきています。

PHP関連の本

PHPが成長していくに従って、開発プラットフォームとして世界中で 有名になっていきました。この流れを確かめる面白い方法のうちの 一つは、初期の頃から現在に到るまで頻繁に出版されつづけている PHPに関する本を観察してみることです。

私たちの知る限り、PHPを扱った最初の本は1999年の4月にチェコで 出版された 'PHP - tvorba interaktivních internetových aplikací'で Jirka Kosekにより書かれたものです。 翌月、Egon Schmid, Christian Cartus, Richard Blumeにおり 書かれたドイツ語の本が続いて出版されました。 PHPについて書かれた英語の本、 Leon Atkinsonによる'Core PHP Programming'がそのすぐ後に出版され ました。これらの本は、いずれも PHP 3.0を扱っていました。 (訳註:日本語では、PHPをタイトルとする本「PHP徹底後略」(堀田、石井、廣川)の 初版が1999年9月に出版されています。)

こうした最初の出版物に続いて、多くの著者、出版社から数多くの 本が出版されました。その数は英語で40冊以上、ドイツ語で50冊以上、 フランス語20冊以上にもなります!さらに他の言語、例えばスペイン語、 韓国語、日本語そしてヘブライ語でも本が出版されています。

このようにPHPに関する書籍が、異なった著者により執筆され、 多くの出版社から出版され、多くの言語で書かれている、という 事実はPHPが世界的に成功している強い証拠であると言えるでしょう。

PHP関連の出版物

私たちの知る限り、PHPに関する記事が最初に掲載された雑誌は 1998年の春に出版されたthe Czech mutation of Computerworldで、 PHP 3.0が扱われていました。本と同様に、これに続いて様々な すばらしい雑誌にPHPに関する記事が掲載されています。

PHPに関する記事はDr. Dobbs, Linux Enterprise, Linux Magazine, そのた多くの雑誌に掲載されています。ASPベースのアプリケーションをWindows上で動作するPHPに移植する記事が、なんと MicrosoftのMSDNに 掲載されています!

PHP 5.1.x から PHP 5.2.x への移行

目次

PHP 5.2.x における変更点

PHP 5.2.x で改良された点のほとんどは、既存のコードには影響を及ぼしません。 ただ、互換性を失う変更 や 新しいエラーメッセージ もあるので、実運用環境の PHP のバージョンをあげる際には事前にテストするようにしましょう。

もし PHP 5.0.x からアップグレードするのなら、このマニュアルの PHP 5.1.x への移行時の注意 も読んでおく必要があります。

同様に、もし PHP 4 からアップグレードするのなら、このマニュアルの PHP 4 から PHP 5 への移行 も読んでおく必要があります。

下位互換性のない変更点

既存の PHP 5 のコードのほとんどは変更なしで動作するはずですが、 以下の下位互換性のない変更点については注意しましょう。

- getrusage() に互換性のない引数を渡した場合に、 PHP 5.2.1 以降では **NULL** を返すようになりました。
- **ZipArchive::setCommentName()** は、PHP 5.2.1 以降は成功した場合に **TRUE** を返すようになりました。
- ZipArchive::setCommentIndex() は、PHP 5.2.1 以降は成功した場合に **TRUE** を返すようになりました。
- **SplFileObject::getFilename()** は、PHP 5.2.1 以降ではファイルへの相対パスではなくファイル名のみを返すようになりました。
- Win32 における環境変数 **PHPRC** の優先順位が変わりました。 環境変数 PHPRC は、Windows レジストリに保存されているパスより優先されるようになりました。
- CLI SAPI は、カレントディレクトリにある php.ini や php-cli.ini を見に行かないようになりました。 明示されていませんでしたが、PHP 5.1.x では CLI バイナリが現在の作業ディレクトリにある PHP 設定ファイルを参照するようになっていました。 これは、予期せぬ設定ファイルを読み込んでしまうという問題を起こす可能性がありました。 この機能は PHP 5.2.0 で削除されました。PHP は、カレントディレクトリの php.ini や php-cli.ini を参照しません。 マニュアルの コマンドライン の部分も参照ください。
- ゼロで割った余りを求めようとすると警告が発生するようになりました。 以前のバージョンの PHP では、integer % 0 は何も警告を発せず、予期せぬ結果である **FALSE** を返していました。 PHP 5.2.0 以降では、このような操作をしようとすると **E_WARNING** が発生するようになりました。これは、ゼロで除算を行った際の挙動と同じです。

```
<?php
print 10 % 0;
/* Warning:  Division by zero in filename on line n */
?>
```

- **__toString()** が、適切な場面で常にコールされるようになりました。 マジックメソッド **__toString()** は、文字列コンテキストでコールされるようになりました。 つまり、オブジェクトを文字列として使用する際には常にコールされるということです。 オブジェクトを文字列として扱った場合に自動的にオブジェクト ID が返されるという機能は、PHP 5.2.0 で廃止されました。 オブジェクト ID は常に一意となるわけではないので、この機能には問題があったわけです。 この変更により、オブジェクト ID が返されることを前提としたアプリケーションは動作がおかしくなってしまいます。 オブジェクトの値を文字列として使用すると、(捕捉可能な) 致命的なエラーとなります。

```
<?php
class foo {}
$foo = new foo;
print $foo;
/* Catchable fatal error:  Object of class foo could
   not be converted to string in filename on line n */
?>
```

たとえ **__toString()** を実装したとしても、 オブジェクトを配列のインデックスやキーとして使用することはできません。 将来的には組み込みの機能でハッシュをサポートする予定ですが、 現時点の PHP 5.2.x の段階では、自分でハッシュ処理を実装するか SPL の関数 spl_object_hash() を使用してください。 **__toString()** メソッド内からは例外をスローすることはできません。

```
<?php
class foo {
     public function __toString() {
         throw new Exception;
     }
}

try {
     print new foo;
     /* Fatal error:  Method foo::__toString() must
          not throw an exception in filename on line n */
} catch(Exception $e) {}
?>
```

- abstract static なクラス関数が削除されました。 ちょっとした手違いで、PHP 5.0.x および 5.1.x では abstract static な関数をクラス内で定義できてしまっていました。PHP 5.2.x では、 これはインターフェイス内でのみ定義できるようになりました。

```
<?php
abstract class foo {
     abstract static function bar();
     /* Strict Standards:  Static function foo::bar()
          should not be abstract in filename on line n */
}
?>
```

- Oracle 拡張モジュール を Windows で使用するには、最低でも Oracle のバージョン 10 が必要となりました。
- RFC2397 (data: ストリーム) に対応しました。 'data' URL スキームに対応したことにより、 Windows 環境での挙動が変わる可能性があ

ります。 NTFS ファイルシステムを使用しており、 アプリケーションでメタストリームを使用していた場合、 もし 'data:' という名前のファイルをパス情報なしでアクセスしようとすると正しく動作しません。 これを避けるには、アクセス時に 'file:' プロトコルを使用します。 » RFC 2397 も参照ください。

```
<?php
/* allow_url_include が OFF (デフォルト) の場合 */
include "data:;base64,PD9waHAgcGhwaW5mbygpOz8+";
/* Warning:  include(): URL file-access is disabled
    in the server configuration in filename on line n */
?>
```

- glob() パターンの退化 バージョン 5.2.4 でのセキュリティ修正の副作用で、 "/foo/*/bar/*" 形式のパターンがうまく動作しなくなってしまっていました。 バージョン 5.2.5 以降では、glob() が openbase_dir 制約に違反した場合に警告を発生させず false を返すようになります。

新しいエラーメッセージ

以下に示す新しいエラーメッセージは、まだこのマニュアルのどこにも説明されていないものです。

Example#1 PHP コア

```
<?php
echo " ";
session_regenerate_id();
/*   Warning:  session_regenerate_id(): Cannot regenerate
     session id - headers already sent in filename on line n */

str_word_count("string", 4);
/* Warning:  str_word_count(): Invalid format value 4
    in filename on line n */

strripos("foo", "f", 4);
/* Notice:  strripos(): Offset is greater than the
    length of haystack string in filename on line n */

strrpos("foo", "f", 4);
/* Notice:  strrpos(): Offset is greater than the
    length of haystack string in filename on line n */

/* As of PHP 5.2.1, when allow_url_include is OFF (default) */
include "php://input";
/* Warning:  include(): URL file-access is disabled
    in the server configuration in filename on line n */
?>
```

Example#2 PHP コアにおける オブジェクト指向のコード

```
<?php
interface foo {
}
class bar implements foo, foo {
}
/* Fatal error: Class bar cannot implement previously
    implemented interface foo in filename on line n */


class foo {
     public $bar;
     function __get($var)
     {
          return $this->bar;
     }
}

$foo = new foo;
$bar =& $foo->prop;
/* Notice: Indirect modification of overloaded property
    foo::$prop has no effect in filename on line n */


class foo implements iterator {
     public function current() {
     }
     public function next() {
     }
     public function key() {
     }
     public function valid() {
     }
     public function rewind() {
     }
}

$foo = new foo();
foreach($foo as &$ref) {}
/* Fatal error: An iterator cannot be used with foreach
    by reference in filename on line n */


class foo {
     private function __construct() {
     }
}
class bar extends foo {
     public function __construct() {
        parent::__construct();
        /* Fatal error:  Cannot call private
             foo::__construct() in filename on line n */
```

```
    }
}
new bar;


stream_filter_register("", "class");
/* Warning:   stream_filter_register(): Filter  name
    cannot be empty in filename on line n */


stream_filter_register("filter", "");
/* Warning:   stream_filter_register(): Class  name
    cannot be empty in filename on line n */
```

Example#3 bzip2 拡張モジュール

```
<?php
bzopen("", "w");
/* Warning:   bzopen(): filename cannot be empty
    in filename on line n */

bzopen("foo", "a");
/* Warning:   bzopen(): 'a' is not a valid mode for
    bzopen(). Only 'w' and 'r' are supported in
    filename on line n */

$fp = fopen("foo", "w");
bzopen($fp, "r");
/* Warning:   bzopen(): cannot read from a stream
    opened in write only mode in filename on line n */
?>
```

Example#4 datetime 拡張モジュール

```
<?php
strtotime("today", "now");
/* Warning:   strtotime() expects parameter 2 to be
    long, string given in filename on line n */

/* PHP 5.2.1 以降 */
new DateTime(new stdclass);
/* Fatal error: Uncaught exception 'Exception' with
     message 'DateTime::__construct() expects parameter
    1 to be string, object given' in filename:n */
?>
```

Example#5 dBase 拡張モジュール

```
<?php
dbase_open("foo", -1);
/* Warning: Invalid access mode -1 in filename on line n */

/* PHP 5.2.1 以降 */
dbase_open("foo", null);
/* Warning: The filename cannot be empty in filename on line n */
?>
```

Example#6 mcrypt 拡張モジュール

```
<?php
$key = "this is a secret key";
$td = mcrypt_module_open('tripledes', '', 'ecb', '');
$iv = mcrypt_create_iv  (mcrypt_enc_get_iv_size($td),
                         MCRYPT_RAND);
mcrypt_generic_init($td, $key, $iv);
$encrypted_data  = mcrypt_generic($td, "");
/* Warning: mcrypt_generic(): An empty string was
    passed in filename on line n */
?>
```

Example#7 oci8 拡張モジュール

```
<?php
oci_connect("user", "pass", "db", "bogus_charset");
/* Warning: Invalid character set name:
    bogus_charset in filename on line n */

$oci = oci_connect("user", "pass", "db");
oci_password_change($oci, "", "old", "new");
/* Warning: username cannot be empty in filename
    on line n */

oci_password_change($oci, "user", "", "new");
/* Warning: old password cannot be empty in filename
    on line n */

oci_password_change($oci, "user", "old", "");
/* Warning: new password cannot be empty in filename
    on line n */
?>
```

Example#8 SPL 拡張モジュール

```
<?php
$obj = new SplFileObject(__FILE__);
$obj->fgetcsv("foo");
/* Warning:   SplFileObject::fgetcsv(): delimiter must
    be a character in filename on line n */

$obj->fgetcsv(",", "foo");
```

```
/* Warning:  SplFileObject::fgetcsv(): enclosure must
    be a character in filename on line n */
?>
```

Example#9 Semaphore (sysvmsg) 拡張モジュール

```
<?php
/* Warning:  maximum size of the message has to be
    greater then zero in filename on line n */
?>
```

Example#10 5.2.1 以降の Zip の例

```
<?php
$obj = new ZipArchive();
$obj->open('archive.zip');
$obj->setCommentName('', 'comment');
/* Notice:  ZipArchive::setCommentName(): Empty string
    as entry name in filename on line n */

/* PHP 5.2.1 以降 */
$obj->getCommentName('');
/* Notice:  ZipArchive::getCommentName(): Empty string
    as entry name in filename on line n */
?>
```

PHP での datetime サポートの変更点

PHP 5.1.0 から、date 拡張モジュールが PHP コアに組み込まれるようになりました。 これは、PHP の日付時刻サポートを新たに実装しなおしたものです。 システムのタイムゾーン設定を自動的に取得しようとはしますが、 手動で設定しておいたほうがよいでしょう。以下の三通りのうち、 いずれかの方法で設定します。

- php.ini で、 date.timezone ディレクティブを設定する
- システムの環境変数 TZ を設定する
- スクリプト中で、関数 date_default_timezone_set() を使用する

サポートしている タイムゾーン の一覧は、PHP マニュアルに記載されています。

PHP 5.2.x では、日付やタイムゾーンをオブジェクトとして扱えるようになりました。 それぞれ DateTime および DateTimeZone という名前になります。 このオブジェクトのメソッドは、既存の手続き型の日付関数群に対応しています。

新しいパラメータ

PHP 5.2.x で、新しいパラメータやオプションのパラメータが追加された関数があります。

PHP コア:

- htmlentities() - PHP 5.2.3 で double_encode が追加されました。
- htmlspecialchars() - PHP 5.2.3 で double_encode が追加されました。
- base64_decode() - strict が追加されました。
- setcookie() - httponly が追加されました。
- setrawcookie() - httponly が追加されました。
- session_set_cookie_params() - httponly が追加されました。
- memory_get_usage() - real_usage が追加されました。
- get_loaded_extensions() - PHP 5.2.4 で zend_extensions が追加されました。

curl:

- curl_multi_info_read() - msgs_in_queue が追加されました。

datetime

- date() - PHP 5.2.2 で、書式文字 "u"（ミリ秒）が追加されました。

imap:

- imap_open() - n_retries が追加されました。
- imap_reopen() - n_retries が追加されました。

mbstring:

- mb_strrpos() - offset が追加されました。
  警告

  offset パラメータが、かつて encoding パラメータのあった場所に入りました。 下位互換性のために encoding を三番目のパラメータとして指定することもできます。 しかし、この下位互換性モードの使用はお勧めしません。 これは将来は廃止される予定だからです。

ming:

- **swfmovie::streamMP3()** - skip が PHP 5.2.1 で追加されました。

openssl:

- openssl_verify() - signature_algo が追加されました。

pgsql:

- pg_escape_bytea() - connection が追加されました。
- pg_escape_string() - connection が追加されました。

simplexml:

- **SimpleXMLElement::__construct()** - is_prefix が追加されました。
- **SimpleXMLElement::attributes()** - is_prefix が追加されました。
- **SimpleXMLElement::children()** - is_prefix が追加されました。
- simplexml_load_file() - is_prefix が追加されました。
- simplexml_load_string() - is_prefix が追加されました。

spl:

- array iterator_to_array(Traversable it [, bool use_keys = true]) - use_keys が PHP 5.2.1 で追加されました。

xmlreader:

- XMLReader::open() - encoding および options が追加されました。
- XMLReader::XML() - encoding および options が追加されました。

XMLWriter:

- **XMLWriter::WriteElement()** - PHP 5.2.3 で content はオプションとなりました。
- **XMLWriter::WriteElementNs()** - PHP 5.2.3 で content はオプションとなりました。

新しい関数

PHP 5.2.x で新しく追加された関数は次のとおりです。

PHP コア:

- array_fill_keys() - 最初のパラメータで指定した要素をキーとして、値を初期化した配列を作成します。
- error_get_last() - 直近に発生したエラーを連想配列として取得します。 エラーが発生していない場合は **NULL** を返します。
- image_type_to_extension() - getimagesize、 exif_read_data、exif_thumbnail、exif_imagetype が返した画像形式に対応する拡張子を取得します。
- memory_get_peak_usage() - PHP のメモリ割り当ての最大値を返します。
- sys_get_temp_dir() - 一時ファイル用のディレクトリのパスを返します (PHP 5.2.1 で追加されました)。
- timezone_abbreviations_list() - 夏時間の情報や時差、タイムゾーン名を含む連想配列を返します。
- timezone_identifiers_list() - すべてのタイムゾーン ID を配列で返します。
- timezone_name_from_abbr() - タイムゾーンの略称を指定して、正式名を返します。
- stream_socket_shutdown() - ストリームに関連付けられたソケット上の全二重通信 (のすべてあるいは一部) をシャットダウンします。 PHP 5.2.1 以降で使用できます。

Image:

- imagegrabscreen() - 画面全体のスクリーンショットを取得します。PHP 5.2.2 以降で使用可能です。
- imagegrabwindow() - ウィンドウをキャプチャします。PHP 5.2.2 以降で使用可能です。

mbstring:

- **mb_list_encodings_alias_names()** - サポートするすべての実体エンコーディングを配列で返します。
- **mb_list_mime_names()** - サポートするすべての mime 名を配列あるいは文字列で返します。
- mb_stripos() - 大文字小文字を区別せず、ある文字列が別の文字列中に最初に現れる位置を探します。
- mb_stristr() - 大文字小文字を区別せず、ある文字列が別の文字列中に最初に現れる場所を探します。
- mb_strrchr() - ある文字が文字列中に最後に現れる場所を探します。
- mb_strrichr() - 大文字小文字を区別せず、ある文字が文字列中に最後に現れる場所を探します。
- mb_strripos() - 大文字小文字を区別せず、ある文字が文字列中に最後に現れる位置を探します。
- mb_strstr() - ある文字列が別の文字列中に最初に現れる場所を探します。

ming (PHP 5.2.1 以降):

- void ming_setSWFCompression(int num) - 出力の圧縮を設定します。
- void swfmovie::namedanchor(string name) - アンカーを作成します。
- void swfmovie::protect([string pasword]) - プロテクトします。

openssl:

- openssl_csr_get_public_key() - CERT から公開鍵を取り出し、使用できるようにします。
- openssl_csr_get_subject() - CERT の subject を返します。
- openssl_pkey_get_details() - 鍵の詳細 (bits、pkey、type) を配列で返します。

spl:

- spl_object_hash() - 指定したオブジェクトのハッシュ ID を返します。
- int iterator_apply(Traversable it, mixed function [, mixed params]) - イテレータのすべての要素に対して関数をコールします。

pcre:

- preg_last_error() - 直近に実行した正規表現のエラーコードを返します。

pgsql:

- pg_field_table() - フィールドが属するテーブルの名前、あるいは oid_only が true の場合にテーブルの oid を返します。

posix:

- posix_initgroups() - 名前で指定したユーザのグループアクセスリストを計算します。

gmp:

- gmp_nextprime() - 指定した数より大きい次の素数を探します。

xmlwriter:

- xmlwriter_full_end_element() - 現在の要素を終了します。エラー時に FALSE を返します。
- xmlwriter_write_raw() - テキストを書き込みます。エラー時に FALSE を返します。
- xmlwriter_start_dtd_entity() - DTD エンティティを開始します。エラー時に FALSE を返します。
- xmlwriter_end_dtd_entity() - 現在の DTD エンティティを終了します。エラー時に FALSE を返します。
- xmlwriter_write_dtd_entity() - 完全な DTD エンティティタグを書き込みます。エラー時に FALSE を返します。

新しいメソッド

5.2.x で新しく追加されたメソッドは次のとおりです。

dom:

- **DOMDocument::registerNodeClass()** - 基底ノード型を作成するために使用する拡張クラスを登録します。
- **DOMElement::setIDAttribute()** - ID 型の属性を名前で宣言します。
- **DOMElement::setIDAttributeNode()** - ID 型の属性をノードで宣言します。
- **DOMElement::setIDAttributeNS()** - ID 型の属性をローカル名および名前空間 URI で宣言します。
- DOMNode::C14N([bool exclusive [, bool with_comments [, array xpath [, array ns_prefixes]]]]) - ノードを文字列に正規化します。
- DOMNode::C14NFile(string uri [, bool exclusive [, bool with_comments [, array xpath [, array ns_prefixes]]]]) - ノードをファイルに正規化します。
- DOMNode::getNodePath() - ノードの xpath を取得します。

soap:

- SoapServer::setObject(object obj) - SOAP リクエストを処理するオブジェクトを設定します。

spl:

- int ArrayObject::asort(void) - エントリを値で並べ替えます。
- int ArrayObject::ksort(void) - エントリをキーで並べ替えます。
- int ArrayObject::natcasesort(void) - エントリをキーで並べ替えます。大文字小文字を区別しない "自然順" アルゴリズムを使用します。
- int ArrayObject::natsort(void) - エントリを値で並べ替えます。"自然順" アルゴリズムを使用します。
- int ArrayObject::uasort(callback cmp_function) - エントリを、ユーザ定義関数を使用して値で並べ替えます。
- int ArrayObject::uksort(callback cmp_function) - エントリを、ユーザ定義関数を使用してキーで並べ替えます。
- ArrayIterator AppendIterator::getArrayIterator() - 内部の ArrayIterator にアクセスします。
- int AppendIterator::getIteratorIndex() - イテレータのインデックスを取得します。
- bool CachingIterator::getCache() - キャッシュを返します。
- int CachingIterator::getFlags() - 内部フラグを返します。
- bool CachingIterator::offsetExists(mixed index) - 指定したインデックスが存在するかどうかを返します。
- string CachingIterator::offsetGet(mixed index) - 内部キャッシュを使用している場合にそれを返します。
- void CachingIterator::offsetSet(mixed index, mixed newval) - キャッシュ内の指定したインデックスを設定します。
- void CachingIterator::offsetUnset(mixed index) - キャッシュ内の指定したインデックスを削除します。
- void CachingIterator::setFlags() - 内部フラグを設定します。
- array("delimiter" =>, "enclosure" =>) SplFileObject::getCsvControl(void) - fgetcsv で使用する区切り文字および囲み文字を取得します。
- void SplFileObject::setCsvControl([string delimiter = ',' [, string enclosure = '"']]) - fgetcsv で使用する区切り文字および囲み文字を設定します。

Tidy

- tidyNode tidyNode::getParent(void) - 現在のノードの親ノードを返します (PHP 5.2.2 で追加されました)。

XMLReader

- boolean XMLReader::setSchema(string filename) - W3C XSD スキーマを使用して、処理するドキュメントを検証します。 最初の Read() を行う前にのみ実行可能です。

zip:

- ZipArchive::addEmptyDir() - 空のディレクトリをアーカイブ内に作成します。

削除された拡張モジュール

これらの拡張モジュールは PECL に移されました。PHP の配布ファイルには含まれません。 これらの拡張モジュールの PECL 版は、必要に応じてユーザが各自で作成します。

- filePro
- Hyperwave API

新しい拡張モジュール

これらの拡張モジュールは、PHP 5.2.0 以降で (デフォルトで) 新しく追加されました。

- Filter - データの検証とフィルタリングを行います。ユーザの入力のように、 安全でないデータを扱うときに使用することを考慮した設計になっています。 この拡張モジュールはデフォルトで有効になります。 デフォルトのモードである RAW は、入力データに一切手を加えません。
- JSON - JavaScript Object Notation (JSON) データ交換フォーマットを実装します。 この拡張モジュールはデフォルトで有効になります。
- Zip - ZIP 圧縮されたアーカイブおよびその中のファイルを 透過的に読み書きすることができます。

新しいクラス

以下のクラスが PHP 5.2.0 で追加されました。

- DateTime
- DateTimeZone
- RegexIterator - extends FilterIterator; implements Iterator, Traversable, OuterIterator 定数
  - **RegexIterator::ALL_MATCHES**
  - **RegexIterator::GET_MATCH**
  - **RegexIterator::MATCH**
  - **RegexIterator::REPLACE**
  - **RegexIterator::SPLIT**
  - **RegexIterator::USE_KEY**

  プロパティ
  - public replacement

  メソッド
  - RegexIterator::__construct(Iterator it, string regex [, int mode [, int flags [, int preg_flags]]]) - 別のイテレータおよび正規表現をもとにして RegexIterator を作成します。
  - bool RegexIterator::accept() - (string)current() を正規表現とマッチさせます。
  - bool RegexIterator::getFlags() - 現在の操作フラグを返します。
  - bool RegexIterator::getMode() - 現在の操作モードを返します。
  - bool RegexIterator::getPregFlags() - 現在の PREG フラグ (使用中の場合。それ以外は **NULL**) を返します。
  - bool RegexIterator::setFlags(int new_flags) - 操作フラグを設定します。
  - bool RegexIterator::setMode(int new_mode) - 操作モードを設定します。
  - bool RegexIterator::setPregFlags(int new_flags) - PREG フラグを設定します。
- RecursiveRegexIterator 定数
  - **RecursiveRegexIterator::ALL_MATCHES**
  - **RecursiveRegexIterator::GET_MATCH**
  - **RecursiveRegexIterator::MATCH**
  - **RecursiveRegexIterator::REPLACE**
  - **RecursiveRegexIterator::SPLIT**
  - **RecursiveRegexIterator::USE_KEY**

  メソッド
  - RecursiveRegexIterator::__construct(RecursiveIterator it, string regex [, int mode [, int flags [, int preg_flags]]]) - 別の再帰イテレータおよび正規表現をもとにして RecursiveRegexIterator を作成します。
  - RecursiveRegexIterator RecursiveRegexIterator::getChildren() - RecursiveRegexIterator に含まれる内部イテレータの子を返します。

- *bool RecursiveRegexIterator::hasChildren() - 内部イテレータの現在の要素が子を持つかどうかを調べます。*

新しいグローバル定数

PHP コア:

- ***M_EULER***
- ***M_LNPI***
- ***M_SQRT3***
- ***M_SQRTPI***
- ***PATHINFO_FILENAME***
- ***PREG_BACKTRACK_LIMIT_ERROR***
- ***PREG_BAD_UTF8_ERROR***
- ***PREG_INTERNAL_ERROR***
- ***PREG_NO_ERROR***
- ***PREG_RECURSION_LIMIT_ERROR***
- ***UPLOAD_ERR_EXTENSION***
- ***STREAM_SHUT_RD***
- ***STREAM_SHUT_WR***
- ***STREAM_SHUT_RDWR***

curl:

- ***CURLE_FILESIZE_EXCEEDED***
- ***CURLE_FTP_SSL_FAILED***
- ***CURLE_LDAP_INVALID_URL***
- ***CURLFTPAUTH_DEFAULT***
- ***CURLFTPAUTH_SSL***
- ***CURLFTPAUTH_TLS***
- ***CURLFTPSSL_ALL***
- ***CURLFTPSSL_CONTROL***
- ***CURLFTPSSL_NONE***
- ***CURLFTPSSL_TRY***
- ***CURLOPT_FTP_SSL***
- ***CURLOPT_FTPSSLAUTH***
- ***CURLOPT_TCP_NODELAY*** *PHP 5.2.1 で追加されました。*
- ***CURLOPT_TIMEOUT_MS*** *PHP 5.2.3 で追加されました。*
- ***CURLOPT_CONNECTTIMEOUT_MS*** *PHP 5.2.3 で追加されました。*

GMP:

- ***GMP_VERSION*** *PHP 5.2.2 で追加されました。*

ming:

- ***SWFTEXTFIELD_USEFONT***
- ***SWFTEXTFIELD_AUTOSIZE***
- ***SWF_SOUND_NOT_COMPRESSED***
- ***SWF_SOUND_ADPCM_COMPRESSED***
- ***SWF_SOUND_MP3_COMPRESSED***
- ***SWF_SOUND_NOT_COMPRESSED_LE***
- ***SWF_SOUND_NELLY_COMPRESSED***
- ***SWF_SOUND_5KHZ***
- ***SWF_SOUND_11KHZ***
- ***SWF_SOUND_22KHZ***
- ***SWF_SOUND_44KHZ***
- ***SWF_SOUND_8BITS***
- ***SWF_SOUND_16BITS***
- ***SWF_SOUND_MONO***
- ***SWF_SOUND_STEREO***

openssl:

- ***OPENSSL_VERSION_NUMBER***
- ***OPENSSL_VERSION_TEXT***

snmp:

- ***SNMP_OID_OUTPUT_FULL***

- **SNMP_OID_OUTPUT_NUMERIC**

Semaphore:

- **MSG_EAGAIN**
- **MSG_ENOMSG**

新しいクラス定数

pdo:

- **PDO::ATTR_DEFAULT_FETCH_MODE**
- **PDO::FETCH_PROPS_LATE**
- **PDO::FETCH_KEY_PAIR** 2 つのカラムからなる結果セットを連想配列に格納します (PHP 5.2.3 で追加されました)。

spl:

- **CachingIterator::FULL_CACHE**
- **CachingIterator::TOSTRING_USE_INNER**
- **SplFileObject::READ_AHEAD**
- **SplFileObject::READ_CSV**
- **SplFileObject::SKIP_EMPTY**

新しい INI 設定項目

PHP 5.2.0 で、php.ini に新しい項目が追加されました。

- allow_url_include この便利なオプションを使用すると、 リモートファイルに対する通常のファイル操作と リモートファイルのインクルードを別々に制御することができるようになります。 前者はよくある要求ですが、 後者については何も考えずに使用するとセキュリティリスクとなる可能性があります。 PHP 5.2.0 以降では、リモートファイルの操作はできるけれども リモートファイルをローカルスクリプトにインクルードすることはできない という設定が可能となりました。実際のところ、これがデフォルトの設定となっています。
- pcre.backtrack_limit PCRE におけるバックトラックの制限値。
- pcre.recursion_limit PCRE の再帰処理の制限値。この値をあまり大きくしすぎると、 プロセススタックを食いつぶして (OS のスタックサイズの制限に到達してしまって) PHP がクラッシュすることがあるので注意しましょう。
- session.cookie_httponly クッキーを、HTTP プロトコルでのみアクセスできるようにします。 つまり、JavaScript のようなスクリプト言語からはアクセスできなくなるということです。 この設定を使用すると、XSS 攻撃によって ID を盗まれる可能性が少なくなります (とはいえ、ブラウザによってはこれをサポートしていないものもあります)。

PHP 5.2.2 で新しく追加された項目です。

- max_input_nesting_level 外部から受け取る入力変数 のネストの深さを制限します。デフォルトは 64 です。

エラー処理

これまで **E_ERROR** となっていた内容のうちのいくつかについて、 ユーザ定義のエラーハンドラで処理できるようになりました。これは **E_RECOVERABLE_ERROR** で表され、ハンドラを作成しなかった場合は、 **E_ERROR** と同様に振る舞います。 この型のエラーは、ログには Catchable fatal error と記録されます。

この変更によって、 error_reporting の定数 **E_ALL** の値は 6143 となりました。 以前は、この値は 2047 でした。PHP の定数は PHP の外部では使用できないので、 外部でこの整数値を指定していることもあるでしょう。その場合は値を変更する必要があります。 たとえば error_reporting の設定を httpd.conf あるいは .htaccess で行っている場合は、値を適切に変更する必要があります。 また、PHP スクリプト内で定数ではなく値自体を指定している場合も同様です。

この変更の副作用として、 track_errors が On の場合にエラーメッセージが重複してしまうことがあるようになりました。 これを防ぐため、エラーハンドラで $php_errormsg を設定した際には **FALSE** を返さなければならないようになりました。 これにより、さまざまなレベルのメッセージをより精密に制御できるようになります。

その他の機能向上

- メモリ管理を改善しデフォルトのメモリ制限を増加しました。 新しいメモリマネージャは、以前のものに比べて より少ないメモリでより高速に動作するようになりました。 システムから取得したメモリをラージブロックに割り当て、 自分でヒープを管理します。 これまでのバージョンでは、emalloc() がコールされるたびに php.ini の memory_limit をチェックしていました。 現在は、システムから実際にメモリを取得する際にのみチェックしています。 つまり、memory_limit はこれまでに比べて大幅に精度が上がったということです。 これまでのメモリマネージャは、 malloc ライブラリが使用するメモリのオーバーヘッドを考慮していませんでした。 このように計算精度を向上させたおかげで、これまでより効率的にメモリを使用できるようになりました。 これに対応するため、デフォルトの memory_limit 設定も変更され、 8 メガバイトから 16 メガバイトとなりました。
- インターフェイスのコンストラクタで、 実装クラスのコンストラクタのシグネチャを強制できるようになりました。 PHP 5.2.0 以降で、インターフェイスにコンストラクタを定義できるようになりました。 しかし、インターフェイスでコンストラクタを宣言した場合は、 そのインターフェイスを実装するクラスは インターフェイスのコンストラクタと同じシグネチャのコンストラクタを持つ必要があります。 ここでいう「シグネチャ」とは、パラメータや返り値の定義のことです。 また、型ヒントや 参照渡し/値渡し の区別なども含まれます。

PHP 5.0.x から PHP 5.1.x への移行

目次

PHP 5.1.x における変更点

PHP 5.1.x では以下のような点が変更されています。

- 日付処理用のコードを完全に書き直し、タイムゾーンのサポートを改善しました。
- PHP 5.0.x と比較してパフォーマンスが大幅に向上しました。
- PDO 拡張モジュールがデフォルトで組み込まれるようになりました。
- さまざまな拡張モジュールや組み込みの機能で、30 を超える新しい関数が追加されました。
- 400 を超えるバグを修正しました。

参照に関する処理の変更

概要

PHP スクリプトの開発者としての観点から見ると、 既存のコードに与える影響がもっとも大きいのは、 PHP 4.4.0 以降での参照に関する扱いの変更です。

PHP 4.3 を含むそれ以前のバージョンでは、本来は値を返すべき場所、 たとえば定数・一時的な値（例: 式の結果）・値を返す関数の結果 などで参照を使用することが可能でした。以下がその例です。

```
<?php
$foo = "123";

function return_value() {
     global $foo;
     return $foo;
}

$bar = &return_value();
?>
```

このコードは PHP 4.3 以下では通常通り動作しますが、一般的に結果は未定義となります。 Zend エンジンはこれらの値を参照として正しく扱えません。このバグにより、 再現不可能な問題が数多く発生していました。特に大規模なコードになるほどこの問題は深刻でした。

PHP 4.4.0、PHP 5.0.4 以降のリリースでは Zend エンジンが修正され、 参照を使用すべきでない場面で参照の操作が行われた場合はそれを「知る」 ことができるようになりました。今ではそのような場合には実際の値が使用されるようになり、 警告を発生するようになりました。この警告の形式は、 PHP 4.4.0 以降では **E_NOTICE**、 PHP 5.0.4 以降では **E_STRICT** となります。

これまでメモリ破壊を起こす可能性のあったコードは、もはやその心配はなくなりました。 しかし、既存のコードの中にはこれまでと挙動が変わってしまうものもあります。

PHP 4.3.x では動作するものの、現在は動作しないコード

```
<?php
function func(&$arraykey) {
    return $arraykey; //  関数は値を返します!
}

$array = array('a', 'b', 'c');
foreach (array_keys($array) as $key) {
    $y = &func($array[$key]);
    $z[] =& $y;
}

var_dump($z);
?>
<
```

参照に関する修正が行われる前の PHP で上のスクリプトを実行すると、このような結果になります。

```
array(3) {
  [0]=>
  &string(1) "a"
  [1]=>
  &string(1) "b"
  [2]=>
  &string(1) "c"
}
```

修正後は、同じコードの結果が次のようになります。

```
array(3) {
  [0]=>
  &string(1) "c"
  [1]=>
  &string(1) "c"
  [2]=>
  &string(1) "c"
}
```

理由は、今回の変更によって func() の結果が値として代入されるようになったことです。 $y の値は代入しなおされ、そして、$z では $y への参照が残り続けます。 この修正が行われる前は、func() の結果が参照として代入され、 先頭の $y は毎回代入されてしまっていました。 一時的な変数を参照で渡そうとすることが、メモリ破壊の原因となっていました。

修正前のバージョンの PHP と修正後のバージョンの PHP とで、 このようなコードの挙動を同じにすることは可能です。 func() の定義を変更して参照を返すようにするか、あるいは func() が返す結果を参照として代入する処理を取り除けばよいのです。

```php
<?php
function func() {
    return '関数の返す値';
}

$x = 'original value';
$y =& $x;
$y = &func();
echo $x;
?>
```

PHP 4.3 では $x は 'もとの値' となりますが、 この変更以降のバージョンでは '関数の返す値' となります。関数は参照を返さず、 値が直接代入されることを覚えておきましょう。もう一度いいます。 このような挙動の違いをなくすには、func() が参照を返すようにするか、 関数の返す値を直接参照として代入することをやめるかのどちらかを行います。

PHP 4.3.x では動作するものの、現在はエラーとなるコード

```php
<?php
class Foo {

     function getThis() {
         return $this;
     }

     function destroyThis() {
         $baz =& $this->getThis();
     }
}

$bar = new Foo();
$bar->destroyThis();
var_dump($bar);
?>
```

PHP 5.0.3 では、$bar はオブジェクトを返さずに **NULL** と評価されます。 なぜなら、getThis() が値を返しているにもかかわらず、 それを参照として代入しているからです。 現在ではこれは期待通りに動作しますが、実際のところこれは不正なコードです。 PHP 4.4 では **E_NOTICE**、 PHP 5.0.4 以降では **E_STRICT** をスローします。

PHP 4.3.x では動作しなかったが、現在は動作するコード

```php
<?php
function &f() {
    $x = "foo";
    var_dump($x);
    print "$x¥n";
    return($a);
}

for ($i = 0; $i < 3; $i++) {
    $h = &f();
}
?>
```

PHP 4.3 では var_dump() を 3 度目にコールした際に **NULL** を返し、 初期化されていない値への参照を返すことでメモリの破壊を引き起こしてしまいます。 このコードは PHP 5.0.4 以降では動作しますが、 それより前のバージョンの PHP ではエラーとなります。

```php
<?php
$arr = array('a1' => array('alfa' => 'ok'));
$arr =& $arr['a1'];
echo '-'.$arr['alfa']."-¥n";
?>
```

PHP 5.0.5 より前のバージョンでは、 このようにして配列の要素に参照を代入することができませんでした。 現在はできるようになっています。

本来は PHP 5.0.x で正常に動作すべきだったコード

参照関連の修正が行われるより前の PHP 5.0 ではいくつかのバグが報告されていましたが、 それらは今では正常に「動作します」。しかし、いずれの場合についても PHP 5.1.x ではエラーがスローされます。なぜなら、まず第一にそのコードが不正なものだからです。 現在では self:: を使用して値を参照で返すことができるようになりましたが、 **E_STRICT** レベルの警告をスローします。 また、オーバーロードされたオブジェクトに参照を代入しようとすると、 たとえ代入自体は正常に動作していても **E_ERROR** に遭遇することになるでしょう。

現れて、そしていなくなった警告

参照を返す関数をネストしてコールすることは、PHP 4.3.x および PHP 5.1.x のどちらでも有効です。しかし、それらの間のバージョンの PHP では、 **E_NOTICE** あるいは **E_STRICT** が間違って発生してしまいます。

```
<?php
function & foo() {
    $var = 'ok';
    return $var;
}

function & bar() {
    return foo();
}

$a =& bar();
echo "$a¥n";
?>
```

[] の読み込み

```
<?php
class XmlTest {

    function test_ref(&$test) {
        $test = "ok";
    }

    function test($test) { }

    function run() {
        $ar = array();
        $this->test_ref($ar[]);
        var_dump($ar);
        $this->test($ar[]);
    }
}

$o = new XmlTest();
$o->run();
?>
```

これは常に **E_ERROR** レベルの致命的なエラーを発生します。なぜなら、PHP では [] を読み込みに使用できないからです。PHP 4.4.2 および PHP 5.0.5 以降で、 これは不正なコードとなりました。

関数のパラメータとしての整数値

PHP 5.0 の登場に伴い、パラメータのパース用の新しい API が公開されました。 多くの PHP 関数でこの API が使用されています。PHP 5.0.x から 5.1.x までの間のすべてのバージョンではこの API での整数の処理が非常に厳しいものとなっており、 パラメータとして整数を受け付けている PHP 関数に不正な値を渡せないようになっていました。 このチェックは現在ではより緩やかなものとなっており、" 123" や "123 " といった不正な値でも受け付けるように変更されています。これは PHP 5.0.x でも同様です。しかし、コードの安全性の確保や入力内容の検証を促進するため、そのような文字列が整数として渡された際には PHP 関数は **E_NOTICE** を発行します。

クラスとオブジェクトの変更点

- instanceof、 is_a()、is_subclass_of() および catch
- 抽象 private メソッド
- インターフェイスのアクセス修飾子
- 継承規則の変更
- クラス定数

instanceof、is_a()、 is_subclass_of() および catch

PHP 5.0 で is_a() は非推奨となり、instanceof 演算子に置き換えられました。 instanceof の初期の実装にはいくつかの問題があり、クラスを探すために __autoload() に頼っていました。もしクラスが存在しなかった場合、 __autoload() に失敗するために instanceof は致命的な **E_ERROR** をスローしていました。 同じ理由で、catch 演算子や is_subclass_of() 関数でも同様の現象が発生していました。

PHP 5.1 では、これらの関数や演算子は __autoload() をコールしません。 また、PHP 5.0.x ではこの問題の回避策として class_exists() が使用可能です。 大きな問題ではありませんが、この回避策はもはや必要ありません。

抽象 private メソッド

PHP 5.0.0 から PHP 5.0.4 まででは抽象 private メソッドがサポートされていましたが、 これは禁止されました。なぜなら private と abstract とは決して両立することのない概念だからです。

インターフェイスのアクセス修飾子

PHP 5.0 では、インターフェイス内での関数定義はクラス内での関数定義と同じように扱われました。 2004 年 10 月以降はそうではなくなり、インターフェイス内での関数定義では public 以外のアクセス修飾子を指定できなくなりました。 そして 2005 年 4 月、PHP 5.0b1 のリリース前に、static 修飾子も許可されるようになりました。しかし、protected 修飾子や private 修飾子を指定しようとすると E_ERROR をスローします。abstract も同様です。 この変更が既存のコードに影響を及ぼすことはないはずです。 結局のところ、インターフェイス内でこれら (protected, private, abstract) を指定してもそれは意味のないものだからです。

継承規則の変更

PHP 5.0 では、基底クラスの同名の関数定義と一致しない関数を 派生クラスの中で定義することが可能でした。たとえば以下のようなものです。

PHP 5.1.x では、このコードは E_STRICT エラーを発生させます。

```
<?php
class Base {
     function &return_by_ref() {
         $r = 1;
          return $r;
     }
}

class Derived extends Base {
     function return_by_ref() {
          return 1;
     }
}
?>
```

クラス定数

PHP 5.0.x では、以下のコードは正しいものでした。

PHP 5.1.x では、クラス定数を再定義すると、致命的なエラー E_ERROR がスローされます。

```
<?php
class test {
     const foobar = 'foo';
     const foobar = 'bar';
}

?>
```

拡張モジュール

PHP コアから削除された拡張モジュール

PHP 5.1.x をダウンロードして最初に気づくことのひとつは、 いくつかの古い拡張モジュールがなくなっていることでしょう。 これらの拡張モジュールのうち、現在も使用可能で活発にメンテナンスが続けられているものについては PHP Extension Community Library (PECL) » http://pecl.php.net/ で入手可能です。Windows 版のバイナリも定期的にビルドされており、 PHP 5.1.x 用の PECL 拡張モジュールのバイナリは » http://pecl4win.php.net/ から取得可能です。

削除された拡張モジュール

拡張モジュール	代替モジュール/開発状況
ext/cpdf	pecl/pdflib
ext/dbx	pecl/dbx
ext/dio	pecl/dio
ext/fam	メンテナンスが滞っています
ext/ingres_ii	pecl/ingres
ext/ircg	メンテナンスが滞っています
ext/mcve	pecl/mcve
ext/mnogosearch	メンテナンスが滞っています
ext/oracle	ext/oci8 あるいは ext/pdo_oci
ext/ovrimos	メンテナンスが滞っています
ext/pfpro	メンテナンスが滞っています
ext/w32api	» pecl/ffi
ext/yp	メンテナンスが滞っています
ext/activescript	» pecl/activescript

PECL モジュールのうち、メンテナンスが滞っているもの (いつの間にかサポートされなくなっていたり、現在メンテナがいなかったり、 PECL パッケージとしてリリースされていなかったりするもの) は、 » http://cvs.php.net/pecl の CVS から取得可能です。しかし、これらのモジュールはサポートされておらず、 インストールして使用するのは大変かもしれません。

PHP 5.1.x の新しい拡張モジュールのクラス定数

Zend Engine 2.1 の API では、拡張モジュールの開発者が クラス定数を定義することが可能となりました。PHP 5.1.x 用に書かれた新しい拡張モジュールである SPL や PDO、 XMLReader そして date などはこの形式の定数を使用しており、 C 言語風の **PDO::CLASS_CONSTANT** ではなく **PDO_CLASS_CONSTANT** のようになっています。 これは、PHP のグローバル名前空間の汚染を最小限に抑えるためです。

日付/時刻のサポート

PHP 5.1.x では日付/時刻のサポートが完全に書き直され、 タイムゾーンを「知る」ためにシステムの設定を使用しないようになりました。 そのかわりに、以下の順番でタイムゾーンを取得します。

- date_default_timezone_set() 関数を使用して設定したタイムゾーン（もし設定されていれば）
- 環境変数 TZ の値（もし空でなければ）
- "magical" な推測（もし OS がそれをサポートしていれば）
- これらのいずれの項目も存在しない場合は、UTC

正確性を保証する（そして警告 **E_STRICT** を出さないようにする）ためには、php.ini の中に以下の形式でタイムゾーンを定義する必要があります。

```
date.timezone = Europe/London
```

サポートされるタイムゾーンをこの形式で並べた一覧表が、PHP マニュアルの 付録 にあります。

また、strtotime() 関数は、失敗時に -1 ではなく **FALSE** を返すようになったことにも注意しましょう。

データベースサポートの変更

- PDO の概要
- MySQL サポートの変更
- SQLite サポートの変更

PDO の概要

PHP Data Objects (PDO) は PHP 5.0 の時に PECL 拡張モジュールとして公開され、その後 PHP 5.1.x で PHP のコア配布物に組み込まれるようになりました。 PDO 拡張モジュールはデータベースへのアクセスのための一貫したインターフェースを提供し、 データベース固有の PDO ドライバとともに使用します。 各ドライバはデータベース固有の関数を保持していることもありますが、 クエリの発行やデータの取得といった基本的な機能については PDO 関数がカバーしています。 この関数は、 PDO::__construct() で指定したドライバを使用します。

PDO 拡張モジュールやドライバは、 共有モジュールとしてのビルドを想定していることに注意しましょう。 これにより、PECL からドライバをアップグレードする際に PHP そのものを再ビルドする必要がなくなります。

PHP 5.1.x のリリース時点では、PDO は広範囲に及ぶテストを済ませており ほとんどの環境でうまく動作するはずです。しかし、PDO やそのドライバは比較的歴史が浅く、 データベース固有の機能のいくつかを実装できていないということを理解しておくことが大切です。 新しいプロジェクトで PDO を使用する際には、事前に動作検証を十分に行うようにしましょう。

既存のコードは、一般的にはこれまで存在したデータベース拡張モジュールに依存しています。 これらについてもメンテナンスが続けられます。

MySQL サポートの変更

PHP 4 では、MySQL 3 のサポートが組み込まれていました。 PHP 5.0 のリリース時には「mysql」および「mysqli」という 2 つの MySQL 拡張モジュールがありました。これらは、それぞれ MySQL < 4.1 および MySQL 4.1 以降をサポートするように設計されていました。 PHP でサポートされるあらゆるデータベースの API への高速なインターフェースを提供する PDO が公開されたことに伴い、PDO を使用して書かれた PHP コードでは、 PDO_MYSQL ドライバによって現行のいずれのバージョン (MySQL 3、4 あるいは 5) でもサポートすることが可能となりました。これは、コンパイル時に使用した MySQL ライブラリのバージョンに依存します。 後方互換性を確保するためにこれまでの MySQL 拡張モジュールも残されていますが、 デフォルトでは有効になりません。

SQLite サポートの変更

PHP 5.0.x では、組み込みの sqlite 拡張モジュールで SQLite 2 がサポートされていました。 PHP 4.3 および PHP 4.4 でも、PECL 拡張モジュールとして同じものが使用可能でした。 PDO が公開されたことで、sqlite 拡張モジュールは 'sqlite2' という名前の PDO ドライバとして機能するようになりました。このため、PHP 5.1.x での sqlite 拡張モジュールは PDO 拡張モジュールに依存しています。

PHP 5.1.x では、sqlite への接続用にいろいろなインターフェイスを公開しています。

sqlite 拡張モジュールでは "classic" な 手続き型/オブジェクト指向 API を提供しており、 これは以前のバージョンの PHP と同じように使用できます。 またこれとは別に PDO 'sqlite2' ドライバを提供しており、こちらは PDO API を使用して SQLite 2 データベースにアクセスします。

さらに、PDO_SQLITE は 'sqlite' バージョン 3 ドライバを提供します。 SQLite バージョン 3 は SQLite バージョン 2 に比べて非常に優れていますが、 2 つのバージョンの間でファイルフォーマットに互換性がありません。

以前のバージョンの PHP で動作している SQLite ベースのプロジェクトがあるのなら、 ext/sqlite を使用し続ければ何の問題もありません。しかし、PDO および sqlite を明示的に有効にする必要があります。新しいプロジェクトの場合は、PDO および 'sqlite' (バージョン 3) ドライバを使用すべきです。 これは SQLite 2 に比べて高速で、ロック処理が改善されており、 プリペアドステートメントやバイナリ列をネイティブにサポートしています。

SQLite 拡張モジュールを使用するには、PDO を有効にする必要があります。 もし PDO を共有モジュールとしてビルドしたのなら、 SQLite 拡張モジュールも同様に共有モジュールとする必要があります。 これは、PDO ドライバを提供している他のすべての拡張モジュールについても同様です。

E_STRICT のチェック

ひとつのスクリプトについてのみチェックしたいのであれば、 PHP コマンドラインの文法チェック機能を使用して **E_STRICT** エラーを抜き出すことができます。

```
    php -d error_reporting=4095 -l script_to_check.php
```

大量のファイルをチェックするのなら、以下のシェルスクリプトで同様のことが可能です。

```
#!/bin/sh

directory=$1

shift

# チェックする拡張子を指定します
extensions="php inc"

check_file ()
{
  echo -ne "Doing PHP syntax check on $1 ..."

  # オプション
   ERRORS=`/www/php/bin/php -d display_errors=1 -d html_errors=0 -d error_prepend_string=" " -d error_append_string=" " -d

  if test -z "$ERRORS"; then
    echo -ne "OK."
  else
    echo -e "Errors found!¥n$ERRORS"
  fi

  echo
}

# loop over remaining file args
for FILE in "$@" ; do
  for ext in $extensions; do
      if echo $FILE | grep "¥.$ext$" > /dev/null; then
        if test -f $FILE; then
          check_file "$FILE"
        fi
      fi
  done
done
```

PHP 4 から PHP 5 への移行

目次

PHP 5における変更点

PHP 5 および PHP 5 に組み込まれた Zend Engine 2 は、 PHP の性能と機能を著しく向上させました。 同時に既存のコードへの影響を最小限とするよう配慮されているため、 PHP 4 から 5 への移行は非常に簡単なはずです。 多くの既存の PHP 4 用のコードは、変更せずに動作するはずです。 しかし、若干の差異に ついても知っておく必要があります。 また、実運用環境のバージョンを変更する前にあなたのコードを十分にテストする必要があります。

下位互換性のない変更点

多くの既存の PHP 4 のコードは変更無しで動作するはずですが、 以下の下位互換性がない変更点について注意する必要があります。

- 新たな予約キーワードがいくつか あります。
- strrpos() と strripos() は、 needle として文字列全体を使用するようになりました。
- 文字列オフセットの不正な使用は、**E_WARNING** ではなく **E_ERROR** を発生します。不正な使用の例は、次のようなものです。 `$str = 'abc'; unset($str[0]);`
- array_merge() は、配列のみを指定できるよう 変更されました。配列以外の変数を指定した場合は、各パラメータ毎に **E_WARNING** が発生します。あなたのコードで 突然 **E_WARNING** が発生し始める可能性が あるため注意してください。
- サーバー変数 PATH_TRANSLATED は、Apache2 SAPI では暗黙のうちに設定されません。 この動作は、Apache により設定されていない場合に SCRIPT_FILENAME と同じ値に 設定する PHP 4 と異なります。 この変更は、» CGI の規約に従う ためのものです。詳細については、 » bug #23610 を参照してください。 また、マニュアルの $_SERVER['PATH_TRANSLATED'] についての記述も参照してください。 この問題は、PHP のバージョン >= 4.3.2 でも影響を及ぼします。
- 定数 **T_ML_CONSTANT** は、 Tokenizerエクステンションで 定義されなくなりました。error_reporting に**E_ALL** を指定した場合、PHPは通知(notice)を生成します。 **T_ML_CONSTANT**は、全く使用されていませんが、 PHP 4 では定義されていました。PHP 4とPHP 5の両方において、 // と /* */ は、共に **T_COMMENT** 定数と 解釈されます。 しかし、PHP5からPHPにより解釈されるようになった PHPDoc形式のコメント /** */ は、**T_DOC_COMMENT** と認識されます。
- variables_order に "S"が含まれる場合に、$_SERVER が argc および argv と共に設定されます。 $_SERVERを作成しないような特別な設定を行った場合、当然、argc および argv は 設定されません。 この変更は、CLI 版において variables_order の 設定によらず常に

argc および argv を作成するために行われました。 これにより、CLI 版では、常にグローバル変数 $argc および $argv が設定されるようになりました。

- プロパティを持たないオブジェクトはもはや"空"とはみなされません。
- クラスを使用する前に宣言する必要がある場合もあります。 これは、PHP 5 の新機能（たとえば interfaces など）を使用する場合にのみ生じます。 その他の場合の動作は従来と同じです。
- get_class(), get_parent_class() および get_class_methods() は、 ケース依存（大文字小文字を区別）で 宣言時に使用されたクラス/メソッドの名前を返すようになっています これにより、以前の動作（クラス名は小文字で返される）に依存している古いスクリプト では問題を発生する可能性があります。 解決策としては、全てのスクリプトでこれらの関数を検索し、 strtolower()を使用するというものが考えられます。 このケース依存性に関する変更は、 自動的に定義される定数 __CLASS__, __METHOD__, および __FUNCTION__にも適用されます。 これらの値は、宣言時と同様に（ケース依存で）返されます。
- ip2long() は、 無効なIPアドレスが関数の引数として渡された場合、 -1ではなく、FALSE を返すようになりました。
- 読み込まれるファイルの中で関数が宣言されている場合、それが return() の前もしくは後のどちらにあるかにかかわらず、 メインファイルの中でその関数を使用可能です。 このファイルが二度読み込まれると、関数が既に宣言済みであるため、 PHP 5 は致命的なエラーを発生します。一方、PHP 4 ではエラーを発生しません。 ファイルが読み込み済であるかどうかを調べて 読みこまれたファ入ルの内容を条件分岐でかえすのではなく、 include_once() を使用することを推奨します。
- Windows上で、 include_once() と require_once() は、まず、読みこまれるファイルのパスを正規化します。 これにより、A.php と a.php を読みこむ場合でもファイルは一度だけ読みこまれます。

Example#1 strrpos() と strripos() は、 needleとして文字列全体を使用する

```
<?php
var_dump(strrpos('ABCDEF','DEF')); //int(3)

var_dump(strrpos('ABCDEF','DAF')); //bool(false)
?>
```

Example#2 プロパティを持たないオブジェクトはもはや"空"とはみなされない

```
<?php
class test { }
$t = new test();

var_dump(empty($t)); // echo bool(false)

if ($t) {
    // 実行される
}
?>
```

以下の例はPHP 4では有効でしたが、PHP 5では致命的なエラーを 発生します。

Example#3 クラスは使用前に宣言する必要がある

```
<?php
$test = new fubar();
$test->barfu();

class fubar {
     function barfu() {
         echo 'fubar';
     }
}
?>
```

CLI と CGI

PHP 5 では、CLI および CGI のファイル名にいくつかの変更があります。 PHP 5 では、CGI版は php-cgi.exe に変更されました。 (以前は php.exe) CLI版はメインディレクトリに 置かれるようになりました。(以前は、cli/php.exe))

PHP 5では、新しいモード php-win.exe が追加されました。 これは、CLI版と同じですが、php-win は出力を行わず、コンソールを提供しない ところが異なります。("DOS窓"は現れません。) この動作は、php-gtk に似ています。

PHP 5では、CLI版は常にグローバル変数$argv と $argcを設定します。

設定ファイルの移行

SAPIモジュールの名前がphp4xxxからphp5xxxに変更されたため、 設定ファイルを変更する必要があります。 また、CLIおよびCGIのファイル名も変更されています。詳細については、 対応するセクションを 参照してください。

Apache の設定の移行は非常に簡単です。必要な変更については、 以下の例を参照してください。

```
# change this line:
LoadModule  php4_module  /php/sapi/php4apache2.dll

# with this one:
LoadModule  php5_module  /php/php5apache2.dll
```

サーバ上でCGIモードでPHPを実行している場合、 CGI版の名前がphp.exeからphp-cgi.exeに変更されたことに注意する必要があります。 Apacheでは、以下のようにする必要があります。

```
# change this line:
Action  application/x-httpd-php  "/php/php.exe"

# with this one:
Action  application/x-httpd-php  "/php/php-cgi.exe"
```

他のWebサーバの場合、CGIまたはSAPIモジュールのファイル名を変更する 必要があります。

新しい関数

PHP 5 では、新しい関数がいくつかあります。 以下にそれらのリストを示します。

配列:

- array_combine() - 配列 1 つをキー、他の配列をその値として指定し、配列を作成する
- array_diff_uassoc() - ユーザが定義したコールバック関数により添字のチェックを行ない、 配列の差異を計算する。
- array_udiff() - データ比較用のコールバック関数を用いて配列の比較を行う
- array_udiff_assoc() - 添字の確認を行いつつ配列の比較を行う。データは、コールバック関数により 比較される。
- array_udiff_uassoc() - 添字の確認を行いつつ配列の比較を行う。データは、コールバック関数により 比較される。添字確認もコールバック関数により行われる。
- array_walk_recursive() - 配列の各メンバにユーザ関数を再帰的に適用する。
- array_uintersect_assoc() - 添字の確認も含め、配列の共通項を計算する。 データの比較にはコールバック関数を使用する。
- array_uintersect_uassoc() - 添字の確認も含め、配列の共通項を計算する。 データおよび添字の比較には、それぞれ個別のコールバック関数を使用する。
- array_uintersect() - 添字の確認も含め、配列の共通項を計算する。 データの比較にはコールバック関数を使用する。

InterBase:

- ibase_affected_rows() - 直近のクエリにより作用された行の数を返す
- ibase_backup() - サービスマネージャのバックアップタスクを起動し、 ただちに戻る
- ibase_commit_ret() - トランザクションを閉じることなくコミットする
- ibase_db_info() - データベースに関する統計情報を得る
- ibase_drop_db() - データベースを破棄する
- ibase_errcode() - エラーコードを返す
- ibase_free_event_handler() - 登録されているイベントハンドラをキャンセルする
- ibase_gen_id() - ジェネレータIDをインクリメントし、新しい値を返す
- ibase_maintain_db() - データベースサーバ上で管理用コマンドを実行する
- ibase_name_result() - 結果セットに名前を付ける
- ibase_num_params() - プリペアドクエリのパラメータの数を返す
- ibase_param_info() - プリペアドクエリのパラメータに関する情報を返す
- ibase_restore() - サービスマネージャのリストアタスクを起動し、 ただちに戻る
- ibase_rollback_ret() - トランザクションをロールバックし、 トランザクションコンテキストに留まる
- ibase_server_info() - データベースサーバについての統計情報を得る
- ibase_service_attach() - サービスマネージャに 接続する
- ibase_service_detach() - サービスマネージャとの接続を切る
- ibase_set_event_handler() - イベントがポストされた時にコールされる コールバック関数を登録する
- ibase_wait_event() - データベースによりポストされるイベントを待つ

iconv:

- iconv_mime_decode() - MIMEヘッダフィールドをデコードする
- iconv_mime_decode_headers() - 複数のMIMEヘッダフィールドを一度にデコードする
- iconv_mime_encode() - MIMEヘッダフィールドを構築する
- iconv_strlen() - 文字列の文字数を返す
- iconv_strpos() - 対象文字列の中で指定した文字列が最初に表れる場所をみつける
- iconv_strrpos() - 対象文字列の中で指定した文字列が最後に表れる場所をみつける
- iconv_substr() - 文字列の一部を切り出す

ストリーム:

- stream_copy_to_stream() - 一つのストリームから 別のストリームにデータをコピーする
- stream_get_line() - ストリームリソースから指定したデリミタまで行を取得する
- stream_socket_accept() - stream_socket_server()により作成されたソケット上の 接続を受け入れる
- stream_socket_client() - インターネットまたは Unixドメインソケット接続を受け入れる
- stream_socket_get_name() - ローカルまたはリモートソケットの名前を取得する
- stream_socket_recvfrom() - 接続の有無によらずソケットからデータを受信する
- stream_socket_sendto() - 接続の有無によらずソケットにメッセージを送信する
- stream_socket_server() - インターネットまたはUnixドメインソケットを作成する

日付・時刻関連:

- idate() - ローカルの時刻/日付を整数値としてフォーマットする
- date_sunset() - 指定した日付および場所での日没時刻
- date_sunrise() - 指定した日付および場所での日の出時刻
- time_nanosleep() - 指定した秒/ナノ秒の待機

文字列:

- str_split() - 文字列を配列に変換する
- strpbrk() - 文字列の中で一連の文字のどれかを探す
- substr_compare() - オフセット、最大文字長を指定して二つの文字列を比較を行う。 バイナリ対応で、オプションで大文字/小文字を無視できる。

その他:

- convert_uudecode() - UUエンコードされた文字列を デコードする
- convert_uuencode() - 文字列をUUエンコードする
- curl_copy_handle() - cURL ハンドルを、 その設定も含めてコピーする
- dba_key_split() - 文字列表現されたキーを配列表現に分割する
- dbase_get_header_info() - dBaseデータベースのヘッダ情報を取得する
- dbx_fetch_row() - DBX_RESULT_UNBUFFEREDフラグがセットされた クエリ結果からレコードを取得する
- fbsql_set_password() - 指定したユーザのパスワードを変更する
- file_put_contents() - 文字列をファイルに書き込む
- ftp_alloc() - アップロードされるファイル用の領域を確保する
- get_declared_interfaces() - 宣言済みの全インターフェイスを配列として返す
- get_headers() - HTTPリクエストへのレスポンスとしてサーバに送信されたヘッダを 全て取得する
- headers_list() - 送信された(または送信準備ができた)レスポンスヘッダのリストを返す
- http_build_query() - URLエンコードされた クエリ文字列を生成する
- image_type_to_extension() - getimagesize(), exif_read_data(), exif_thumbnail(), exif_imagetype()により返されるイメージ型の ファイル拡張子を得る
- imagefilter() - カスタム引数を用いてイメージにフィルタを適用する
- imap_getacl() - 指定したメールボックスのACLを取得する
- ldap_sasl_bind() - SASLによりLDAPディレクトリにバインドする
- pcntl_getpriority() - プロセスの優先度を取得する
- **mb_list_encodings()** - サポートされる全てのエンコーディングの配列を返す
- pcntl_getpriority() - 任意のプロセスの優先度を取得する
- pcntl_wait() - waitpid()システムコールにより定義され、フォークされた子プロセスを 待つか、またはステータスを返す
- pg_version() - クライアント、プロトコル、サーバのバージョン(取得可能な場合)を配列として返す
- php_check_syntax() - 指定したファイルの構文を確認する
- php_strip_whitespace() - コメントと 空白を除いたソースを返す
- proc_nice() - カレントプロセスの優先度を変更する
- pspell_config_data_dir() - 言語データファイルの 位置を変更する
- pspell_config_dict_dir() - 主単語リストの位置を 変更する
- setrawcookie() - 値を URL エンコードせずにクッキーを送信する
- scandir() - 指定したパス内のファイルとディレクトリを一覧表示する
- snmp_read_mib() - MIB ファイルを読み込み、アクティブなMIBツリーとしてパースする
- sqlite_fetch_column_types() - 指定したテーブルのカラム型を配列として返す

注意: Tidy 拡張モジュールの API も、 完全に変わりました。

新しいディレクティブ

php.ini にも変更が行われており、PHP 5でディレクティブが 追加されています。以下のそのリストを示します。

- mail.force_extra_paramaters - 指定されたパラメータをextraパラメータとしてsendmailバイナリに強制的に追加 します。これらのパラメータは、セーフモードの場合でも mail() の5番目のパラメータの値を置換します。
- register_long_arrays - は、古い $HTTP_*_VARS をPHPが登録するかどうかを設定します。
- session.hash_function - は、ハッシュ関数を選択します。(MD5またはSHA-1)
- session.hash_bits_per_character - は、バイナリハッシュデータを可読な形式に変換する際に各文字に保存するビット数 (4 から 6 まで)を指定します。
- zend.ze1_compatibility_mode - Zend Engine 1 (PHP 4)互換モードを有効にします。

データベース

PHP 5では、データベースに関していくつかの変更があります。(MySQL および SQLite).

PHP 5では、ライセンス上の問題やその他の問題により、MySQLクライアントライブラリは バンドルされていません。 詳細については、 FAQエントリを参照してください。

新しい拡張モジュールとして、MySQL 4.1以降で動作するよう設計された MySQLi (Improved MySQL) もあります。

PHP 5以降、SQLite 拡張モジュールが PHPに組み込まれています。 SQLiteは、組み込み可能なSQLデータベースエンジンで、 (MySQLまたはPostgreSQLのような) 大きなデータベースサーバーへの接続に使用されるクライアントライブラリでは ありません。SQLiteは組み込み可能なSQLデータベースエンジンで、 クライアントライブラリではなく、ディスク上のデータベースファイルから 直接読み書きします。

新しいオブジェクトモデル

PHP 5 では、新しいオブジェクトモデルが採用されています。 PHP におけるオブジェクトの処理は完全に書き直され、性能と機能が向上しています。以前のバージョンでは、(integer や string といった) 通常の型と同じようにオブジェクトを扱っていました。 この方式の欠点は、変数にオブジェクトを代入したり メソッドのパラメータとしてオブジェクトを渡した場合に オブジェクト全体がコピーされてしまうということでした。 新しい方式では、オブジェクトを指すハンドルを用いてオブジェクトを管理します。 値そのものを管理するのではなくなります (ハンドルというのは、 要はそのオブジェクトの ID のことです)。

Many PHP programmers aren't even aware of the copying quirks of the old object model and, therefore, the majority of PHP applications will work out of the box, or with very few modifications.

新しいオブジェクトモデルについては 言語リファレンス を参照ください。

PHP 5 では、クラス名と同じ名前の関数がそのクラス内で定義されている場合にのみ それがコンストラクタとしてコールされます。 PHP 4 では、同じクラスだけでなく親クラスで定義されている場合にもコールされていました。

PHP 4 との互換性については、 zend.ze1_compatibility_mode ディレクティブも参照ください。

エラー報告

PHP 5 以降、新しいエラー報告定数 E_STRICT が追加されました。値は 2048 です。 これは、コードの可搬性や将来のバージョンとの互換性に関する警告を実行時に表示するものです。 これを使用することで、将来のバージョンを見据えた最適なコーディングを行うことができます。 たとえば、廃止予定の関数を使用している場合などに STRICT メッセージが発生します。

> **注意**: E_ALL には E_STRICT は含まれないので、デフォルトでは有効にはなりません。

PHP 3 から PHP 4 への移行

目次

PHP 4での変更点

PHP 4とこれに組み込まれたZendエンジンは、PHPの性能と機能を大 幅に向上させますが、既存のコードへの変更は最小限ですむように多くの 考慮がなされています。このため、PHP 3から4への移行は、 PHP/FI 2.0から PHP 3への移行と比べて非常に容易です。既存の PHP 3のコードの多くは無修正で実行可能です。しかし、実用環境でバージョ ンを切替える際には、バージョン間の若干の差異を知り、使用するコード について確認を行うことが必要です。以下の記述は、これらの事項につい ていくつかの示唆を与えるものです。

PHP 3とPHP 4を同時に実行する

最近のオペレーティングシステムでは、バージョン管理及びスコープ監理 を行う機能が提供されています。この機能により、 1 つのApacheサーバ上のモジュールとしてPHP 3とPHP 4を同時に実行することが可能です。

この機能は、以下のプラットフォーム上での動作が確認されています。

- 最近のbinutils (binutils 2.9.1.0.25 でテスト済み) を備えたLinux
- Solaris 2.5以降
- FreeBSD (3.2, 4.0 でテスト済み)

この機能を使用するには、PHP 3 および PHP 4 で APXS (--with-apxs) および 必要なリンク拡張 (--enable-versioning) を指定します。それ以外の場合、 通常の命令が使用されます。例えば、

```
$ ./configure ¥
   --with-apxs=/apache/bin/apxs ¥
   --enable-versioning ¥
   --with-mysql ¥
   --enable-track-vars
```

設定ファイルの移行

グローバル設定ファイルphp3.iniは、名前が php.iniに変更となりました。

Apache設定ファイルの場合、変更点はやや多くなっています。PHPモジュー ルにより認識されるMIME型が変更となりました。

```
application/x-httpd-php3           -->     application/x-httpd-php
application/x-httpd-php3-source    -->     application/x-httpd-php-source
```

以下の構文を使用することにより、(サーバに組み込まれているバージョンに基づき)両方のバージョンで動作する設定ファイルを作成することが可能です。

```
AddType   application/x-httpd-php3           .php3
AddType    application/x-httpd-php3-source  .php3s

AddType   application/x-httpd-php            .php
AddType    application/x-httpd-php-source    .phps
```

加えて、Apache用のPHPディレクティブが変更となっています。

PHP 4以降、PHPに関連するApacheディレクティブは以下の4種類のみとなっています。

```
php_value [PHPディレクティブ名] [値]
php_flag [PHPディレクティブ名] [On|Off]
php_admin_value [PHPディレクティブ名] [値]
php_admin_flag [PHPディレクティブ名] [On|Off]
```

Adminの値とそれ以外の値には次の2つの違いがあります。

- Admin の値(またはフラグ)は、サーバ全体のApache設定ファイル(すなわちhttpd.conf)でのみ設定可能です。
- 通常の値(またはフラグ)はセーフモードのようなある種のPHPディレクティブを制御できません(セーフモードの設定を.htaccessで上書きできるとしたら、セーフモードの目的は達成されなくなります)。逆に Admin値では全てのPHPディレクティブの値を修整可能です。

移行の過程をより容易にするために、PHP 4にはApache設定命令と .htaccessファイルをPHP 3とPHP 4の両方で動作するように変換するスクリプトがバンドルされています。これらのスクリプトは、MIME型の行を変換しません。これらは自分で変更を行う必要があります。

Apache設定ファイルを変換するために、(scripts/apache/ ディレクトリに ある) apconf-conv.sh スクリプトを実行してください。 例えば、

```
~/php4/scripts/apache:#    ./apconf-conv.sh  /usr/local/apache/conf/httpd.conf
```

元の設定ファイルは、httpd.conf.origに保存されます。

.htaccess ファイルを変換するには、(同じくscripts/apache/ディレクトリにある)aphtaccess-conv.sh スクリプトを実行してください。

```
~/php4/scripts/apache:#   find / -name .htaccess -exec ./aphtaccess-conv.sh {} ¥;
```

同様に古い.htaccessファイルには.origが付加されて保存されます。

この変換スクリプトを使用するには、'awk' をインストールしておく必要があります。

パーサの動作

パーサと実行は、二つの完全に分割されたステップになりました。ファイル中のコードの実行は、ファイル全体と必要なものが全て完全にパースされるまで行われません。

この分割により PHP 4では require や include で読み込まれるファイルが構文的に完全であることが新たに必要となっています。PHP 4では、制御構造の異なる制御部をファイル境界をまたいで配置することはできません。 このため、for ループまたは while ループ、if 命令または switch ブロックをあるファイルで開始し、 else、endif, case、break 命令を別のファイルに置くことはできません。

ループまたは他の制御構造の中で追加のコードを読み込むことは、PHP 4 では、全く問題ありません。制御用のキーワードや対応する波括弧 {...} だけは、同じコンパイル単位(ファイルまたは eval()された文字列)の中に置く必要があります。

しかし、配布されているコードではこうした記法は非常に悪いスタイルで あるとみなされているようなので、この制約の影響は大きくないはずです。

使用ができなくなった他の機能としては、PHP 3ではほとんど使用され ていませんが、requireで読みこまれたファイルから値を返すというもの があります。includeで読み込まれたファイルから値を返すことは PHP 4でも可能です。

エラー出力

設定変更

PHP 3では、エラー出力レベルは異なったエラーレベルを意味する数 を合計した単純な数値で表されていました。よく使用されるのは、全て のエラーと警告を出力する15と好ましくない形式や事項を通知するメッ セージ以外の全てを出力する7です。

PHP 4では、より多くの異なったエラーおよび警告レベルを設定して おり、設定ファイルにおいて意図する動作を設定するために定数シンボ ルを使用することが可能になっています。

エラー出力レベルは、出力したくないエラーメッセージの警告レベルを 定数シンボル E_ALL と排他的論理和をとることに より明示的に取り除くことにより指定することも可能になっています。 解りにくいでしょうか? では、ここで定数シンボル E_NOTICE に分類される簡単な警告以外のエラー出力を行うように設定したい場合を考えます。この場合、 php.iniにおいて次のような指定を行います。 error_reporting = E_ALL & ~ (E_NOTICE) ここでさらに警告の出力を行わないようにしたい場合は、次のように括 弧の中でバイナリまたは演算子'|'を使用して適当な定数を追加します。 error_reporting= E_ALL & ~ (E_NOTICE | E_WARNING)

警告

PHP 3 から PHP 4 にアップグレードする際には、 これらの設定を確認したうえで error_reporting() をコールします。 新しいエラー型、特に E_COMPILE_ERROR を無効にしたくなるかもしれませんが、 そうすると、何かエラーが発生したときに真っ白なページが表示されてしまい、 原因がつかみにくくなることがあります。

警告

従来の値7および15を使用してエラー出力を設定する方法は、パースエラーを含む新たに追加されたエラークラスのいくつかが使用できなくなるため、非常に悪い方法です。この場合、スクリプトが動作しない場合 でもエラーメッセージをどこにも表示しないため、非常に奇妙な動作と なることがあります。

これにより、過去において、原因を追求が困難なスクリプトエンジンの 問題と報告された多くの再現不可能なバグレポートが、実際には、 requireで読み込まれたファイルにいくつかの '}' が足りず、エラー出力の設定を誤っていたためにパーサがエラーを出力できなかったという ことがあります。

このため、スクリプトが何も出力せずに終了する場合、エラー出力設定 をまず確認するべきです。現在のZend エンジンは十分に完成されており、こ

の種の奇妙な動作を発生しません。

追加された警告メッセージ

既存のPHP 3コードの多くは、このコードのように非常に悪いスタイル の言語構造を使用しており、意図された通りに動作している場合でも、 他の部分を変更すると動作しなくなることがあります。PHP 4は、PHP 3が出力しないような場面でも多くの通知メッセージを出力します。 E_NOTICE メッセージをオフにすれば簡単に修正できますが、 通常は問題のあるコードを修正する方が良いでしょう。

通知メッセージが出力される場面で最も多いのが、引用符で括られてい ない文字列定数を配列の添字として使用している場合です。その名前の キーワードまたは定数が存在しない場合、PHP 3と4は共にこれら を文字列として解釈し直します。しかし、そのコードのどこか他の場所 でその名前の定数が定義されている場合、このスクリプトは意図した通 りに動作しない可能性があります。これは、侵入者が文字列定数を再定 義して使用されるスクリプトが侵入者が有さないアクセス権を侵入者に 与えるようにする場合には、セキュリティのリスクとなることさえあり ます。そこで、PHP 4では、例えば、 $HTTP_SERVER_VARS[REQUEST_METHOD] のように引用 符で括られていない文字列定数を使用した場合に警告を発生するように なっています。これを $_SERVER['REQUEST_METHOD'] に変更する ことにより、パーサは正常に動作し、コードのスタイルとセキュリティ は大幅に改善されます。

その他、PHP 4では初期化されていない変数または配列要素を使用し ていることを知らせるようになっています。

イニシャライザ

静的変数とクラスメンバのイニシャライザは、PHP 3では有効な式なら ばなんでも指定可能でしたが、PHP 4ではスカラー値のみが指定可能で す。これは、前記のようにパースと実行が分割されており、パーサがイニ シャライザを処理する際にはまだコードは実行されていないためです。

クラスの場合、メンバ変数を初期化する際にはコンストラクタを使用する べきです。静的変数の場合、単純なスカラー値以外のものに意味があるこ とはまれです。

---

empty("0")

恐らく最も動作上の問題のある変更は、empty()の動 作に関するものでしょう。文字 '0' (ゼロ)のみを含む文字列は、PHP 3 とは異なり空文字列とみなされるようになりました。

数値の入力が要求された場合でも全てのinputフィールドは文字列を返し、 PHPは自動的に型変換を行う機能を有しているため、この動作の変更は、Webアプリケーションにおいて有意義なものです。しかし、一方、コード の正常動作を阻害する可能性があり、動作の内容を知らない場合には、 原因を追求しがたい動作の不備を生じる可能性があります。

廃止された関数

PHP 4は多くの新しい機能と拡張を導入していますが、同時にいくつか のバージョン3の関数は廃止されています。コア関数のいくつかは、 Zendエンジンを使用する4で導入されたパーサと実行モジュールの分割 という新たな形態では動作しないため、廃止されています。他の関数や拡 張機能全体もより新しい関数や拡張が同じタスクをより良くより一般的な 手法で行うため古くなっています。いくつかの関数は単にまだ移植されて いないだけであり、いくつかの関数や拡張はライセンス上の問題により廃 止されています。

コンセプトの変更により廃止された関数

PHP 4ではパースと実行が分割されており、実行時に(現在、Zendエン ジンに組み込まれている)パーサの動作を変更することは、パースが実行 前に既に行われているため、もうできません。 php.ini ファイルで適当な値を設定することによ りパーサの動作を変更することが可能です。

PHP 3の機能でPHP 4でサポートされていない機能としては、 このマニュアルで記述されている実験的なデバッグインターフェースの サポートがあります。独立した本格的なデバッガがZendの製品としてリ リースされています。

推奨されない関数と拡張

Adabas および Solid データベース拡張は、もはや推奨されません。 今後は、unified ODBC 拡張を代わりに使用してください。

unset()のステータスの変更

unset()はまだ使用可能ですが、PHP 4ではリテラ ルとして実装されるように変更されており、このため、'真の'関数とし てはもはや数えられていません。

これは、unset() の動作を変更するものではないの で、直接的な変更点があるわけではありませんが、 function_exists() により "unset" を調べた場合、 echo()のような他の低レベル関数と同様に FALSE が返されます。

その他のより実際的な変更としては、unset() を間 接的にコールすることができなくなったというものがあります。このた め、$func="unset"; $func($somevar) はもう動作 しません。

PHP 3拡張機能

PHP 3用に描かれた拡張モジュールは、PHP 4ではバイナリレベルでも ソースレベルでも動作しません。元のソースにアクセス可能な場合、 PHP 4にその拡張モジュールを移植することは困難ではありません。 実際の移植の手順についての詳細な説明はこの文章には(まだ)含まれて いません。

文字列中の変数置換

PHP 4では文字列中の新しい変数置換の機構が追加されています。文字 列中の多次元配列からオブジェクトのメンバ変数と要素にアクセスが可能 となっています。

これを行うには、開き括弧の直後にドル記号が付くように変数を波括弧で 括る必要があります。 {$...}

オブジェクトのメンバ変数の値を文字列に埋め込むには、 "text {$obj->member} text" とします。一方、PHP 3 では、"text ".$obj->member." text" のよ うにする必要がありました。

これによりコードの可読性が高まりますが、PHP 3用に描かれた既存の スクリプトは動作しなくなる可能性があります。しかし、コード中の {$ とい

う文字の組み合わせを調べることにより容易 にこの種の問題の確認が可能で、任意の検索/置換ツールにより ¥{$ に置換することにより対処可能です。

クッキー

PHP 3は、コード中のsetcookie()のコールとは逆 の順番でクッキーを設定するという好ましくない動作をしていました。 PHP 4はこの動作を変更し、コードでクッキーを設定したのと同じ順番 で正しくクッキーを設定するヘッダ行を生成します。

これによりいくつかの既存のコードが動作しなくなる可能性がありますが、 従来の動作は理解しにくく、将来的な更なる問題を防ぐために変更望まれていました。

グローバル変数の扱い

PHP 3とPHP 4の初期のバージョンでは、グローバル変数の扱いは「簡単であること」 に重点が置かれていましたが、PHP 4では「安全であること」に焦点が当てられています。 PHP 3では後述の例は問題なく動作しますが、PHP 4では unset(unset($GLOBALS["id"])); とする 必要があります。これはグローバル変数の扱いに関する一つの例にすぎません。 PHP 4では多くの場合に $GLOBALS 変数を使用しなければならなくなりました。 詳細についてはグローバル リファレンスの章を参照してください。

Example#1 グローバル変数の移行

```
<?php
$id = 1;
function test()
{
     global $id;
     unset($id);
}
test();
echo($id); // PHP 4では1と表示されます
?>
```

PHP/FI 2 から PHP 3 への移行

目次

3.0 における互換性

PHP 3.0 は 1 から書き直されました。 これには 2.0 のパーサに比べ、 より堅牢で一貫性のある適切なパーサが内蔵されています。 3.0 はまた、劇的に速くなり、メモリ消費量も少なくなっています。 しかしながら、これらの改善事項のうちのいくつかは、 書式と機能の両面において互換性を欠いた変更になってしまっています。

加えて、PHP の開発者は、 PHP 3.0 における書式と意味の双方をきれいにし直そうとしましたが、 これも互換性を欠く要因となっています。 長い目で見れば、 これらの変更はより良いものであると私たちは信じています。

この章では、あなたが PHP/FI 2.0 から PHP 3.0 へ移行する際に遭遇するであろう非互換性と、 それらへの解決策についてのガイドを提供しようと思います。 必要でない限り、新しい機能については述べられていません。

あなたの古い PHP/FI 2.0 スクリプトを自動的に変換できる変換プログラムがあります。 これは PHP 3.0 ディストリビューションの convertor サブディレクトリにあります。 このプログラムは文法的な変更を捕らえるだけですので、 どちらにしてもこの章を注意深く読む必要があるでしょう。

---

old_function

old_function を使用すると、 PHP/FI2 と同じ構文で関数を宣言できるようになります (ただし、'function' を 'old_function' に書き換える必要があります)。

この機能は非推奨です。PHP/FI2->PHP 3 コンバータ以外で使用してはいけません。

警告

old_function で宣言した関数は、 PHP の内部コードからコールすることはできません。 つまり、usort() や array_walk()、そして register_shutdown_function() といった関数内では使用できないということです。 この制限を回避するには、その old_function をコールするラッパー関数を (通常の PHP 3 形式で) 作成します。

開始／終了タグ

PHP の開始と終了のタグが変わっていることに、 おそらく最初に気付かれるでしょう。 古い <? > 形式は、3 つの新しい 形式に置き換えられました。

Example#1 移行: 古い開始／終了タグ

```
<? echo "これは PHP/FI 2.0 のコードです。\n"; >
```
バージョン 2.0 で、 PHP/FI は次のバリエーションもサポートするようになりました。

Example#2 移行: 最初の新しい開始／終了タグ

```
<? echo "これは PHP 3.0 のコードです!\n"; ?>
```
終了タグは、単なる '>' に代わり '?>' で構成されます。 しかしながら、サーバ上で XML を使いたい場合は、 この最初の形式では不具合が生じるでしょう。 なぜなら、PHP は XML ドキュメントの中の XML マークアップを PHP コードとして実行しようとするかもしれないからです。 このため、以下のバリエーションが導入されました。

Example#3 移行: 2 番目の新しい開始／終了タグ

```
<?php echo "これは PHP 3.0 のコードです!\n"; ?>
```
エディタ上で、 処理している命令タグが全く認識されないという問題が発生した方がいました。 Microsoft FrontPage はそんなエディタのうちの 1 つです。 これらを回避するため、さらに以下のバリエーションが導入されました。

Example#4 移行: 3 番目の新しい開始／終了タグ

```
<script language="php">

  echo "これは PHP 3.0 のコードです!\n";

</script>
```

---

if..endif の文法

if(); elseif(); else; endif; を使って if/elseif/else ステートメントを記述するための「選択肢」については、 3.0 パーサに対してかなり複雑な処理を追加してやらないと、 効率的な実装を行うことができません。このため、 文法が変更されました。

Example#1 移行: 古い if..endif の文法

```
if ($foo);
     echo "はい\n";
elseif ($bar);
     echo "だいたい\n";
else;
     echo "いいえ\n";
endif;
```

Example#2 移行: 新しい if..endif の文法

```
if ($foo):
     echo "はい\n";
elseif ($bar):
     echo "だいたい\n";
else:
     echo "いいえ\n";
endif;
```
評価式を終了させるもの(endif)を除き、 すべてのステートメントにおいてセミコロンはコロンに変更されました。

---

while の文法

if..endif と同様に、while..endwhile の文法も変更されました。

Example#1 移行: 古い while..endwhile の文法

```
while ($more_to_come);
    ...
endwhile;
```

Example#2 移行: 新しい while..endwhile の文法

```
while ($more_to_come):
    ...
endwhile;
```

警告

PHP 3.0 で古い形式の while..endwhile を使った場合は、 無限ループになってしまいます。

式の型

PHP/FI 2.0 では、結果の型を決めるのに左辺式を使っていました。 PHP 3.0 では、結果の型を決めるのに両辺を使うようになったので、 3.0 環境の元で 2.0 のスクリプトを実行すると、 予期しない結果になる場合があります。

以下の例を考えてみましょう。

```
$a[0]=5;
$a[1]=7;

$key = key($a);
while ("" != $key) {
     echo "$keyn";
     next($a);
}
```
PHP/FI 2.0 では、この例は $a の 2 つのインデックス双方を表示します。 一方、PHP 3.0 ではなにも表示されません。この理由は、PHP 2.0 では、 左側の引数が文字列なので文字列の比較が行われますが、もちろん "" は "0" とは等しくならないためです。 一方 PHP 3.0 では、文字列が数字と比較されると、 (文字列が数字に変換され、) 数字による比較が行われます。 0は atoi("") で比較され、 さらに変数リスト の方も 0 と評価され、0==0 なので、 その結果ループは 1 度も実行されないということになります。

これを解決するのは簡単です。 while ステートメントを以下のように変更します。

```
while ((string)$key != "") {
```

エラーメッセージの変更

PHP 3.0 のエラーメッセージは、2.0 よりも通常の場合正確になりました。 しかし、もはやソースコードのうちエラー原因となった部分は表示されません。 そのかわり、エラーの原因となったファイル名と行番号が表示されるので、 驚くかもしれません。

短絡的なブール評価

PHP 3.0 におけるブール評価は短絡的です。というのは、 (1 || test_me()) のような評価式があるとすると、 関数 **test_me()** は実行されないということです。 これは、1 を評価した後は、 最終的な評価結果を変えることはできないと分かっているからです。

これは互換性の問題としては些細なことかもしれませんが、 思わぬ副作用があるかもしれません。

関数の TRUE/FALSE 返り値

ほとんどの内部関数が書き直されたのに伴い、PHP/FI 2.0 では成功時に 0、失敗時に -1 を返していたのが、それぞれ **TRUE** と **FALSE** を返すように変更されました。 この新しい振る舞いにより、 $fp = fopen("/your/file") or fail("darn!"); のようなより論理的なコードを書く事ができるようになりました。 PHP/FI 2.0 には、関数がその実行に失敗した時に、 何を返すべきかという明瞭なルールがなかったので、 そのようなスクリプトのほとんどは、2.0 から 3.0 のコンバータを使った後に、 手作業でチェックしなければならないと思います。

Example#1 2.0 からの移行: 返り値、古いコード

```
$fp = fopen($file, "r");
if ($fp == -1);
     echo("$file を読み込み専用として開くことが出来ませんでした<br />\n");
endif;
```

Example#2 2.0 からの移行: 返り値、新しいコード

```
$fp = @fopen($file, "r") or print("$file を読み込み用として開くことができませんでした<br />\n");
```

その他の互換性

- PHP 3.0 の Apache モジュールは、もはや バージョン 1.2 以前の Apache をサポートしません。Apache 1.2 以降が必要となります。
- echo() はもはやフォーマット文字列をサポート しません。printf() 関数を使ってください。
- PHP/FI 2.0 では、実装面に起因する副作用として、 $foo[0] が $foo と同じ値になるということがありました。 これは PHP 3.0 では等しくなくなりました。
- $array[] で配列を読み込む機能は、 もはやサポートされません。 すなわち、$data = $array[] といったループで配列の中身を取り出すことはできないということです。 この代わりに current() と next() を使ってください。 また、$array1[] = $array2 では $array2 の値を $array1 に追加することにはならず、 $array1 の最後のエントリに $array2 を追加することになってしまいます。 多次元配列のサポートを参照してください。
- "+" は、もはや文字列の連結演算子にはならず、 その代わりにその引数を数値に変換して数値の加算を行ってしまいます。 "." を使用してください。

Example#1 2.0 からの移行: 文字列の連結

```
echo "1" + "1";
```

PHP 2.0 ではこれで 11 が表示されますが、PHP 3.0 ではこれは 2 になります。代わりに次のようにしてください。

```
echo  "1"."1";
$a = 1;
$b = 1;
echo $a + $b;
```

これは PHP 2.0 でも 3.0 でも 2 になります。

```
$a = 1;
$b = 1;
echo  $a.$b;
```

これは PHP 3.0 では 11 になります。

PHP のデバッグ

目次

デバッガについて

PHP 3 ではネットワーク対応のデバッガがサポートされています。

PHP 4 内部の機能としてはデバッガはサポートされていませんが、 外部デバッガモジュールを使用することが出来ます。 » Zend IDE にはデバッガが付属していますし DBG(» http://dd.cron.ru/dbg/) や » Advanced PHP Debugger(APD)、 » Xdebug といったフリーの外部デバッガモジュールもあります。 » Xdebug はこのセクションでも 述べますが PHP3 のデバッグ機能と互換性のあるデバッグインターフェースを持っています。

デバッガの使用法

PHP 3 の内部デバッガはバグを追跡し、回避するのに役立ちます。 このデバッガは、PHP 3 が開始されるたびに TCP ポートに 接続することにより動作します。 そのリクエストからのすべてのエラーメッセージは、 この TCP コネクションに送られます。 この情報は IDE や(Emacs のような) プログラマブル・エディタの中で動作する 「デバッグ用サーバ」のために使われます。

デバッガの設定方法:

1. 設定ファイル (debugger.port) でデバッガ用の TCP ポートを設定し、それを (debugger.enabled) で有効にします。
2. どこかでそのポートに対して TCP リスナを設定します。 (たとえば、Unix systems では**socket -l -s 1400**)
3. コードの中で「debugger_on(host)」 を実行します。ここで host は TCP リスナが動作しているホストの IP アドレス又はホスト名です。

これで、すべての警告や通知などがそのリスナのソケット上に表示されます。 この表示は、この後 error_reporting() でエラー出力を抑制しても、有効のままです。

デバッガのプロトコル

PHP 3 のデバッガのプロトコルは行ベースです。各行には タイプ を持ち、また メッセージ を構成する行もあります。 各メッセージはタイプ start を持つ行で始まり、タイプ end を持つ行で終わります。 PHP 3 は異なったメッセージを持つ行を同時に送ることがあります。

A line has this format:

date time host(pid) type: message-data

日付
ISO 8601 フォーマットの日付 (yyyy-mm-dd)
時刻
マイクロ秒単位の時刻: hh:mm:uuuuuu
ホスト名
スクリプト・エラーを生成したホストのDNS名 またはIPアドレス。
プロセス ID
ホストにおいて このエラーを生成した PHP 3 スクリプトのプロセス ID
タイプ

行のタイプ。以降に続くデータをどう解釈すべきかを、 受信プログラムに知らせます。

デバッガの行タイプ

名前	意味
start	デバッガ・メッセージが、 ここから始まることを受信プログラムに伝える役割をします。 データ の内容は、 以下に示すエラーメッセージのタイプです。
message	PHP 3 エラーメッセージ
location	エラーが発生したファイル名と行番号。最初の location 行は常に最上位レベルの位置が入っています。 データは ファイル名: 行番号 のようになります。 message の後、および、すべての function の後の行には常に location が付きます。
frames	後に続くスタックダンプの中のフレーム数。もし4フレームあれば、 4レベルに渡ってコールされた関数の情報があることになります。 "frames" 行がない場合は、ネストの深さは0 (エラーはトップレベルで起こった)と仮定されます。
function	エラーが発生した関数名。関数コール・スタックにおいて、 各レベル毎に繰り返されます。
end	受信プログラムに対して、 デバッガのメッセージがここで終わることを知らせます。

データ
行データ

デバッガのエラータイプ

デバッガ	PHP 3 内部
警告	E_WARNING
エラー	E_ERROR
パース	E_PARSE
通知	E_NOTICE
コア・エラー	E_CORE_ERROR
コア警告	E_CORE_WARNING
未定義	(その他)

Example#1 デバッガメッセージの例

```
1998-04-05 23:27:400966 lucifer.guardian.no(20481) start: notice
1998-04-05 23:27:400966 lucifer.guardian.no(20481) message: Uninitialized variable
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: (NULL):7
1998-04-05 23:27:400966 lucifer.guardian.no(20481) frames: 1
1998-04-05 23:27:400966 lucifer.guardian.no(20481) function: display
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: /home/ssb/public_html/test.php3:10
1998-04-05 23:27:400966 lucifer.guardian.no(20481) end: notice
```

Configure オプション

中心となる configure オプションのリスト

以下のリストは、Unix 系の環境で PHP をコンパイルする際に用いられる configure スクリプトのオプションの一部です。 ほとんどのオプションはそれぞれの拡張モジュールのリファレンスページで 説明されており、ここにはありません。最新の configure オプションの 完全なリストを得るには、PHP のソースディレクトリで **autoconf** を実行した後、**./configure --help** を実行してください（インストールと設定 も参照ください）。--prefix=PREFIX のような追加の **configure** オプションについて興味がある方は、 » GNU configure のドキュメントが 参考になるでしょう。

> **注意**: これらの設定はコンパイル時にのみ使用可能です。PHP の動作を 実行時に設定したい場合は、実行時設定 の章を参照ください。

PHP 4 の Configure オプション

> **注意**: これらのオプションは PHP 4.1.0 の時点でのものです。中にはもっと古い PHP 4 あるいは PHP 3 でさえも有効なものがあるかもしれませんが、PHP 4.1.0 でしか使えないものもあるでしょう。古いバージョンをコンパイルされる場合、 おそらくいくつかのオプションは使えないでしょう。

一般的なオプション

--enable-debug

デバッグシンボルつきでコンパイルします。

--with-layout=TYPE

インストールされるファイルのレイアウトを設定します。Type には PHP（デフォルト）または GNU のどちらかが指定できます。

--with-pear=DIR

PEAR を DIR（デフォルトは PREFIX/lib/php）にインストールします。

--without-pear

PEAR をインストールしません。

--enable-sigchild

PHP 独自の SIGCHLD ハンドラを有効にします。

--disable-rpath

実行時にライブラリの検索パスを追加できないようにします。

--enable-libgcc

明示的に libgcc とリンクします。

--enable-php-streams

実験的な PHP ストリーム機能を組み込みます。PHP そのものをテストする 目的以外では使用しないでください。

--with-zlib-dir[=DIR]

zlib のインストールディレクトリを指定します。

--enable-trans-sid

透過的なセッション ID の伝播を有効にします。PHP 4.1.2 以前でのみ 有効です。PHP 4.2.0 以降、この機能は常に有効となっています。

--with-tsrm-pthreads

POSIX スレッドを使用します（デフォルト）。

--enable-shared[=PKGS]

共有ライブラリをビルドします [default=yes]。

--enable-static[=PKGS]

静的ライブラリをビルドします [default=yes]。

--enable-fast-install[=PKGS]

インストール速度を上げるように最適化します [default=yes]。

--with-gnu-ld

C コンパイラが GNU ld を使用するとみなします [default=no]。

--disable-libtool-lock

ロックを回避します（並列ビルドに失敗する可能性があります）。

--with-pic

PIC/non-PIC オブジェクトのどちらか一方のみを使用するようにします [default=use both]。

--enable-memory-limit

メモリ制限（memory limit）のサポートつきでコンパイルします (PHP 5.2.1 以降では使用できず、常に有効となります)。

--disable-url-fopen-wrapper

URL を指定した fopen で、HTTP や FTP を経由してファイルを開く機能を無効にします (PHP 5.2.5 以降では使用できません)。

--enable-versioning

要求されるシンボルのみをエクスポートします。 詳細な情報は INSTALL を参照ください。

--with-imsp[=DIR]

IMSp サポートを有効にします（DIR は IMSP の include ディレクトリ および libimsp.a のディレクトリです）。 PHP 3 限定です!

--with-mck[=DIR]

Cybercash MCK サポートを有効にします。DIR は cybercash mck のビルドディレクトリで、デフォルトは /usr/src/mck-3.2.0.3-linux です。詳細な情報は extra/cyberlib を参照ください。PHP 3 限定です!

--with-mod-dav=DIR

Apache mod_dav を用いた DAV サポートを有効にします。 DIR は mod_dav のインストールディレクトリです （Apache モジュールバージョンのみ!）。 PHP 3 限定です!

--enable-debugger

リモートデバッグ関数を有効にします。PHP 3 限定です!

--enable-versioning

Solaris 2.x および Linux が提供するバージョン管理機能を 有効にします。PHP 3 限定です!

PHP オプション

--enable-maintainer-mode

一般の利用者にとってあまり便利でない（そして時に混乱のもととなる） make ルールや依存性を有効にします。

--with-config-file-path=PATH

php.ini を探すパスを設定します。デフォルト値は PREFIX/lib です。

--enable-safe-mode

セーフモードをデフォルトで有効にします。

--with-exec-dir[=DIR]

セーフモード時には、このディレクトリにある実行ファイルのみ実行を許可します。 デフォルト値は /usr/local/php/bin です。

--enable-magic-quotes

magic quotes をデフォルトで有効にします。

--disable-short-tags

開始タグの短縮形 <? をデフォルトで無効にします。

SAPI オプション

以下のリストには、PHP で有効な SAPI（Server Application Programming Interface）が含まれています。

--with-aolserver=DIR

インストールされている AOLserver のパスを指定します。

--with-apxs[=FILE]

Apache 共有モジュールをビルドします。FILE は Apache apxs ツールへの パスで、デフォルトは apxs です。Apache のソース Tarball に含まれる apxs を指定するのではなく、そのシステムに実際にインストールされている バージョンの apxs を指定するようにしましょう。

--with-apache[=DIR]

静的 Apache モジュールをビルドします。DIR は Apache のビルドディレクトリで、 デフォルトは /usr/local/apache です。

--with-mod_charset

（ロシアの Apache 用の）mod_charset のための変換テーブルを 有効にします。

--with-apxs2[=FILE]

Apache 2.0 共有モジュールをビルドします。FILE は、オプションで指定する Apache apxs ツールへのパスです。デフォルトは apxs です。

--with-caudium=DIR

PHP を、Caudium で使用する Pike モジュールとしてビルドします。 DIR は Caudium サーバのディレクトリで、デフォルトは /usr/local/caudium/server です。

--disable-cli

PHP 4.3.0 で使用可能になりました。PHP の CLI バージョンのビルドを 無効にします（自動的に --without-pear も指定されます）。詳細な情報は PHP をコマンドラインから使用する を参照ください。

--enable-embed[=TYPE]

組み込み SAPI ライブラリのビルドを有効にします。TYPE は shared あるいは static のいずれかで、デフォルトは shared です。 PHP 4.3.0 で使用可能になりました。

--with-fhttpd[=DIR]

fhttpd モジュールをビルドします。DIR は fhttpd のソースディレクトリで、 デフォルトは /usr/local/src/fhttpd です。 PHP 4.3.0 以降、このオプションは使用できなくなりました。

--with-isapi=DIR

PHP を、Zeus の ISAPI モジュールとしてビルドします。

--with-nsapi=DIR

インストール済みの Netscape/iPlanet/SunONE Webserver へのパスを指定します。

--with-phttpd=DIR

このオプションについての情報はありません。

--with-pi3web=DIR

PHP を Pi3Web のモジュールとしてビルドします。

--with-roxen=DIR

PHP を Pike モジュールとしてビルドします。DIR は Roxen の ディレクトリで、通常は /usr/local/roxen/server です。

--enable-roxen-zts

Zend Thread Safety を使用した Roxen モジュールをビルドします。

--with-servlet[=DIR]

サーブレットのサポートを含めます。DIR は JSDK のインストールディレクトリです。 java 拡張モジュールが共有ライブラリとしてビルドされていることが 前提条件です。

--with-thttpd=SRCDIR

PHP を thttpd モジュールとしてビルドします。

--with-tux=MODULEDIR

PHP を TUX モジュールとしてビルドします（Linux 限定）。

--with-webjames=SRCDIR

PHP を WebJames モジュールとしてビルドします（RISC OS 限定）。

--disable-cgi

PHP の CGI バージョンをビルドしないようにします。 PHP 4.3.0 で使用可能になりました。

--enable-force-cgi-redirect

サーバ内でのリダイレクトの際のセキュリティチェックを有効にします。 Apache で CGI バージョンを稼動させる場合、このオプションを 使用すべきです。

--enable-discard-path

有効にすると、PHP の CGI バイナリを web ディレクトリツリーの外に 配置できるようになります。これで .htaccess によるセキュリティを 回避できなくなります。

--with-fastcgi

PHP を FastCGI アプリケーションとしてビルドします。PHP 4.3.0 以降、 このオプションは使用できません。代わりに --enable-fastcgi を使用してください。

--enable-fastcgi

有効にすると、CGI モジュールが FastCGI のサポートつきでビルドされます。 PHP 4.3.0 で使用可能になりました。

--disable-path-info-check

無効にすると、/info.php/test?a=b のようなパスは 動作しません。PHP 4.3.0 で使用可能になりました。詳細な情報は » Apache マニュアルを参照ください。

---

php.ini ディレクティブ

目次

php.ini ディレクティブのリスト

以下のリストには、PHP の設定を行うための php.ini ディレクティブが 含まれます。

設定オプション

名前	デフォルト	変更の可否	変更履歴
allow_call_time_pass_reference	"1"	PHP_INI_PERDIR	PHP <= 4.0.0 で PHP_INI_ALL。PHP 6.0.0 で削除。
allow_url_fopen	"1"	PHP_INI_ALL	PHP <= 4.3.4 で PHP_INI_ALL。PHP < 6 で PHP_INI_SYSTEM。PHP 4.0.4 以降で利用可能。
allow_url_include	"0"	PHP_INI_ALL	PHP 5 で PHP_INI_SYSTEM。PHP 5.2.0 以降で使用可能。
always_populate_raw_post_data	"0"	PHP_INI_PERDIR	PHP <= 4.2.3 で PHP_INI_ALL。 PHP 4.1.0 以降で利用可能。
apc.cache_by_default	"1"	PHP_INI_ALL	APC <= 3.0.12 で PHP_INI_SYSTEM。APC 3.0.0 以降で利用可能。
apc.enabled	"1"	PHP_INI_SYSTEM	APC 2 で PHP_INI_SYSTEM、APC <= 3.0.12 で PHP_INI_ALL。
apc.enable_cli	"0"	PHP_INI_SYSTEM	APC 3.0.7 以降で利用可能。
apc.file_update_protection	"2"	PHP_INI_SYSTEM	APC 3.0.6 以降で利用可能。
apc.filters	NULL	PHP_INI_SYSTEM	
apc.gc_ttl	"3600"	PHP_INI_SYSTEM	
apc.include_once_override	"0"	PHP_INI_SYSTEM	APC 3.0.12 以降で利用可能。
apc.localcache	"0"	PHP_INI_SYSTEM	APC 3.0.14 以降で利用可能。
apc.localcache.size	"512"	PHP_INI_SYSTEM	APC 3.0.14 以降で利用可能。
apc.max_file_size	"1M"	PHP_INI_SYSTEM	APC 3.0.7 以降で利用可能。
apc.mmap_file_mask	NULL	PHP_INI_SYSTEM	
apc.num_files_hint	"1000"	PHP_INI_SYSTEM	
apc.optimization	"0"	PHP_INI_ALL	APC 2 で PHP_INI_SYSTEM、APC 3.0.13 で削除。
apc.report_autofilter	"0"	PHP_INI_SYSTEM	APC 3.0.11 以降で利用可能。
apc.rfc1867	"0"	PHP_INI_SYSTEM	APC 3.0.13 以降で利用可能。
apc.rfc1867_freq	"0"	PHP_INI_SYSTEM	
apc.rfc1867_name	"APC_UPLOAD_PROGRESS"	PHP_INI_SYSTEM	
apc.rfc1867_prefix	"upload_"	PHP_INI_SYSTEM	
apc.shm_segments	"1"	PHP_INI_SYSTEM	
apc.shm_size	"30"	PHP_INI_SYSTEM	
apc.slam_defense	"0"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.stat	"1"	PHP_INI_SYSTEM	APC 3.0.10 以降で利用可能。
apc.stat_ctime	"0"	PHP_INI_SYSTEM	APC 3.0.13 以降で利用可能。
apc.ttl	"0"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.user_entries_hint	"4096"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.user_ttl	"0"	PHP_INI_SYSTEM	APC 3.0.0 以降で利用可能。
apc.write_lock	"1"	PHP_INI_SYSTEM	APC 3.0.11 以降で利用可能。

名前	デフォルト	変更の可否	変更履歴

PHP_INI_* 定数の定義

定数	値	意味
PHP_INI_USER	1	ユーザスクリプトまたはWindowsレジストリ で設定可能なエントリ
PHP_INI_PERDIR	2	*php.ini, .htaccess* または *httpd.conf*で設定可能なエントリ
PHP_INI_SYSTEM	4	*php.ini* または *httpd.conf*で設定可能なエントリ
PHP_INI_ALL	7	どこでも設定可能なエントリ

コア php.ini ディレクティブに関する説明

このリストには、PHPを設定する際に使用可能なコア php.ini ディレクティブが 含まれています。拡張モジュールにより処理されるディレクティブは、それぞれの 拡張モジュールのドキュメントページにリストと詳細が記述されています。 例えば、セッション用ディレクティブに関する情報は、セッションのページにあります。

httpd オプション

httpd オプション

名前	デフォルト	変更の可否	変更履歴
async_send	"0"	PHP_INI_ALL	

言語オプション

言語およびその他の設定オプション

名前	デフォルト	変更の可否	変更履歴
short_open_tag	On	PHP_INI_PERDIR	PHP <= 4.0.0 で PHP_INI_ALL
asp_tags	"0"	PHP_INI_PERDIR	PHP <= 4.0.0 で PHP_INI_ALL
precision	"14"	PHP_INI_ALL	
y2k_compliance	"1"	PHP_INI_ALL	
allow_call_time_pass_reference	"1"	PHP_INI_SYSTEM¦PHP_INI_PERDIR	PHP <= 4.0.0 で PHP_INI_ALL
expose_php	"1"	*php.ini* のみ	
zend.ze1_compatibility_mode	"0"	PHP_INI_ALL	PHP 5.0.0 以降で利用可能

以下に設定ディレクティブに関する簡単な説明を示します。

short_open_tag *boolean*

PHP タグの短縮型（**<? ?>** ）を使用 可能にするかどうかを設定します。PHPをXMLと組み合わせて用いる 場合は、**<?xml ?>** をファイル中で用 いるためにこのオプションをオフにする必要があります。オンにし た場合にPHPでXMLを出力するには、例えば、次のようにします。 **<?php echo '<?xml version="1.0"'; ?>** これをオフにした場合、長い方の形式のタグ（**<?php ?>** ）を使用する必要があります。

> **注意**: このディレクティブは、**<? echo** と同じ形式の ショートカット**<?=** も影響を受けます。 このショートカットを使用するには、*short_open_tag* をonとする 必要があります。

asp_tags *boolean*
ASP 形式のタグ <% %> を通常のタグ <?php ?> に加えて使用可能にします。 このスイッチにより、出力の短縮形 <%= $value %> も 使用できるようになります。 詳細な情報は、HTML からのエスケープを参照ください。

> **注意**: ASP形式のタグは3.0.4で追加されました。

precision *integer*
浮動小数点数に関して表示される最大桁数を指定します。
y2k_compliance *boolean*
2000年問題対応を強制します（2000年問題非対応のブラウザにおいて、 問題が発生する可能性があります）。
allow_call_time_pass_reference *boolean*

関数のコール時に引数が参照で渡された場合に、警告するかどうかを 設定します。この機能は過去のものであり、将来のバージョンの PHP/Zend ではサポートされない可能性があります。 推奨される方法は、関数宣言時に参照渡しとするべき引数を指定する ことです。将来のバージョンでの動作を保障するために、この オプションを off とし、スクリプトがこの状態で正しく動作することを 確認することが推奨されます（この機能を使用する度に警告が発生します）。

関数コール時に参照で引数を渡すことは、コードの明解さを損なうために 廃止されています。関数は、引数が参照渡しであると宣言されて いない場合でも、文書化されていない方法で、その引数を修正できます。 副作用を回避するためには、どの引数を参照渡しとするかを関数宣言でのみ 指定すると良いでしょう。

参照に関する説明も 参照ください。

expose_php *boolean*

（例えば、Web サーバヘッダに PHP のサインを追加することにより、）PHP がサーバにインストールされていることを表示するかどうかを 指定します。これは全くセキュリティ上の脅威ではなく、サーバ上 で PHP を使用しているかどうかを調べられるようにするものです。

zend.ze1_compatibility_mode *boolean*

Zend Engine 1 (PHP 4) との互換モードを有効にします。 この設定は、オブジェクトのコピー、キャスト (プロパティを 保持しないオブジェクトが FALSE あるいは 0 のいずれになるか)、 そして 比較 に影響を与えます。このモードの場合、オブジェクトを渡す際の デフォルトの方法は、参照渡しではなく値渡しとなります。

PHP 4 から PHP 5 への移行というタイトルの セクションも参照してください。

リソース制限

リソース制限

名前	デフォルト	変更の可否	変更履歴
memory_limit	"128M"	PHP_INI_ALL	PHP 5.2.0 より前は "8M"、PHP 5.2.0 では "16M"

以下に設定ディレクティブに関する簡単な説明を示します。

memory_limit integer

スクリプトが確保できる最大メモリをバイト数で指定します。この命令は、 正しく書かれていないスクリプトがサーバーのメモリを食いつぶすことを防止するのに役立ちます。 もし、使用可能メモリに制限を設けたくない場合は、 ここに -1 を指定してください。

PHP 5.2.1 より前のバージョンでは、このディレクティブを使うためには、 コンパイル時に configure で --enable-memory-limit を指定しなければなりません。 これは、関数 memory_get_usage() および memory_get_peak_usage() を使用する際にも必要となります。

integerを使用する際、その値はバイト単位で測られます。 この FAQ に記載された短縮表記を使用することも可能です。

max_execution_time も参照ください。

パフォーマンスチューニング

パフォーマンスチューニング

名前	デフォルト	変更の可否	変更履歴
realpath_cache_size	"16K"	PHP_INI_SYSTEM	PHP 5.1.0 以降で使用可能。
realpath_cache_ttl	"120"	PHP_INI_SYSTEM	PHP 5.1.0 以降で使用可能。

以下に設定ディレクティブに関する簡単な説明を示します。

realpath_cache_size integer

PHP が使用する realpath キャッシュの大きさを設定します。 PHP で大量にファイルをオープンする際に、 この値を大きくすることによってファイル操作のパフォーマンスを 向上させます。

realpath_cache_ttl integer

与えられたファイルやディレクトリについての realpath 情報キャッシュの有効期限を (秒単位で) 設定します。 ファイルを変更することがほとんどない場合は、 この値を大きくすることを検討してください。

データ処理

データ処理設定オプション

名前	デフォルト	変更の可否	変更履歴
track_vars	"On"	PHP_INI_??	
arg_separator.output	"&"	PHP_INI_ALL	PHP 4.0.5 以降で利用可能
arg_separator.input	"&"	PHP_INI_SYSTEM¦PHP_INI_PERDIR	PHP 4.0.5 以降で利用可能
variables_order	"EGPCS"	PHP_INI_PERDIR	PHP <= 5.0.5 では PHP_INI_ALL。
auto_globals_jit	"1"	PHP_INI_PERDIR	PHP 5.0.0 以降で利用可能
register_globals	"0"	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL
register_argc_argv	"1"	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL
register_long_arrays	"1"	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL
post_max_size	"8M"	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_SYS。PHP 4.0.3 以降で利用可能
gpc_order	"GPC"	PHP_INI_ALL	
auto_prepend_file	""	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL
auto_append_file	""	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL
default_mimetype	"text/html"	PHP_INI_ALL	
default_charset	"iso-8859-1"	PHP_INI_ALL	
always_populate_raw_post_data	"0"	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL。PHP 4.1.0 以降で利用可能
allow_webdav_methods	"0"	PHP_INI_PERDIR	

以下に設定ディレクティブに関する簡単な説明を示します。

track_vars boolean

サーバ変数はそれぞれ、グローバル連想配列 $_ENV, $_GET, $_POST, $_COOKIE, $_SERVERで参照することができます。

PHP 4.0.3 以降, track_vars は常にonとなっている ことに注意してください。

arg_separator.output string

PHPがURLを生成する際にURL引数を区分するために使用されるセパレータ。

arg_separator.input string

入力されたURLを変数にパースする際にPHPが使用するセパレータのリスト。

注意: このディレクティブで指定した全ての文字は、 セパレータとして認識されます!

variables_order string

EGPCS (Environment (環境変数)、 Get、Post、 Cookie (クッキー)、そして Server) 変数のパースの順番を設定します。例えば variables_order を "SP" に設定すると、PHP は superglobals $_SERVER および $_POST を作成しますが、 $_ENV、$_GET および $_COOKIE は作成しません。"" に設定すると、一切 superglobals を設定しません。

非推奨のディレクティブ register_globals が on になっていると (これは PHP 6.0.0 で廃止される予定です)、 variables_order の設定は、ENV、 GET、POST、 COOKIE および SERVER の各変数がグローバルスコープに取り込まれる順番も左右します。 つまり、たとえば variables_order が "EGPCS" で register_globals が有効になっていたとすると、 $_GET['action'] と $_POST['action'] が両方設定された場合の $action の値は $_POST['action'] の内容になります。 これは、この例におけるディレクティブの設定で P が G より後になっているからです。

警告

CGI および FastCGI SAPI の両方で、 $_SERVER にも環境変数の値が格納されます。 つまり、S を指定すると、 ES と指定したのと同じ意味になります。 これは、E が他のどこかで指定されていたとしても同じです。

注意: $_REQUEST の内容や順序も、このディレクティブの影響を受けます。

auto_globals_jit boolean

有効にした場合、SERVER および ENV 変数はスクリプトの開始時ではなく、 最初に使用された時 (Just In Time) に作成されるようになります。これらの変数がスクリプトの中で使用されない場合、 このディレクティブを使用することで性能が向上します。

このディレクティブを有効にするには、PHP ディレクティブ register_globals, register_long_arrays, および register_argc_argv を無効にしておく必要があります。PHP 5.1.3 以降では、 register_argc_argv を無効にする必要はありません。

register_globals boolean

EGPCS(Environment, GET, POST, Cookie, Server)変数を グローバル変数として登録するかどうかを指定します。

» PHP 4.2.0 以降、 このディレクティブのデフォルトは、offです。

関連する情報については、セキュリティの章の register_globalsの使用 を参照してください。

register_globals は、 (ini_set()で)実行時に設定することができないことに 注意してください。しかし、前記のようにホストが許可している場合には、 .htaccess を使用することができます。 .htaccess エントリの例を以下に示します。 **php_flag register_globals off**

注意: register_globals は、 variables_order ディレクティブの影響を受けます。

警告

この機能は 非推奨 であり、PHP 6.0.0 で 削除 されます。この機能を使用しないことを強く推奨します。

register_argc_argv boolean
PHPが変数argvとargcを宣言するかどうかを指定します (これらにはGETの情報が格納されます)。 コマンドライン も参照ください。このディレクティブはPHP 4.0.0で 追加されました。以前のバージョンでは常に"on"です。
register_long_arrays boolean
PHP が、$HTTP_*_VARS のような古い長い 定義済みの変数 を登録するかどうかを指定します。 On (デフォルト) とした場合、$HTTP_GET_VARS のような 定義済みの長い PHP 変数が定義されます。 これらの変数を使用していない場合には、性能面からこのオプションを off とすることが推奨されています。 かわりに $_GET のようなスーパーグローバル変数を使用してください。 このディレクティブは PHP 5.0.0 で利用可能となりました。 その後 PHP 6.0.0 で削除されました。
post_max_size integer
POSTデータに許可される最大サイズを設定します。この設定は、ファ イルアップロードにも影響します。大きなファイルをアップロード するには、この値を upload_max_filesize より大きく設定する必要があります。 configureスクリプトでメモリ制限を有効とした場合、memory_limitもファイルアップロードに影響します。一般的に memory_limit は、 post_max_sizeよりも大きく する必要があります。 integerを使用する際、その値はバイト単位で測られます。 この FAQ に記載された短縮表記を使用することも可能です。 POSTデータの大きさが、post_max_sizeより大きい場合、 $_POST と $_FILES superglobals は空になります。この事象は、いくつかの方法で検出することができます。 例えば、$_GET 変数をデータを <form action="edit.php?processed=1">のように 処理するスクリプトに渡し、 $_GET['processed'] が設定されているかどうかを 確認する方法があります。
gpc_order string

GET/POST/COOKIE 変数処理の順番を設定します。この命令の デフォルトの設定は、"GPC"です。例えば、これを"GP"にPHPはクッキーを 完全に無視し、GETメソッド変数を同名のPOSTメソッド変数で上書き します。

注意: このオプションは、PHP 4では利用できません。 代わりに、variables_order を使用してください。

auto_prepend_file string

メインファイルの前に自動的に付加されるファイルの名前を指定します。 このファイルは、require() 関数のコール時と同様に読み込まれます。 このため、include_path が使用されます。

特別な値 **none** を指 定すると、ファイルを前に追加する機能は無効となります。

auto_append_file string

メインファイルの後に自動的に追加されるファイルの名前を指定します。 このファイルは、require() 関数のコール時と同様に読み込まれます。 このため、include_path が使用されます。

none を値として 指定するとこの自動付加機能はオフとなります。

注意: スクリプトが exit() で終了する場合、 この自動付加機能は使用されません 。

default_mimetype string

default_charset string

4.0.0 以降、PHP は、デフォルトで常にContent-type:ヘッダで character encodingを出力するようになっています。charsetの送信 を無効にするには、これを空にしてください。

always_populate_raw_post_data boolean

常に $HTTP_RAW_POST_DATA にアクセス可能とします。 この変数には生の POST データが格納されています。指定しなかった場合は、 データの MIME 型が判別できない場合のみこの変数にアクセス可能となります。 しかし、生の POST データにアクセスする方法としては php://input のほうが推奨されます。 $HTTP_RAW_POST_DATA は、 enctype="multipart/form-data" の場合には無効です。

allow_webdav_methods boolean

PHPスクリプトの中で WebDAV http リクエスト (例：PROPFIND, PROPPATCH, MOVE, COPY, 等..) の処理を可能にします。 このディレクティブは、PHP 4.3.2 以降存在しません。 これらのリクエストのPOSTデータを取得したい場合、 always_populate_raw_post_data も同時に設定する必要があ ります。

magic_quotes_gpc, magic-quotes-runtime, および magic_quotes_sybase も参照ください。

パスおよびディレクトリ

パスおよびディレクトリ設定オプション

名前	デフォルト	変更の可否	変更履歴
include_path	".:/path/to/php/pear"	PHP_INI_ALL	
doc_root	NULL	PHP_INI_SYSTEM	
user_dir	NULL	PHP_INI_SYSTEM	
extension_dir	"/path/to/php"	PHP_INI_SYSTEM	
extension	NULL	*php.ini* のみ	
cgi.check_shebang_line	"1"	PHP_INI_SYSTEM	PHP 5.2.0 以降で利用可能。
cgi.fix_pathinfo	"1"	PHP_INI_SYSTEM	PHP 4.3.0 以降で利用可能。PHP 5.2.1 までは PHP_INI_ALL でした。
cgi.force_redirect	"1"	PHP_INI_SYSTEM	PHP 4.2.0 以降で利用可能。PHP 5.2.1 までは PHP_INI_ALL でした。
cgi.redirect_status_env	NULL	PHP_INI_SYSTEM	PHP 4.2.0 以降で利用可能。PHP 5.2.1 までは PHP_INI_ALL でした。
fastcgi.impersonate	"0"	PHP_INI_SYSTEM	PHP 4.3.0 以降で利用可能。PHP 5.2.1 までは PHP_INI_ALL でした。
cgi.rfc2616_headers	"0"	PHP_INI_ALL	PHP 4.3.0 以降で利用可能。

以下に設定ディレクティブに関する簡単な説明を示します。

include_path string

require(), include(), fopen(), file(), readfile() および file_get_contents() 関数がファイルを 探すディレクトリのリストを指定します。フォーマットは、システ ムの環境変数 PATHと同じです。つまり、UNIXでは コロンで、Windowsではセミコロンで区切ったディレクトリのリスト で指定します。

Example#1 Unix include_path

```
include_path=".:/php/includes"
```

Example#2 Windows include_path

```
include_path=".;c:¥php¥includes"
```

このディレクティブのデフォルト値は、.(カレ ントディレクトリのみ)です。

doc_root string

サーバーにおけるPHPの"ルートディレクトリ"です。この値は空で無 い場合のみ使用されます。PHPが セーフモード で設定されている場合には、 このディレクトリの外側にあるファイルは使用されません。 PHPがFORCE_REDIRECTを指定してコンパイルされていない場合、 (IIS以外の)Webサーバのもとでは CGIとしてPHPを実行する際には、 doc_rootを指定するべきです。 他の方法としては、後述の cgi.force_redirect 設定の使用があります。

user_dir string

PHPファイル用にユーザーのホームディレクトリ として使用する基本ディレクトリの名前。例えば、 public_html となります。

extension_dir string

動的にロード可能な拡張モジュールを置くディレクトリを 指定します。 enable_dl と dl() も参照ください。

extension string

PHP の開始時に、どの動的ロード可能な拡張モジュールをロードするかを指定します。

cgi.check_shebang_line boolean

CGI 版の PHP が、実行するスクリプトの先頭にある #! から始まる行 (shebang) をチェックするかどうかを指定します。 同じスクリプトをスタンドアロンと PHP CGI 経由の両方で使用したい場合などに、 この行が必要になるでしょう。このディレクティブを on にしておくと、 CGI 版の PHP はこの行の内容を読み飛ばすようになります。

cgi.fix_pathinfo boolean

本来の PATH_INFO/PATH_TRANSLATED サポートをCGIで提供します。 PHPの以前の動作は、 SCRIPT_FILENAMEにPATH_TRANSLATEDを設定するというもので、 PATH_INFOの定義を理解していませんんでした。 PATH_INFOに関する詳細については、cgiの仕様を参照してください。 このオプションを 1 にすることにより、 PHP CGIはこのパスを仕様にあうように修正します。 ゼロとすると、 PHPは以前と同様に動作します。デフォルトは、ゼロ

です。 PATH_TRANSLATEDではなく、SCRIPT_FILENAMEを使用するようにスクリプト を修正する必要があります。

cgi.force_redirect boolean

cgi.force_redirect は、ほとんどのWebサーバのもとで CGI として PHP を実行する際のセキュリティを確保するために必要です。 未定義のままの場合、PHPはデフォルトでこれを on にします。 これを off にする時は、自己責任 の下に 行なってください。

> **注意**: Windowsユーザ: IISでは安全にこれをoffにすることができ、 実際には、off にすることが「必要」です。 OmniHTTPD または Xitami を動作させるには、これを off にする「必要」 があります。

cgi.redirect_status_env string

cgi.force_redirect を on にし、Apache または Netscape (iPlanet) Webサーバのもとで実行していない場合、 実行を継続して良いかどうかをPHPが判断するために 環境変数の名前を設定する必要があるかもしれません。

> **注意**: この変数を設定することにより、セキュリティ上の問題を発生する 場合があります。行うことのリスクをまず把握してください。

fastcgi.impersonate string

IIS (または WINNT ベースの OS) のもとでの FastCGI は、 クライアントをコールする際にセキュリティトークンを 匿名化する機能をサポートしています。 これにより、IIS がリクエストを処理するセキュリティコンテキストを 定義できるようになります。Apacheのもとで実行される mod_fastcgi は現在 (2002/03/17) この機能をサポートしていません。 IIS のもとで実行するには、1 に設定してください。 デフォルトは 0 です。

cgi.rfc2616_headers int

PHP に、HTTP レスポンスを返す際に、どの形式のヘッダーを使うか 指示します。0 にセットした場合は、Apache やその他の web サーバで サポートされている Status: ヘッダーを送信します。このオプションの値を 1 にセットした場合は、PHP は » RFC 2616 の仕様に適合した形式のヘッダーを送信します。この意味がわからないときは、 0 のままにしておいてください。

ファイルアップロード

ファイルアップロード設定オプション

名前	デフォルト	変更の可否	変更履歴
file_uploads	"1"	PHP_INI_SYSTEM	PHP <= 4.2.3 では PHP_INI_ALL、PHP 4.0.3 以降で利用可能
upload_tmp_dir	NULL	PHP_INI_SYSTEM	
upload_max_filesize	"2M"	PHP_INI_PERDIR	PHP <= 4.2.3 では PHP_INI_ALL

以下に設定ディレクティブに関する簡単な説明を示します。

file_uploads boolean

HTTP ファイルアップロード を有効とするかどうか。 upload_max_filesize, upload_tmp_dir, post_max_size ディレクティブも参照ください。

integerを使用する際、その値はバイト単位で測られます。 この FAQ に記載された短縮表記を使用することも可能です。
upload_tmp_dir string

ファイルアップロード時にファイル保存に用いるテンポラリディレクトリ。 PHPの実行ユーザーが書きこみ可能である必要が あります。指定されない場合、PHPはシステムのデフォルト設定を 使用します。

upload_max_filesize integer

アップロードされるファイルの最大サイズ。

integerを使用する際、その値はバイト単位で測られます。 この FAQ に記載された短縮表記を使用することも可能です。

SQL全般

SQL全般の設定オプション

名前	デフォルト	変更の可否	変更履歴
sql.safe_mode	"0"	PHP_INI_SYSTEM	

以下に設定ディレクティブに関する簡単な説明を示します。

sql.safe_mode boolean

オンにすると、デフォルト値が指定されているデータベース接続関数は、 引数で指定された値よりもデフォルト値を優先して使用します。 デフォルト値については、関連するデータベースのドキュメントを参照ください。

デバッガ設定ディレクティブ

警告

PHP 3のみがデフォルトのデバッガを実装しています。 詳細については、PHP のデバッグを参照してください。

debugger.host string

デバッガにより使用されるホストのDNS名またはIPアドレス。

debugger.port string

デバッガにより使用されるポート番号。

debugger.enabled boolean

デバッガを有効にするかどうか。

サポートされるタイムゾーンのリスト

目次

これは、PHP がサポートしているタイムゾーンの完全なリストです。例えば date_default_timezone_set() などで使用します。

> **注意**: 最新バージョンのタイムゾーンデータベースは、PECL の » timezonedb からインストールできます。Windows ユーザ向けには、コンパイル済みの DLL » php_timezonedb.dll が PECL4Win サイトからダウンロードできます。

アフリカのタイムゾーンのリスト

アフリカ

Africa/Abidjan	Africa/Accra	Africa/Addis_Ababa	Africa/Algiers	Africa/Asmera
Africa/Bamako	Africa/Bangui	Africa/Banjul	Africa/Bissau	Africa/Blantyre
Africa/Brazzaville	Africa/Bujumbura	Africa/Cairo	Africa/Casablanca	Africa/Ceuta
Africa/Conakry	Africa/Dakar	Africa/Dar_es_Salaam	Africa/Djibouti	Africa/Douala
Africa/El_Aaiun	Africa/Freetown	Africa/Gaborone	Africa/Harare	Africa/Johannesburg
Africa/Kampala	Africa/Khartoum	Africa/Kigali	Africa/Kinshasa	Africa/Lagos
Africa/Libreville	Africa/Lome	Africa/Luanda	Africa/Lubumbashi	Africa/Lusaka
Africa/Malabo	Africa/Maputo	Africa/Maseru	Africa/Mbabane	Africa/Mogadishu
Africa/Monrovia	Africa/Nairobi	Africa/Ndjamena	Africa/Niamey	Africa/Nouakchott
Africa/Ouagadougou	Africa/Porto-Novo	Africa/Sao_Tome	Africa/Timbuktu	Africa/Tripoli
Africa/Tunis	Africa/Windhoek			

アメリカのタイムゾーンのリスト

アメリカ

America/Adak	America/Anchorage	America/Anguilla	America/Antigua	Amer
America/Argentina/Buenos_Aires	America/Argentina/Catamarca	America/Argentina/ComodRivadavia	America/Argentina/Cordoba	Amer
America/Argentina/La_Rioja	America/Argentina/Mendoza	America/Argentina/Rio_Gallegos	America/Argentina/San_Juan	Amer
America/Argentina/Ushuaia	America/Aruba	America/Asuncion	America/Atikokan	Amer
America/Bahia	America/Barbados	America/Belem	America/Belize	Amer
America/Boa_Vista	America/Bogota	America/Boise	America/Buenos_Aires	Amer
America/Campo_Grande	America/Cancun	America/Caracas	America/Catamarca	Amer
America/Cayman	America/Chicago	America/Chihuahua	America/Coral_Harbour	Amer
America/Costa_Rica	America/Cuiaba	America/Curacao	America/Danmarkshavn	Amer
America/Dawson_Creek	America/Denver	America/Detroit	America/Dominica	Amer
America/Eirunepe	America/El_Salvador	America/Ensenada	America/Fort_Wayne	Amer
America/Glace_Bay	America/Godthab	America/Goose_Bay	America/Grand_Turk	Amer
America/Guadeloupe	America/Guatemala	America/Guayaquil	America/Guyana	Amer
America/Havana	America/Hermosillo	America/Indiana/Indianapolis	America/Indiana/Knox	Amer
America/Indiana/Petersburg	America/Indiana/Vevay	America/Indiana/Vincennes	America/Indianapolis	Amer
America/Iqaluit	America/Jamaica	America/Jujuy	America/Juneau	Amer
America/Kentucky/Monticello	America/Knox_IN	America/La_Paz	America/Lima	Amer

America/Louisville	America/Maceio	America/Managua	America/Manaus	Amer
America/Mazatlan	America/Mendoza	America/Menominee	America/Merida	Amer
America/Miquelon	America/Moncton	America/Monterrey	America/Montevideo	Amer
America/Montserrat	America/Nassau	America/New_York	America/Nipigon	Amer
America/Noronha	America/North_Dakota/Center	America/North_Dakota/New_Salem	America/Panama	Amer
America/Paramaribo	America/Phoenix	America/Port-au-Prince	America/Port_of_Spain	Amer
America/Porto_Velho	America/Puerto_Rico	America/Rainy_River	America/Rankin_Inlet	Amer
America/Regina	America/Rio_Branco	America/Rosario	America/Santiago	Amer
America/Sao_Paulo	America/Scoresbysund	America/Shiprock	America/St_Johns	Amer
America/St_Lucia	America/St_Thomas	America/St_Vincent	America/Swift_Current	Amer
America/Thule	America/Thunder_Bay	America/Tijuana	America/Toronto	Amer
America/Vancouver	America/Virgin	America/Whitehorse	America/Winnipeg	Amer
America/Yellowknife				

南極のタイムゾーンのリスト

南極

Antarctica/Casey	Antarctica/Davis	Antarctica/DumontDUrville	Antarctica/Mawson	Antarctica/McMurdo
Antarctica/Palmer	Antarctica/Rothera	Antarctica/South_Pole	Antarctica/Syowa	Antarctica/Vostok

北極のタイムゾーンのリスト

北極

Arctic/Longyearbyen

アジアのタイムゾーンのリスト

アジア

Asia/Aden	Asia/Almaty	Asia/Amman	Asia/Anadyr	Asia/Aqtau
Asia/Aqtobe	Asia/Ashgabat	Asia/Ashkhabad	Asia/Baghdad	Asia/Bahrain
Asia/Baku	Asia/Bangkok	Asia/Beirut	Asia/Bishkek	Asia/Brunei
Asia/Calcutta	Asia/Choibalsan	Asia/Chongqing	Asia/Chungking	Asia/Colombo
Asia/Dacca	Asia/Damascus	Asia/Dhaka	Asia/Dili	Asia/Dubai
Asia/Dushanbe	Asia/Gaza	Asia/Harbin	Asia/Hong_Kong	Asia/Hovd
Asia/Irkutsk	Asia/Istanbul	Asia/Jakarta	Asia/Jayapura	Asia/Jerusalem
Asia/Kabul	Asia/Kamchatka	Asia/Karachi	Asia/Kashgar	Asia/Katmandu
Asia/Krasnoyarsk	Asia/Kuala_Lumpur	Asia/Kuching	Asia/Kuwait	Asia/Macao
Asia/Macau	Asia/Magadan	Asia/Makassar	Asia/Manila	Asia/Muscat
Asia/Nicosia	Asia/Novosibirsk	Asia/Omsk	Asia/Oral	Asia/Phnom_Penh
Asia/Pontianak	Asia/Pyongyang	Asia/Qatar	Asia/Qyzylorda	Asia/Rangoon
Asia/Riyadh	Asia/Saigon	Asia/Sakhalin	Asia/Samarkand	Asia/Seoul
Asia/Shanghai	Asia/Singapore	Asia/Taipei	Asia/Tashkent	Asia/Tbilisi
Asia/Tehran	Asia/Tel_Aviv	Asia/Thimbu	Asia/Thimphu	Asia/Tokyo
Asia/Ujung_Pandang	Asia/Ulaanbaatar	Asia/Ulan_Bator	Asia/Urumqi	Asia/Vientiane
Asia/Vladivostok	Asia/Yakutsk	Asia/Yekaterinburg	Asia/Yerevan	

大西洋のタイムゾーンのリスト

大西洋

Atlantic/Azores	Atlantic/Bermuda	Atlantic/Canary	Atlantic/Cape_Verde	Atlantic/Faeroe

Atlantic/Jan_Mayen	Atlantic/Madeira	Atlantic/Reykjavik	Atlantic/South_Georgia	Atlantic/St_Helena
Atlantic/Stanley				

オーストラリアのタイムゾーンのリスト

オーストラリア

Australia/ACT	Australia/Adelaide	Australia/Brisbane	Australia/Broken_Hill	Australia/Canberra
Australia/Currie	Australia/Darwin	Australia/Hobart	Australia/LHI	Australia/Lindeman
Australia/Lord_Howe	Australia/Melbourne	Australia/North	Australia/NSW	Australia/Perth
Australia/Queensland	Australia/South	Australia/Sydney	Australia/Tasmania	Australia/Victoria
Australia/West	Australia/Yancowinna			

ヨーロッパのタイムゾーンのリスト

ヨーロッパ

Europe/Amsterdam	Europe/Andorra	Europe/Athens	Europe/Belfast	Europe/Belgrade
Europe/Berlin	Europe/Bratislava	Europe/Brussels	Europe/Bucharest	Europe/Budapest
Europe/Chisinau	Europe/Copenhagen	Europe/Dublin	Europe/Gibraltar	Europe/Guernsey
Europe/Helsinki	Europe/Isle_of_Man	Europe/Istanbul	Europe/Jersey	Europe/Kaliningrad
Europe/Kiev	Europe/Lisbon	Europe/Ljubljana	Europe/London	Europe/Luxembourg
Europe/Madrid	Europe/Malta	Europe/Mariehamn	Europe/Minsk	Europe/Monaco
Europe/Moscow	Europe/Nicosia	Europe/Oslo	Europe/Paris	Europe/Prague
Europe/Riga	Europe/Rome	Europe/Samara	Europe/San_Marino	Europe/Sarajevo
Europe/Simferopol	Europe/Skopje	Europe/Sofia	Europe/Stockholm	Europe/Tallinn
Europe/Tirane	Europe/Tiraspol	Europe/Uzhgorod	Europe/Vaduz	Europe/Vatican
Europe/Vienna	Europe/Vilnius	Europe/Volgograd	Europe/Warsaw	Europe/Zagreb
Europe/Zaporozhye	Europe/Zurich			

インドのタイムゾーンのリスト

インド

Indian/Antananarivo	Indian/Chagos	Indian/Christmas	Indian/Cocos	Indian/Comoro
Indian/Kerguelen	Indian/Mahe	Indian/Maldives	Indian/Mauritius	Indian/Mayotte
Indian/Reunion				

太平洋のタイムゾーンのリスト

太平洋

Pacific/Apia	Pacific/Auckland	Pacific/Chatham	Pacific/Easter	Pacific/Efate
Pacific/Enderbury	Pacific/Fakaofo	Pacific/Fiji	Pacific/Funafuti	Pacific/Galapagos
Pacific/Gambier	Pacific/Guadalcanal	Pacific/Guam	Pacific/Honolulu	Pacific/Johnston
Pacific/Kiritimati	Pacific/Kosrae	Pacific/Kwajalein	Pacific/Majuro	Pacific/Marquesas
Pacific/Midway	Pacific/Nauru	Pacific/Niue	Pacific/Norfolk	Pacific/Noumea
Pacific/Pago_Pago	Pacific/Palau	Pacific/Pitcairn	Pacific/Ponape	Pacific/Port_Moresby
Pacific/Rarotonga	Pacific/Saipan	Pacific/Samoa	Pacific/Tahiti	Pacific/Tarawa
Pacific/Tongatapu	Pacific/Truk	Pacific/Wake	Pacific/Wallis	Pacific/Yap

その他のタイムゾーンのリスト

その他

Brazil/Acre	Brazil/DeNoronha	Brazil/East	Brazil/West	Canada/Atlantic
Canada/Central	Canada/East-Saskatchewan	Canada/Eastern	Canada/Mountain	Canada/Newfoundland
Canada/Pacific	Canada/Saskatchewan	Canada/Yukon	CET	Chile/Continental
Chile/EasterIsland	CST6CDT	Cuba	EET	Egypt
Eire	EST	EST5EDT	Etc/GMT	Etc/GMT+0
Etc/GMT+1	Etc/GMT+10	Etc/GMT+11	Etc/GMT+12	Etc/GMT+2
Etc/GMT+3	Etc/GMT+4	Etc/GMT+5	Etc/GMT+6	Etc/GMT+7
Etc/GMT+8	Etc/GMT+9	Etc/GMT-0	Etc/GMT-1	Etc/GMT-10
Etc/GMT-11	Etc/GMT-12	Etc/GMT-13	Etc/GMT-14	Etc/GMT-2
Etc/GMT-3	Etc/GMT-4	Etc/GMT-5	Etc/GMT-6	Etc/GMT-7
Etc/GMT-8	Etc/GMT-9	Etc/GMT0	Etc/Greenwich	Etc/UCT
Etc/Universal	Etc/UTC	Etc/Zulu	Factory	GB
GB-Eire	GMT	GMT+0	GMT-0	GMT0
Greenwich	Hongkong	HST	Iceland	Iran
Israel	Jamaica	Japan	Kwajalein	Libya
MET	Mexico/BajaNorte	Mexico/BajaSur	Mexico/General	MST
MST7MDT	Navajo	NZ	NZ-CHAT	Poland
Portugal	PRC	PST8PDT	ROC	ROK
Singapore	Turkey	UCT	Universal	US/Alaska
US/Aleutian	US/Arizona	US/Central	US/East-Indiana	US/Eastern
US/Hawaii	US/Indiana-Starke	US/Michigan	US/Mountain	US/Pacific
US/Pacific-New	US/Samoa	UTC	W-SU	WET
Zulu				

***注意**: ここに挙げられているタイムゾーンは使用しないでください。 これらは過去のバージョンとの互換性のためにのみ残されています。*

拡張モジュールの分類

目次

この付録では、PHP マニュアルに掲載されている 150 以上の拡張モジュールを分類します。

目的

基本 PHP 拡張

変数や型に関連する拡張

- *配列*
- *Classkit*
- *クラス/オブジェクト*
- *ctype*
- *Filter*
- *関数*
- *オブジェクトの集約*
- *変数操作*

テキスト処理

- *PCRE*
- *POSIX Regex*

- Strings
- xdiff

PHP の振る舞いを変更する

- APC
- APD
- bcompiler
- エラーとログ記録
- http
- PHP オプション/情報
- 出力制御
- runkit

セッション拡張

- Msession
- muscat
- セッション
- Session PgSQL

その他の基本拡張

- GeoIP
- JSON
- その他
- Parsekit
- SPL
- tidy
- Tokenizer
- URLs

データベース拡張

抽象化レイヤ

- dba
- dbx
- PDO
- SDO-DAS-Relational
- ODBC

ベンダ固有のデータベース拡張

- dBase
- DB++
- FrontBase
- filePro
- Firebird/InterBase
- ibm_db2
- Informix
- Ingres II
- MaxDB
- mSQL
- MS SQL Server
- MySQL
- mysqli
- OCI8

- Oracle
- OvrimosSQL
- Paradox
- MS SQL Server (PDO)
- Firebird/Interbase (PDO)
- IBM (PDO)
- Informix (PDO)
- MySQL (PDO)
- Oracle (PDO)
- ODBC and DB2 (PDO)
- PostgreSQL (PDO)
- SQLite (PDO)
- PostgreSQL
- SQLite
- Sybase

XML 操作

- DOM
- DOM XML
- libxml
- qtdom
- SCA
- SDO
- SDO DAS XML
- SimpleXML
- XML
- XMLReader
- XMLWriter
- XSL
- XSLT

Web サービス

- SOAP
- WDDX
- XML-RPC

クレジットカード処理

- Cybercash
- CyberMUT
- MCVE
- Verisign Payflow Pro

数学と暗号化

数学的な拡張

- BC math
- GMP
- Math
- Statistics

暗号化拡張

- クラック関数 (Crack)
- hash

- mcrypt
- mhash
- OpenSSL

自然言語と文字エンコーディングのサポート

- enchant
- FriBiDi
- gettext
- i18n
- iconv
- マルチバイト文字列
- Pspell
- Recode
- Unicode

ファイルシステムとプロセス制御

ファイルシステムに関連する拡張

- DIO 関数
- ディレクトリ
- fam
- Fileinfo
- ファイルシステム
- Mimetype
- xattr

プロセス処理拡張

- プログラム実行
- Expect
- PCNTL
- POSIX
- Semaphore
- shmop

サービスへのアクセス

メールに関連する拡張

- Cyrus IMAP
- IMAP
- Mail
- Mailparse
- vpopmail

認証サービス

- kadm5
- radius

その他のサービス

- CURL
- FTP
- Hyperwave
- Hyperwave API
- IRC Gateway
- Java

- LDAP
- Memcache
- mnoGoSearch
- gopher
- ネットワーク
- YP/NIS
- Lotus Notes
- SAM
- SNMP
- ソケット
- ssh2
- ストリーム
- SVN
- TCP Wrappers
- YAZ

圧縮に関連する拡張

- Bzip2
- LZF
- phar
- Rar
- Zip
- Zlib

カレンダーやイベントに関連する拡張

- カレンダー
- 日付/時刻
- MCAL

ユーティリティ拡張

非 Text の MIME 出力

- ClibPDF
- FDF
- gnupg
- haru
- Ming (flash)
- PDF
- PS
- RPMReader
- SWF
- swish

画像処理・作成

- Exif
- イメージ
- Imagick 画像ライブラリ

音声フォーマットの操作

- id3
- OGG/Vorbis
- openal

コマンドラインの拡張

- Ncurses
- Newt
- Readline

Windows 限定の拡張

- COM
- .NET
- Printer
- W32api
- win32ps
- win32service

サーバ固有の拡張

- Apache
- IIS Functions
- NSAPI

所属

コア拡張

これらは実際のところ拡張ではありません。 PHP のコアに組み込まれており、コンパイルオプションで無効にすることはできません。

- 配列
- クラス/オブジェクト
- 日付/時刻
- ディレクトリ
- エラーとログ記録
- プログラム実行
- ファイルシステム
- 関数
- i18n
- PHP オプション/情報
- Mail
- Math
- その他
- ネットワーク
- 出力制御
- POSIX Regex
- セッション
- ストリーム
- Strings
- Tokenizer
- Unicode
- URLs
- 変数操作

バンドルされている拡張

これらの拡張は PHP にバンドルされています。

- Apache
- BC math
- Bzip2

- カレンダー
- COM
- ctype
- CURL
- dba
- dBase
- DOM
- .NET
- Exif
- FrontBase
- FDF
- FTP
- gettext
- GMP
- hash
- Firebird/InterBase
- iconv
- Informix
- イメージ
- IMAP
- JSON
- LDAP
- libxml
- マルチバイト文字列
- mcrypt
- mhash
- Mimetype
- Ming (flash)
- mSQL
- MS SQL Server
- MySQL
- mysqli
- Ncurses
- NSAPI
- オブジェクトの集約
- OCI8
- OpenSSL
- PCNTL
- PCRE
- PDO
- MS SQL Server (PDO)
- Firebird/Interbase (PDO)
- MySQL (PDO)
- Oracle (PDO)
- ODBC and DB2 (PDO)
- PostgreSQL (PDO)
- SQLite (PDO)
- PostgreSQL
- POSIX
- Pspell

- Readline
- Recode
- Semaphore
- shmop
- SimpleXML
- SNMP
- SOAP
- ソケット
- SPL
- SQLite
- SVN
- Sybase
- tidy
- ODBC
- WDDX
- XML
- XMLReader
- XML-RPC
- XSL
- XSLT
- Zip
- Zlib

外部拡張

これらの拡張をコンパイルするには、外部ライブラリが必要となります。

- BBCode
- Bzip2
- ClibPDF
- クラック関数 (Crack)
- CURL
- dBase
- DB++
- dbx
- DOM
- DOM XML
- enchant
- fam
- FrontBase
- FDF
- Fileinfo
- FriBiDi
- GeoIP
- gettext
- GMP
- haru
- Hyperwave
- Hyperwave API
- i18n
- Firebird/InterBase
- ibm_db2

- Informix
- Imagick 画像ライブラリ
- IMAP
- Ingres II
- IRC Gateway
- Java
- LDAP
- libxml
- MaxDB
- MCAL
- mcrypt
- Memcache
- mhash
- Ming (flash)
- mnoGoSearch
- Msession
- mSQL
- MS SQL Server
- muscat
- MySQL
- mysqli
- Ncurses
- Newt
- OCI8
- OGG/Vorbis
- openal
- OpenSSL
- Oracle
- OvrimosSQL
- Paradox
- PDF
- MS SQL Server (PDO)
- Firebird/Interbase (PDO)
- IBM (PDO)
- Informix (PDO)
- MySQL (PDO)
- Oracle (PDO)
- ODBC and DB2 (PDO)
- PostgreSQL (PDO)
- Verisign Payflow Pro
- PostgreSQL
- PS
- Pspell
- qtdom
- radius
- Rar
- Readline
- Recode
- Session PgSQL
- SimpleXML

- SNMP
- SOAP
- ssh2
- SWF
- swish
- Sybase
- tidy
- Unicode
- ODBC
- vpopmail
- WDDX
- xattr
- xdiff
- XML
- XMLReader
- XSL
- XSLT
- YAZ

PECL 拡張

これらの拡張は » PECL にあります。 PECL には、この他にもまだ PHP マニュアルで文書化されていない拡張があります。

- APC
- APD
- BBCode
- bcompiler
- Classkit
- ClibPDF
- クラック関数 (Crack)
- Cybercash
- CyberMUT
- Cyrus IMAP
- DB++
- dbx
- DIO 関数
- DOM XML
- enchant
- Expect
- fam
- Fileinfo
- filePro
- Filter
- FriBiDi
- GeoIP
- gnupg
- haru
- hash
- http
- Hyperwave
- Hyperwave API
- ibm_db2

- id3
- IIS Functions
- Imagick 画像ライブラリ
- Ingres II
- IRC Gateway
- Java
- JSON
- kadm5
- LZF
- Mailparse
- MaxDB
- MCAL
- MCVE
- Memcache
- mnoGoSearch
- Msession
- muscat
- Ncurses
- gopher
- Newt
- YP/NIS
- Lotus Notes
- OGG/Vorbis
- openal
- Oracle
- OvrimosSQL
- Paradox
- Parsekit
- PDF
- PDO
- MS SQL Server (PDO)
- Firebird/Interbase (PDO)
- IBM (PDO)
- Informix (PDO)
- MySQL (PDO)
- Oracle (PDO)
- ODBC and DB2 (PDO)
- PostgreSQL (PDO)
- SQLite (PDO)
- Verisign Payflow Pro
- phar
- Printer
- PS
- qtdom
- radius
- Rar
- RPMReader
- runkit
- SAM
- SCA

- SDO
- SDO_DAS_XML
- SDO-DAS-Relational
- Session_PgSQL
- ssh2
- Statistics
- SVN
- SWF
- swish
- TCP_Wrappers
- tidy
- vpopmail
- W32api
- win32ps
- win32service
- xattr
- xdiff
- XMLReader
- XMLWriter
- YAZ
- Zip

状態

このパートでは、実用向けではない拡張を挙げます。- これらは "古すぎる"（廃止予定 / deprecated）あるいは "新しすぎる"（実験的 / experimental）のどちらかです。

廃止予定の拡張

これらの拡張は非推奨です。通常は同じ機能を持つ別の拡張が存在します。

- Aspell
- CCVS
- ClibPDF
- DBM
- Mimetype
- Oracle
- オブジェクトのオーバーロード
- Satellite
- SESAM

実験的な拡張

これらの拡張の振る舞いは、PHP の将来のリリースで（その関数名や文書化されているあらゆる内容を含めて）予告なく変更される可能性があります。自己責任のもとで利用してください。

- bcompiler
- DB++
- .NET
- haru
- hash
- Imagick 画像ライブラリ
- Java
- Ming (flash)
- muscat
- Ncurses

- Newt
- Lotus Notes
- オブジェクトの集約
- Paradox
- MS SQL Server (PDO)
- Firebird/Interbase (PDO)
- Oracle (PDO)
- PS
- qtdom
- SCA
- SDO-DAS-Relational
- swish
- vpopmail
- W32api
- XML-RPC

関数エイリアスのリスト

数はかなり少ないですが、PHPには複数の名前でコールされる関数があります。 いくつかの関数は単に二つの名前があり、実際の機能を持ちません。（例えばis_int()とis_integer() は完全に同じです） いくつかの場合、これら複数の名前のどちらかが好ましいとはされていません。 例えば、is_int() と is_integer() は等しく好ましいとされています。 しかし、APIの整理や他の理由により名前が変更された関数があり、古い名前が 下位互換性の維持のためだけに残されている場合があります。 エイリアスの使用は古かったり、名前が変更されていたりで移植性の 低いスクリプトとなるため推奨されません。このリストは、古いスクリプト から新しい構文に更新しようとする人の助けとなるために提供されるものです。

このリストは、PHP 4.0.6に基づきます。

エイリアス

エイリアス	元の関数	使用されている拡張モジュール
_	gettext()	Gettext
add	swfmovie_add()	Ming (flash)
add	swfsprite_add()	Ming (flash)
add_root	domxml_add_root()	DOM XML
addaction	swfbutton_addAction()	Ming (flash)
addcolor	swfdisplayitem_addColor()	Ming (flash)
addentry	swfgradient_addEntry()	Ming (flash)
addfill	swfshape_addfill()	Ming (flash)
addshape	swfbutton_addShape()	Ming (flash)
addstring	swftext_addString()	Ming (flash)
addstring	swftextfield_addString()	Ming (flash)
align	swftextfield_align()	Ming (flash)
attributes	domxml_attributes()	DOM XML
children	domxml_children()	DOM XML
chop	rtrim()	Base syntax
close	closedir()	Base syntax
com_get	com_propget()	COM
com_propset	com_propput()	COM
com_set	com_propput()	COM
cv_add	ccvs_add()	CCVS
cv_auth	ccvs_auth()	CCVS
cv_command	ccvs_command()	CCVS
cv_count	ccvs_count()	CCVS
cv_delete	ccvs_delete()	CCVS
cv_done	ccvs_done()	CCVS
cv_init	ccvs_init()	CCVS
cv_lookup	ccvs_lookup()	CCVS
cv_new	ccvs_new()	CCVS
cv_report	ccvs_report()	CCVS
cv_return	ccvs_return()	CCVS
cv_reverse	ccvs_reverse()	CCVS
cv_sale	ccvs_sale()	CCVS
cv_status	ccvs_status()	CCVS
cv_textvalue	ccvs_textvalue()	CCVS
cv_void	ccvs_void()	CCVS
die	exit()	Miscellaneous functions
dir	getdir()	Base syntax
diskfreespace	disk_free_space()	Filesystem
domxml_getattr	domxml_get_attribute()	DOM XML
domxml_setattr	domxml_set_attribute()	DOM XML
doubleval	floatval()	Base syntax
drawarc	swfshape_drawarc()	Ming (flash)
drawcircle	swfshape_drawcircle()	Ming (flash)
drawcubic	swfshape_drawcubic()	Ming (flash)
drawcubicto	swfshape_drawcubicto()	Ming (flash)
drawcurve	swfshape_drawcurve()	Ming (flash)
drawcurveto	swfshape_drawcurveto()	Ming (flash)
drawglyph	swfshape_drawglyph()	Ming (flash)
drawline	swfshape_drawline()	Ming (flash)

エイリアス	元の関数	使用されている拡張モジュール

予約語の一覧

目次

以下に PHP で定義済みの ID の一覧を示します。ここに示す ID はいずれも スクリプトの中で ID として使用することはできません。これらの一覧には キーワード、定義済みの変数、定数、クラス名が含まれています。これらの一覧は、 全てを網羅しているわけではありません。

キーワードのリスト

これらのキーワードは、PHP では特別な意味があります。これらのいくつかは 関数やメソッドのようなものを表し、いくつかは定数のようなものを表す、 といったようになっていますが、実際にはそうではありません。実際には、 これらは言語を構成するものです。以下のキーワードはいずれも定数、クラス名、 関数名として使用することはできません。 これらを変数名として使用することは一般的には可能ですが、 混乱を生じる可能性があります。

PHP のキーワード

and	or	xor	__FILE__	exception (PHP 5)	php_user_filter
__LINE__	array()	as	break	case	
class	const	continue	declare	default	
die()	do	echo()	else	elseif	
empty()	enddeclare	endfor	endforeach	endif	
endswitch	endwhile	eval()	exit()	extends	
for	foreach	function	global	if	
include()	include_once()	isset()	list()	new	
print()	require()	require_once()	return()	static	
switch	unset()	use	var	while	
__FUNCTION__	__CLASS__	__METHOD__	final (PHP 5 以降)	php_user_filter (PHP 5 以降)	
interface (PHP 5 以降)	implements (PHP 5 以降)	extends	public (PHP 5 以降)	private (PHP 5 以降)	
protected (PHP 5 以降)	abstract (PHP 5 以降)	clone (PHP 5 以降)	try (PHP 5 以降)	catch (PHP 5 以降)	
throw (PHP 5 以降)	cfunction (PHP 4 のみ)	old_function (PHP 4 のみ)	this (PHP 5 以降)	namespace (PHP 6 のみ)	
import (PHP 6 のみ)	goto (PHP 6 のみ)				

定義済の変数

PHP 4.1.0 以降、 外部から来る変数 を取得するのに推奨される方法は以下に述べるスーパーグローバルを 用いることです。それまでは、 register_globals または定義済みの PHP 配列($HTTP_*_VARS)に 依存していました。 PHP 5.0.0 以降、PHP の長い 定義済みの変数 配列は register_long_arrays ディレクティブにより無効にすることができます。

サーバ変数: $_SERVER

注意: 4.1.0 で導入されました。これ以前のバージョンでは、 $HTTP_SERVER_VARS を使用してください。

$_SERVER は、ヘッダ、パス、スクリプトの位置のような 情報を有する配列です。この配列のエントリは、Web サーバにより 生成されます。全ての Web サーバがこれら全てを提供する保障はありません。 サーバは、これらのいくつかを省略したり、この一覧にない他のものを 定義する可能性があります。これらの変数の多くは、 » CGI 1.1 specification で定義されています。したがって、これらについては定義されていることを 期待することができます。

これは、'スーパーグローバル(superglobal)'、または自動グローバル (automatic global)、変数です。これは、スクリプトの全てのスコープで利用可能であることを意味します。関数やメソッドの中からこの変数に アクセスする際に $HTTP_SERVER_VARS のように **global $_SERVER;** とする必要はありません。

$HTTP_SERVER_VARS の最初の情報は同じですが、 スーパーグローバルではありません (HTTP_SERVER_VARS と $_SERVER は異なる変数であり、PHP は異なる変数として処理を行うことに注意してください)。

register_globals ディレクティブを設定した場合、これらの変数は、スクリプトの グローバルスコープ、つまり配列 $_SERVER 及び $HTTP_SERVER_VARS 以外のグローバル変数として 利用可能となります。関連情報については、 register_globals の使用法 という名前のセキュリティに関する章を参照ください。 これらの各グローバル変数は、スーパーグローバルではありません。

以下の各要素のいくつかは $_SERVER に現れない可能性があります。PHP をコマンドラインで実行している場合には、 使用できるものは僅かであることに注意してください。

'PHP_SELF'
現在実行しているスクリプトのファイル名です。 ドキュメントルートから取得されます。 例えば、http://example.com/test.php/foo.bar というアドレス上にあるスクリプトでは $_SERVER['PHP_SELF'] は /test.php/foo.bar となります。 __FILE__ 定数 には、カレント(すなわち読み込まれた)ファイルのパスとファイル名が含まれます。 PHP がコマンドラインから実行される場合、PHP 4.3.0 以降、 この変数にはスクリプト名が含まれます。これより前のバージョンでは、 この変数は使用できません。
'argv'
スクリプトに渡された引数の配列です。スクリプトがコマンドラインから 実行された場合、C 言語スタイルでコマンドライン引数に アクセスすることができます。GET メソッドを通してコールされた場合には 検索引数が格納されます。
'argc'
スクリプトに渡された引数の数が格納されます （コマンドライン上で実行された場合）。
'GATEWAY_INTERFACE'
サーバが使用している CGI のバージョンです。 例 'CGI/1.1'
'SERVER_ADDR'
現在のスクリプトが実行されているサーバの IP アドレスです。
'SERVER_NAME'
現在のスクリプトが実行されているサーバのホスト名です。 スクリプトがバーチャルホスト上で実行されている場合は そのバーチャルホスト名となります。
'SERVER_SOFTWARE'
レスポンスヘッダ上に書かれている、 サーバの認識文字列です。
'SERVER_PROTOCOL'
ページがリクエストされた際のプロトコル名とバージョンです。 例.'HTTP/1.0'
'REQUEST_METHOD'
ページにアクセスする際に使用されたリクエストのメソッド名です。 'GET', 'HEAD', 'POST', 'PUT' など。

> **注意**: リクエストのメソッドが HEAD だった場合、 PHP スクリプトはヘッダを送信した後（言い換えれば、 出力バッファリングを行わずに全出力を処理した後）に終了します。

'REQUEST_TIME'
リクエストの開始時のタイムスタンプ。PHP 5.1.0 以降で利用可能。
'QUERY_STRING'
ページがアクセスされた際にもし検索引数があればそれが格納されます。
'DOCUMENT_ROOT'
現在実行されているスクリプトが存在するドキュメントルート ディレクトリです。サーバのコンフィグレーションファイルで 定義されています。
'HTTP_ACCEPT'
現在のリクエストの Accept: ヘッダがもしあれば その内容。
'HTTP_ACCEPT_CHARSET'
現在のリクエストの Accept-Charset: ヘッダが もしあればその内容。例: 'iso-8859-1,*,utf-8'
'HTTP_ACCEPT_ENCODING'
現在のリクエストに Accept-Encoding: ヘッダが もしあればその内容。例: 'gzip'
'HTTP_ACCEPT_LANGUAGE'
現在のリクエストに Accept-Language: ヘッダが もしあればその内容。例: 'en'
'HTTP_CONNECTION'
現在のリクエストに Connection: ヘッダが もしあればその内容。例: 'Keep-Alive'
'HTTP_HOST'
現在のリクエストに Host: ヘッダが もしあればその内容。
'HTTP_REFERER'
現在のページに遷移する前にユーザエージェントが参照していた ページのアドレス（もしあれば）。これはユーザエージェントに よってセットされます。全てのユーザエージェントが これをセットしているわけではなく、また、HTTP_REFERER を変更する機能を持つものもあります。 要するに、信頼するべきものではありません。
'HTTP_USER_AGENT'
現在のリクエストに User-Agent: ヘッダが もしあればその内容。ページにアクセスしてきているユーザエージェント のしるしの文字列です。典型的な例は、 Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)。たとえば、 get_browser() でこの値を使って ページの出力をそのブラウザにあわせたものにすることも できるでしょう。
'HTTPS'
スクリプトが HTTPS プロトコルを通じて実行されている場合に 空でない値が設定されます。 ISAPI を IIS で使用している場合は、HTTPS プロトコルを通さないでリクエストが行われたときの値は off となることに注意しましょう。
'REMOTE_ADDR'
現在ページをみているユーザの IP アドレス。
'REMOTE_HOST'
現在のページにアクセスしているホスト名。DNS の逆引き検索は ユーザの REMOTE_ADDR に基づいています。

> **注意**: Web サーバがこの値を生成できるように設定されている必要があります。 例えば Apache の場合 HostnameLookups On が httpd.conf に設定されていなければこの値は生成されません。 gethostbyaddr() もご覧ください。

'REMOTE_PORT'
ユーザのマシンから Web サーバへの通信に使用されているポート番号
'SCRIPT_FILENAME'

現在実行されているスクリプトの絶対パス

> **注意**: file.php あるいは ../file.php のような相対パスを指定して CLI でスクリプトが実行されている場合、$_SERVER['SCRIPT_FILENAME'] には ユーザが指定した相対パスが含まれます。

'SERVER_ADMIN'
Web サーバの設定ファイルの SERVER_ADMIN (Apache の場合)ディレクティブ にセットされている値。スクリプトがバーチャルホスト上で 実行されている場合、バーチャルホストに対して値が定義されます。
'SERVER_PORT'
Web サーバの通信ポートとして使用されているポート番号。デフォルトでは '80' ですが、例えば SSL を使用している場合は セキュア HTTP ポートとして設定されている値に変わります。
'SERVER_SIGNATURE'
サーバ上で生成されたページに追加される、 サーバのバージョン名とバーチャルホスト名の文字列。 Web サーバの設定で有効になっていることが必要です。
'PATH_TRANSLATED'
バーチャルからリアルへのマッピングがなされた後の、 現在のスクリプトのファイルシステム上（ドキュメントルートではなく） でのパス。

> **注意**: PHP 4.3.2 以降、PATH_TRANSLATED は、 Apache 2 SAPI において暗黙のうちに設定されなく なりました。一方、Apache 1 では、この値が Apache により設定されない場合、 SCRIPT_FILENAME と同じ値に設定されます。 この変更は、PATH_TRANSLATED は PATH_INFO が定義されている場合のみ 存在するべきであるという CGI の規約を満たすために 行われました。 Apache 2 ユーザは、PATH_INFO を定義するために httpd.conf の中で AcceptPathInfo = On を使用することが可能です。

'SCRIPT_NAME'
現在のスクリプトのパス。 スクリプト自身のページを指定するのに有用です。 __FILE__ 定数には、カレント(すなわち読み込まれた)ファイルのパスとファイル名が 含まれます。
'REQUEST_URI'
ページにアクセスするために指定された URI。例えば、 '/index.html'
'PHP_AUTH_DIGEST'
PHP を Apache のモジュールとして実行し、HTTP ダイジェスト認証を 行っている場合、クライアントから送られた 'Authorization' ヘッダの内容が設定されます（適切な認証処理を行うために利用します）。
'PHP_AUTH_USER'
PHP を Apache または IIS（PHP 5 での ISAPI）のモジュールとして 実行している場合に、HTTP 認証しているときにそのユーザ名がセットされます。

'PHP_AUTH_PW'
PHP を Apache または IIS (PHP 5 での ISAPI) のモジュールとして 実行している場合に、HTTP 認証しているときにそのユーザの パスワードがセットされます。
'AUTH_TYPE'
PHP を Apache のモジュールとして実行している場合に、 HTTP 認証しているときにその認証形式がセットされます。

環境変数: $_ENV

注意: 4.1.0 で導入されました。これ以前のバージョンの場合は、 $HTTP_ENV_VARS を使用してください。

これらの変数は PHP パーサが実行されている環境から PHP のグローバル名前空間に取り込まれます。 その多くは、PHP が実行されているシェルに由来するものであり、 システムが違えばシェルも違ってくるため、確定的なリストを 得ることは不可能です。定義されている環境変数のリストについては 使用しているシェルのドキュメントをご覧ください。

PHP がサーバモジュールとして実行されているか CGI プロセッサとして 実行されているかに関わらず、その他の環境変数は CGI 変数を含みます。

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 $HTTP_ENV_VARS を使うときそうするように、 この変数に関数やメソッドの中からアクセスする際に **global $_ENV;** のようにする必要はありません。

$HTTP_ENV_VARS は同じ情報を持っていますが、 これはスーパーグローバルではありません ($HTTP_ENV_VARS と $_ENV は違う変数であり、PHP はそれぞれ別に扱います)。

register_globals が オンにセットされている場合、これらの変数はグローバルスコープ で有効となります。例えば、$_ENV と $HTTP_ENV_VARS 配列は分けられます。 関連する情報として、 Using Register Globals というセキュリティの章をご覧ください。 個々のグローバルはスーパーグローバルではありません。

HTTPクッキー: $_COOKIE

注意: 4.1.0 で導入されました。これ以前のバージョンの場合は、 $HTTP_COOKIE_VARS を使用してください。

カレントのスクリプトから渡された HTTP クッキーの情報が 格納された連想配列。

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 $HTTP_COOKIE_VARS を使うときにそうするように、 この変数に関数やメソッドの中からアクセスする際に **global $_COOKIE;** のようにする必要はありません。

$HTTP_COOKIE_VARS は同じ情報を持っていますが、 これはスーパーグローバルではありません ($HTTP_COOKIE_VARS と $_COOKIE は違う変数であり、PHP はそれぞれ別に扱います)。

register_globals が オンにセットされている場合、これらの変数はグローバルスコープで 有効となります。例えば、$_COOKIE と $HTTP_COOKIE_VARS 配列は分けられます。 関連する情報として、 Using Register Globals というセキュリティの章をご覧ください。 個々のグローバルはスーパーグローバルではありません。

HTTP GET 変数: $_GET

注意: 4.1.0 で導入されました。これ以前のバージョンの場合は、 $HTTP_GET_VARS を使用してください。

カレントのスクリプトから HTTP GET を通して渡された情報が 格納された連想配列。自動的にどのスコープでもグローバルとなります。

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 $HTTP_GET_VARS を使うときにそうするように、 この変数に関数やメソッドの中からアクセスする際に **global $_GET;** のようにする必要はありません。

$HTTP_GET_VARS は同じ情報を持っていますが、 これはスーパーグローバルではありません (HTTP_GET_VARS と $_GET は違う変数であり、PHP はそれぞれ別に扱います)。

register_globals が オンにセットされている場合、これらの変数はグローバルスコープで 有効となります。例えば、$_GET と $HTTP_GET_VARS 配列は分けられます。 関連する情報として、 Using Register Globals というセキュリティの章をご覧ください。 個々のグローバルはスーパーグローバルではありません。

HTTP POST 変数: $_POST

注意: 4.1.0 で導入されました。これ以前のバージョンの場合は、 $HTTP_POST_VARS を使用してください。

カレントのスクリプトから HTTP POST を通して渡された情報が 格納された連想配列。自動的にどのスコープでもグローバルとなる。

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 $HTTP_POST_VARS を使うときにそうするように、 この変数に関数やメソッドの中からアクセスする際に **global $_POST;** のようにする必要はありません。

$HTTP_POST_VARS は同じ情報を持っていますが、 これはスーパーグローバルではありません (HTTP_POST_VARS と $_POST は違う変数であり、PHPはそれぞれ別に扱います)。

register_globals が オンにセットされている場合、これらの変数はグローバルスコープで 有効となります。例えば、$_POST と $HTTP_POST_VARS 配列は分けられます。 関連する情報として、 Using Register Globals というセキュリティの章をご覧ください。 個々のグローバルはスーパーグローバルではありません。

HTTP ファイルアップロード変数: $_FILES

注意: 4.1.0 で導入されました。これ以前のバージョンの場合は、 $HTTP_POST_FILES を使用してください。

カレントのスクリプトから HTTP POST メソッドを通してアップロードされた 情報が格納された連想配列。自動的にどのスコープでもグローバルとなります。

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 $HTTP_POST_FILES を使うときにそうするように、 この変数に関数やメソッドの中からアクセスする際に **global $_FILES;** のようにする必要はありません。

$HTTP_POST_FILES は同じ情報を持っていますが、 これはスーパーグローバルではありません ($HTTP_POST_FILES と $_FILES は異なる変数であり、PHP はこれらを異なる変数として扱うことに注意してください)。

register_globals が オンにセットされている場合、これらの変数はグローバルスコープで 有効となります。例えば、$_FILES と $HTTP_POST_FILES 配列は分けられます。 関連する情報として、 Using Register Globals というセキュリティの章をご覧ください。 個々のグローバルはスーパーグローバルではありません。

リクエスト変数: $_REQUEST

注意: 4.1.0 で導入されました。以前のバージョンには、同等な配列はありません。

$_GET, $_POST, $_COOKIE, $_FILES の内容を格納した連想配列

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 この変数に関数やメソッドの中からアクセスする際に **global $_FILES;** のようにする必要はありません。

register_globals が オンにセットされている場合、これらの変数はグローバルスコープでn 有効となります。例えば、$_REQUEST 配列は分けられます。 関連する情報として、 Using Register Globals というセキュリティの章をご覧ください。 個々のグローバルはスーパーグローバルではありません。

セッション変数: $_SESSION

注意: 4.1.0 で導入されました。これ以前のバージョンでは、 $HTTP_SESSION_VARS を使用してください。

現在のスクリプトで有効なセッション情報が格納された配列です。 使用法の詳細については セッション処理関数を ご覧ください。

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 $HTTP_SESSION_VARS を使うときにそうするように、 この変数に関数やメソッドの中からアクセスする際に **global $_SESSION;** のようにする必要はありません。

$HTTP_SESSION_VARS は同じ情報を格納していますが、 スーパーグローバルではありません (**$HTTP_SESSION_VARS** と $_SESSION は異なる変数であり、PHP はこれらを異なる変数として扱うことに注意してください)。

register_globals が オンにセットされている場合、これらの変数はグローバルスコープで 有効となります。例えば、$_SESSION と $HTTP_SESSION_VARS 配列は分けられます。 関連する情報として、 Using Register Globals というセキュリティの章をご覧ください。 個々のグローバルはスーパーグローバルではありません。

グローバル変数: $GLOBALS

注意: $GLOBALS は、PHP 3.0.0 以降で利用可能です。

カレントのスクリプト上でグローバルスコープで定義されている全ての 変数を格納した連想配列。変数の名前は配列のキー。

これは'スーパーグローバル'又は自動グローバルな変数です。 つまり、スクリプトの全てのスコープで有効な変数であるということです。 この変数に関数やメソッドの中からアクセスする際に **global $_GLOBALS;** のようにする必要はありません。

直近のエラーメッセージ: $php_errormsg

$php_errormsg は、PHP によって発せられた 最後のエラーメッセージのテキストを格納する変数です。 エラーが発生したスコープ内、かつ track_errors 設定オプションが オン（デフォルトはオフ）にセットされている場合にのみ有効です。

生の POST データ: $HTTP_RAW_POST_DATA

$HTTP_RAW_POST_DATA には生の POST データが格納されます。 always_populate_raw_post_data を参照ください。

HTTP レスポンスヘッダ: $http_response_header

配列 $http_response_header は get_headers() 関数の結果と同等です。 HTTP ラッパー を使用する時には、 $http_response_header に HTTP レスポンスヘッダが格納されます。

定義済のクラス

ここでは、標準で定義されているクラスをとりあげます。 さまざまな拡張モジュールで定義されるその他のクラスについては、 個々の拡張モジュールのリファレンスで述べられています。

標準で定義されているクラス

以下のクラスは、PHP ビルドに含まれる標準関数セットで定義されています。

Directory
このクラスから、 dir のインスタンスが生成される。
stdClass
__PHP_Incomplete_Class

PHP 5 以降で定義済みのクラス

以下に PHP 5.0.0 で導入されたその他の定義済みのクラスを示します。

exception
php_user_filter

定義済みの定数

コアの定義済みの定数

これらの定数は PHP のコアで定義済みの定数です。 PHP, Zend engine, SAPI モジュールも含みます。

PHP_VERSION (string)
PHP_OS (string)
PHP_SAPI (string)
PHP 4.2.0 以降で利用可能。 php_sapi_name() も参照ください。
PHP_EOL (string)
PHP 4.3.10 および PHP 5.0.2 以降で利用可能
PHP_INT_MAX (integer)
PHP 4.4.0 および PHP 5.0.5 以降で利用可能

PHP_INT_SIZE (integer)
PHP 4.4.0 および PHP 5.0.5 以降で利用可能
DEFAULT_INCLUDE_PATH (string)
PEAR_INSTALL_DIR (string)
PEAR_EXTENSION_DIR (string)
PHP_EXTENSION_DIR (string)
PHP_PREFIX (string)
PHP 4.3.0 以降で利用可能
PHP_BINDIR (string)
PHP_LIBDIR (string)
PHP_DATADIR (string)
PHP_SYSCONFDIR (string)
PHP_LOCALSTATEDIR (string)
PHP_CONFIG_FILE_PATH (string)
PHP_CONFIG_FILE_SCAN_DIR (string)
PHP_SHLIB_SUFFIX (string)
PHP 4.3.0 以降で利用可能
PHP_OUTPUT_HANDLER_START (integer)
PHP_OUTPUT_HANDLER_CONT (integer)
PHP_OUTPUT_HANDLER_END (integer)
E_ERROR (integer)
E_WARNING (integer)
E_PARSE (integer)
E_NOTICE (integer)
E_CORE_ERROR (integer)
E_CORE_WARNING (integer)
E_COMPILE_ERROR (integer)
E_COMPILE_WARNING (integer)
E_USER_ERROR (integer)
E_USER_WARNING (integer)
E_USER_NOTICE (integer)
E_ALL (integer)
E_STRICT (integer)
PHP 5.0.0 以降で利用可能
__COMPILER_HALT_OFFSET__ (integer)
PHP 5.1.0 以降で利用可能

マジック定数 も参照ください。

標準の定義済み定数

PHP上でデフォルトで定義されている定数です。

EXTR_OVERWRITE (integer)
EXTR_SKIP (integer)
EXTR_PREFIX_SAME (integer)
EXTR_PREFIX_ALL (integer)
EXTR_PREFIX_INVALID (integer)
EXTR_PREFIX_IF_EXISTS (integer)
EXTR_IF_EXISTS (integer)
SORT_ASC (integer)
SORT_DESC (integer)
SORT_REGULAR (integer)
SORT_NUMERIC (integer)
SORT_STRING (integer)
CASE_LOWER (integer)
CASE_UPPER (integer)
COUNT_NORMAL (integer)
COUNT_RECURSIVE (integer)
ASSERT_ACTIVE (integer)
ASSERT_CALLBACK (integer)
ASSERT_BAIL (integer)
ASSERT_WARNING (integer)
ASSERT_QUIET_EVAL (integer)
CONNECTION_ABORTED (integer)
CONNECTION_NORMAL (integer)
CONNECTION_TIMEOUT (integer)
INI_USER (integer)
INI_PERDIR (integer)
INI_SYSTEM (integer)
INI_ALL (integer)
M_E (float)
M_LOG2E (float)
M_LOG10E (float)
M_LN2 (float)
M_LN10 (float)
M_PI (float)
M_PI_2 (float)
M_PI_4 (float)
M_1_PI (float)
M_2_PI (float)
M_2_SQRTPI (float)
M_SQRT2 (float)
M_SQRT1_2 (float)
CRYPT_SALT_LENGTH (integer)
CRYPT_STD_DES (integer)
CRYPT_EXT_DES (integer)
CRYPT_MD5 (integer)
CRYPT_BLOWFISH (integer)
DIRECTORY_SEPARATOR (string)
SEEK_SET (integer)
SEEK_CUR (integer)
SEEK_END (integer)
LOCK_SH (integer)
LOCK_EX (integer)
LOCK_UN (integer)
LOCK_NB (integer)
HTML_SPECIALCHARS (integer)
HTML_ENTITIES (integer)
ENT_COMPAT (integer)

ENT_QUOTES (integer)
ENT_NOQUOTES (integer)
INFO_GENERAL (integer)
INFO_CREDITS (integer)
INFO_CONFIGURATION (integer)
INFO_MODULES (integer)
INFO_ENVIRONMENT (integer)
INFO_VARIABLES (integer)
INFO_LICENSE (integer)
INFO_ALL (integer)
CREDITS_GROUP (integer)
CREDITS_GENERAL (integer)
CREDITS_SAPI (integer)
CREDITS_MODULES (integer)
CREDITS_DOCS (integer)
CREDITS_FULLPAGE (integer)
CREDITS_QA (integer)
CREDITS_ALL (integer)
STR_PAD_LEFT (integer)
STR_PAD_RIGHT (integer)
STR_PAD_BOTH (integer)
PATHINFO_DIRNAME (integer)
PATHINFO_BASENAME (integer)
PATHINFO_EXTENSION (integer)
PATH_SEPARATOR (string)
CHAR_MAX (integer)
LC_CTYPE (integer)
LC_NUMERIC (integer)
LC_TIME (integer)
LC_COLLATE (integer)
LC_MONETARY (integer)
LC_ALL (integer)
LC_MESSAGES (integer)
ABDAY_1 (integer)
ABDAY_2 (integer)
ABDAY_3 (integer)
ABDAY_4 (integer)
ABDAY_5 (integer)
ABDAY_6 (integer)
ABDAY_7 (integer)
DAY_1 (integer)
DAY_2 (integer)
DAY_3 (integer)
DAY_4 (integer)
DAY_5 (integer)
DAY_6 (integer)
DAY_7 (integer)
ABMON_1 (integer)
ABMON_2 (integer)
ABMON_3 (integer)
ABMON_4 (integer)
ABMON_5 (integer)
ABMON_6 (integer)
ABMON_7 (integer)
ABMON_8 (integer)
ABMON_9 (integer)
ABMON_10 (integer)
ABMON_11 (integer)
ABMON_12 (integer)
MON_1 (integer)
MON_2 (integer)
MON_3 (integer)
MON_4 (integer)
MON_5 (integer)
MON_6 (integer)
MON_7 (integer)
MON_8 (integer)
MON_9 (integer)
MON_10 (integer)
MON_11 (integer)
MON_12 (integer)
AM_STR (integer)
PM_STR (integer)
D_T_FMT (integer)
D_FMT (integer)
T_FMT (integer)
T_FMT_AMPM (integer)
ERA (integer)
ERA_YEAR (integer)
ERA_D_T_FMT (integer)
ERA_D_FMT (integer)
ERA_T_FMT (integer)
ALT_DIGITS (integer)
INT_CURR_SYMBOL (integer)
CURRENCY_SYMBOL (integer)
CRNCYSTR (integer)
MON_DECIMAL_POINT (integer)
MON_THOUSANDS_SEP (integer)
MON_GROUPING (integer)
POSITIVE_SIGN (integer)
NEGATIVE_SIGN (integer)
INT_FRAC_DIGITS (integer)
FRAC_DIGITS (integer)
P_CS_PRECEDES (integer)
P_SEP_BY_SPACE (integer)
N_CS_PRECEDES (integer)
N_SEP_BY_SPACE (integer)
P_SIGN_POSN (integer)
N_SIGN_POSN (integer)
DECIMAL_POINT (integer)
RADIXCHAR (integer)

THOUSANDS_SEP (integer)
THOUSEP (integer)
GROUPING (integer)
YESEXPR (integer)
NOEXPR (integer)
YESSTR (integer)
NOSTR (integer)
CODESET (integer)
LOG_EMERG (integer)
LOG_ALERT (integer)
LOG_CRIT (integer)
LOG_ERR (integer)
LOG_WARNING (integer)
LOG_NOTICE (integer)
LOG_INFO (integer)
LOG_DEBUG (integer)
LOG_KERN (integer)
LOG_USER (integer)
LOG_MAIL (integer)
LOG_DAEMON (integer)
LOG_AUTH (integer)
LOG_SYSLOG (integer)
LOG_LPR (integer)
LOG_NEWS (integer)
LOG_UUCP (integer)
LOG_CRON (integer)
LOG_AUTHPRIV (integer)
LOG_LOCAL0 (integer)
LOG_LOCAL1 (integer)
LOG_LOCAL2 (integer)
LOG_LOCAL3 (integer)
LOG_LOCAL4 (integer)
LOG_LOCAL5 (integer)
LOG_LOCAL6 (integer)
LOG_LOCAL7 (integer)
LOG_PID (integer)
LOG_CONS (integer)
LOG_ODELAY (integer)
LOG_NDELAY (integer)
LOG_NOWAIT (integer)
LOG_PERROR (integer)

リソース型の一覧

以下にPHPのリソースを生成、使用、破棄するための関数の一覧を示します。 is_resource()によりある変数がリソースであるかど うかを調べることが可能です。また、 get_resource_type()は変数のリソース型を返します。

Resource Types

リソース型の名前	生成する関数	使用する関数	破棄する関数	説明
aspell	aspell_new()	aspell_check(), aspell_check_raw(), aspell_suggest()	None	Aspell 辞書
bzip2	bzopen()	bzerrno(), bzerror(), bzerrstr(), bzflush(), bzread(), bzwrite()	bzclose()	Bzip2 ファイル
COM	com_load()	com_invoke(), com_propget(), com_get(), com_propput(), com_set(), com_propput()	None	COM オブジェクト リファレンス
VARIANT				
cpdf	cpdf_open()	cpdf_page_init(), cpdf_finalize_page(), cpdf_finalize(), cpdf_output_buffer(), cpdf_save_to_file(), cpdf_set_current_page(), cpdf_begin_text(), cpdf_end_text(), cpdf_show(), cpdf_show_xy(), cpdf_text(), cpdf_set_font(), cpdf_set_leading(), cpdf_set_text_rendering(), cpdf_set_horiz_scaling(), cpdf_set_text_rise(), cpdf_set_text_matrix(), cpdf_set_text_pos(), cpdf_set_text_pos(), cpdf_set_word_spacing(), cpdf_continue_text(), cpdf_stringwidth(), cpdf_save(), cpdf_translate(), cpdf_restore(), cpdf_scale(), cpdf_rotate(), cpdf_setflat(), cpdf_setlinejoin(), cpdf_setlinecap(), cpdf_setmiterlimit(), cpdf_setlinewidth(), cpdf_setdash(), cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto(), cpdf_lineto(), cpdf_rlineto(), cpdf_circle(), cpdf_arc(), cpdf_rect(), cpdf_closepath(), cpdf_stroke(), cpdf_closepath_fill_stroke(), cpdf_fill_stroke(), cpdf_clip(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray_stroke(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor(), cpdf_add_outline(), cpdf_set_page_animation(), cpdf_import_jpeg(), cpdf_place_inline_image(), cpdf_add_annotation()	cpdf_close()	CPDFライブラリのPDFドキュメント
cpdf outline				
curl	curl_copy_handle(), curl_init()	curl_copy_handle(), curl_errno(), curl_error(), curl_exec(), curl_getinfo(), curl_setopt()	curl_close()	Curl セッション
dbm	dbmopen()	dbmexists(), dbmfetch(), dbminsert(), dbmreplace(), dbmdelete(), dbmfirstkey(), dbmnextkey()	dbmclose()	DBMデータベースへのリンク
dba	dba_open()	dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()	dba_close()	DBAデータベースへのリンク
dba persistent	dba_popen()	dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()	None	DBAデータベースへの持続的なリンク

リソース型の名前	生成する関数	使用する関数	破棄する関数	説明

サポートされるプロトコル/ラッパー

目次

fopen() および copy() のような ファイルシステム関数で使用するために PHP に組み込まれた、URL 型式の プロトコルの一覧を示します。これらのラッパーに加え、PHP 4.3.0 以降では PHP スクリプトと stream_wrapper_register() によって ユーザ独自のラッパーを書くことができます。

ファイルシステム

PHP の全バージョン。PHP 5.0.0 以降では file:// を用いて明示的に指定します。

- /path/to/file.ext
- relative/path/to/file.ext
- fileInCwd.ext
- C:/path/to/winfile.ext
- C:¥path¥to¥winfile.ext
- ¥¥smbserver¥share¥path¥to¥winfile.ext
- file:///path/to/file.ext

PHP で使用されるデフォルトのラッパーは Filesystem で、 これはローカルファイルシステムを表します。 相対パス(/, \, \\,または Windows のドライブ文字で始まらないパス)が指定された場合、 指定されたパスは、現在の作業ディレクトリに対して適用されます。 多くの場合、これは、スクリプトがあるディレクトリです。ただし、カレントディレクトリが 変更されている場合を除きます。 CLI SAPI を使用した場合、このデフォルトはスクリプトがコールされたディレクトリとなります。

fopen() や file_get_contents() のようないくつかの関数については、相対パスと同時に include_path も検索されます。

ラッパーの概要

属性	サポートの有無
allow_url_fopen で制約される	No
読み込み許可	Yes
書き込み許可	Yes
追加許可	Yes
同時読み書き許可	Yes
stat() のサポート	Yes
unlink() のサポート	Yes
rename() のサポート	Yes
mkdir() のサポート	Yes
rmdir() のサポート	Yes

ソケット

ここでは、ソケット越しに動作するラッパー すなわち tcp、http あるいは ftp でサポートされるオプションを扱います。

PHP 5.1.0 以降、ひとつのオプション bindto のみが サポートされます。これは PHP がネットワークにアクセスする際の IP アドレス (IPv4 あるいは IPv6 のどちらか) やポート番号を指定する際に使用されます。 その書式は ip:port (IP やポート番号に 0 を指定すると、それをシステムに自動的に設定させる ことができます) です。

> **注意**: FTP の通常の操作では 2 つのソケット接続を使用するので、 bindto オプションでポート番号を指定することができません。 そのため、FTP ラッパーでサポートされる唯一の書式は ip:0 となります。

Example#1 bindto オプションの使用例

```
<?php
// IP アドレス '192.168.0.100' でインターネットに接続する
$opts = array('socket' =>
            array('bindto' => '192.168.0.100:0'));
```

```
// IP アドレス '192.168.0.100' とポート番号 '7000' でインターネットに接続する
$opts = array('socket' =>
            array('bindto' => '192.168.0.100:7000'));

// ポート番号 '7000' でインターネットに接続する
$opts = array('socket' =>
            array('bindto' => '0:7000'));

// コンテキストを作成し…
$context = stream_context_create($opts);

// …そしてデータを取得するためにそれを使用する
echo file_get_contents('http://www.example.com', false, $context);

?>
```

HTTP と HTTPS

PHP 3, PHP 4, PHP 5。https:// は PHP 4.3.0以降。

- http://example.com
- http://example.com/file.php?var1=val1&var2=val2
- http://user:password@example.com
- https://example.com
- https://example.com/file.php?var1=val1&var2=val2
- https://user:password@example.com

HTTP 1.0 により HTTP GET メソッドを用いてファイル/リソースに読み込みのみの アクセスが可能です。仮想ホストにホスト名でアクセスするために、 Host: ヘッダが送信されます。 ini ファイルまたはストリームコンテキストによって user_agent 文字列を設定している場合、 それはリクエストの中にも含まれます。

警告

IIS のような、いくつかの標準に対応してない Web サーバは、PHP に警告を発生させるような手順でデータを送信します。このようなサーバを使用する場合は、 error_reporting を警告を発生しないレベルまで小さくする必要があります。 PHP 4.3.7 以降では、https:// ラッパーでストリームをオープンする際にバグがある IIS サーバソフトウエアを検出することができ、この警告を抑制することができます。あなたが ssl:// ソケットを作性するために fsockopen() を使用している場合、自らこの警告を検出し、抑制する必要があります。

PHP 4.0.5 以降、リダイレクトがサポートされています。これ以前のバージョンを 使用している場合、URL の末尾にスラッシュを付ける必要があります。 (全てのリダイレクトが処理された後に)ドキュメント取得元のリソースの URL を知ることが 重要な場合、ストリームから返された一連のレスポンスヘッダを処理する必要があります。

```
<?php
$url = 'http://www.example.com/redirecting_page.php';

$fp = fopen($url, 'r');

/* PHP 4.3.0 の場合、stream_get_meta_data() の代わりに
    $http_response_header を使用してください */
$meta_data = stream_get_meta_data($fp);
foreach($meta_data['wrapper_data'] as $response) {

  /* リダイレクトされているか? */
  if (substr(strtolower($response), 0, 10) == 'location: ') {
    /* $url をリダイレクト先に書き換える */
    $url = substr($response, 18);
  }

}

?>
```

Example#1 ページの取得および POST データの送信

```
<?php

$postdata = http_build_query(
     array(
         'var1' => 'some content',
         'var2' => 'doh'
     )
);

$opts = array('http' =>
     array(
         'method'  => 'POST',
         'header'  => 'Content-type: application/x-www-form-urlencoded',
         'content' => $postdata
     )
);

$context  = stream_context_create($opts);

$result = file_get_contents('http://example.com/submit.php', false, $context);

?>
```

ストリームにより、リソースの body にアクセスすることが できます。ヘッダは、$http_response_header 変数に保存されます。 PHP 4.3.0 以降は、ヘッダは stream_get_meta_data() により取得できます。

HTTP 接続は読み込みのみ可で、HTTP リソースにデータを書き込んだり ファイルをコピーしたりすることはできません。

注意: HTTPS は、PHP 4.3.0 以降で OpenSSL のサポートを組み込んだ場合にサポートされます。

ラッパーの概要

属性	サポートの可否
allow_url_fopen で制約される	Yes
読み込み可	Yes
書き込み可	No
追記可能	No
同時読み書き可	N/A
stat() をサポート	No
unlink() をサポート	No
rename() をサポート	No
mkdir() をサポート	No
rmdir() をサポート	No

コンテキストのオプション

名前	使用法	デフォルト
method	リモートサーバでサポートされる **GET**、 **POST** あるいはその他の HTTP メソッド。	**GET**
header	リクエストに付加されるヘッダ。ここで指定した値が （*User-agent:*、*Host:*、 および *Authentication:* などの）他の値を 上書きすることもあります。	
user_agent	User-Agent: ヘッダとして送信する値。上で説明した *header* オプションで user-agent が 定義されていない場合のみ使用されます。	*php.ini* での *user_agent* の設定
content	ヘッダの後に送信する追加データ。通常、POST または PUT リクエストの際に使用されます。	
proxy	プロキシサーバを示す URI（例: *tcp://proxy.example.com:5100*）。 （HTTP プロキシを使用した）HTTPS のプロキシ処理は、 PHP 5.1.0 以降でのみ動作します。	
request_fulluri	**TRUE** を指定すると、リクエストを生成する際に完全な URI （*GET http://www.example.com/path/to/file.html HTTP/1.0*） が用いられます。これは標準のリクエストフォーマットではありませんが、 このようなフォーマットを要求するプロキシサーバも存在します。	**FALSE**
max_redirects	リダイレクトをたどる最大数。値が *1* 以下の場合は リダイレクトをたどらないことを意味します。PHP 5.1.0 で追加されました。	20
protocol_version	HTTP プロトコルのバージョン。PHP 5.1.0 で追加されました。	1.0
timeout	読み込みタイムアウト秒数を float (例 *10.5*) で指定します。PHP 5.2.1 で追加されました。	default_socket_timeout

注意: **基盤となるソケットストリームのコンテキストオプション** これ以外のコンテキストオプションが 基盤となるトランスポート でサポートされています。 http:// ストリームの場合は、tcp:// のコンテキストオプションを参照ください。 https:// ストリームの場合は、ssl:// のコンテキストオプションを参照ください。

バージョン 5 より前では、HTTP リクエストで独自のヘッダを送信することができます。 これは INI 設定項目 user_agent を処理する際の副作用によるものです。 user_agent に何らかの正常な文字列（たとえばデフォルト設定の PHP/version など）を指定し、さらに続けて キャリッジリターン(\r) / ラインフィード(\n) を置いた後に任意のヘッダを記述します。 この方法は、PHP 4 以降のバージョンで動作します。

Example#2 HTTP リクエストで独自のヘッダを送信する

```
<?php
ini_set('user_agent', "PHP¥r¥nX-MyCustomHeader:  Foo");

$fp = fopen('http://www.example.com/index.php', 'r');
?>
```

送信されるリクエストは次のようになります。

```
GET  /index.php  HTTP/1.0
Host:  www.example.com
User-Agent:  PHP
X-MyCustomHeader:  Foo
```

FTP と FTPS

PHP 3、PHP 4、PHP 5。ftps:// は PHP 4.3.0 以降。

- ftp://example.com/pub/file.txt
- ftp://user:password@example.com/pub/file.txt
- ftps://example.com/pub/file.txt

- *ftps://user:password@example.com/pub/file.txt*

FTP 経由でのファイルの読み込みと新しいファイルの作成を許可します。 サーバがパッシブモードの FTP をサポートしていない場合、接続は失敗します。

読み込み用または書き込み用のどちらかでファイルをオープンすることが 可能ですが、それらを両方同時に指定することはできません。FTP サーバ上の 既存のファイルを書き込み用にオープンしようとした場合、もし コンテキストオプション overwrite が指定されていなければ 接続は失敗します。既存のファイルを FTP 越しに上書きしたい場合は、 コンテキストオプション overwrite を指定したうえで 書き込み用にファイルをオープンします。別の方法としては、 FTP 拡張モジュール を使用することも可能です。

> ***注意**: **追記** PHP 5.0.0 では、ftp:// URL ラッパー経由での ファイルの追記が可能となりました。それ以前のバージョンでは ftp:// 経由でのファイルの追記は失敗していました。*

ftps:// は PHP 4.3.0 から有効になりました。 これは ftp:// と同じですが、FTP サーバとの 安全な接続を確立しようと試みます。もしサーバが SSL をサポートして いなければ、通常の（暗号化されない）FTP を使用します。

> ***注意**: FTPS のサポートは PHP 4.3.0 から始まりました。これを使用するには OpenSSL サポートを含めて PHP をコンパイルする必要があります。*

ラッパーの概要

属性	PHP 4	PHP 5
allow_url_fopen で制約される	Yes	Yes
読み込み許可	Yes	Yes
書き込み許可	Yes（新規ファイルのみ）	Yes（新規ファイル あるいは 既存のファイルで *overwrite* を指定）
追加許可	No	Yes
同時読み書き許可	No	No
stat() のサポート	No	PHP 5.0.0 では filesize()、 filetype()、file_exists()、 is_file() および is_dir() のみ。 PHP 5.1.0 で filemtime()。
unlink() のサポート	No	Yes
rename() のサポート	No	Yes
mkdir() のサポート	No	Yes
rmdir() のサポート	No	Yes

コンテキストオプション（PHP 5.0.0 以降）

名前	使用法	デフォルト
overwrite	リモートサーバ上の既存のファイルに対する上書きを許可する。 書き込みモード（アップロード）時のみ有効。	**FALSE**（無効）
resume_pos	転送を開始するファイル内の位置。 読み込みモード（ダウンロード）時のみ有効。	0（ファイルの先頭）
proxy（PHP 5.1.0 以降）	FTP リクエストを、http プロキシサーバ経由で行う。 ファイルの読み込み操作にのみ適用される。 例: *tcp://squid.example.com:8000*	

> ***注意**: **基盤となるソケットストリームのコンテキストオプション** これ以外のコンテキストオプションが 基盤となるトランスポート でサポートされています。 ftp:// ストリームの場合は、tcp:// のコンテキストオプションを参照ください。 ftps:// ストリームの場合は、ssl:// のコンテキストオプションを参照ください。*

PHP 入出力ストリーム

- *php://stdin*
- *php://stdout*
- *php://stderr*
- *php://output*
- *php://input*
- *php://filter (PHP 5.0.0 以降で使用可能)*
- *php://memory (PHP 5.1.0 以降で使用可能)*
- *php://temp (PHP 5.1.0 以降で使用可能)*

*php://stdin、php://stdout および php://stderr は、PHP プロセスの 対応する入出力ストリームへのアクセスを許可します。 これらのストリームは複製されたファイル記述子を参照します。そのため、 php://stdin をオープンしたあとでそれを閉じたとしても、 識別子のコピーが閉じられるだけです。**STDIN** で参照される実際のストリームは何も影響を受けません。 PHP 5.2.1 より前のバージョンでは、これに関連する挙動にバグがあります。 これらのラッパーを使うのではなく、定数 **STDIN**, **STDOUT** および **STDERR** を使用することを推奨します。*

php://output は、 print() および echo() と同じ方法での出力バッファへの書き込みを許可します。

php://input は、POST の生データの読み込みを 許可します。これは $HTTP_RAW_POST_DATA に比べて メモリ消費量が少なく、特別な php.ini ディレクティブを設定する 必要がありません。php://input は、 enctype="multipart/form-data" に対しては 使用できません。

php://stdin および php://input は読み込み専用で、 php://stdout、 php://stderr および php://output は書き込み専用です。

php://filter は、フィルタアプリケーションが ストリームをオープンすることを許可するために設計されたメタラッパーです。 これは、readfile()、file() および file_get_contents() のようなオールインワンの ファイル関数とともに使用すると有用です。これらの関数には、コンテンツが 読み込まれる前にストリームにフィルタを適用する手段がありません。

php://filter の対象は、以下のように 'parameters' を 'path' の一部として保持します。

- /resource=<フィルタの対象となるストリーム> （必須）このパラメータは、 php://filter 指定の最後に存在し、フィルタリング したいストリームを指している必要があります。

```
<?php
/* これは単純に以下と同じです。
     readfile("http://www.example.com");
   なぜなら、実際のところ何のフィルタ処理も行われないからです。 */

readfile("php://filter/resource=http://www.example.com");
?>
```

- /read=<読み込みチェーンに適用するフィルタのリスト> （任意）このパラメータは 1 つ以上のフィルタ名を パラメータとしてとり、それらはパイプ文字 | で区切られます。

```
<?php
/* これは、www.example.com のすべての内容を
   大文字に変換して出力します。 */
readfile("php://filter/read=string.toupper/resource=http://www.example.com");

/* これは上の例と同じですが、それに加えて
   ROT13 エンコード処理を行います。 */
readfile("php://filter/read=string.toupper|string.rot13/resource=http://www.example.com");
?>
```

- /write=<書き込みチェーンに適用するフィルタのリスト> （任意）このパラメータは 1 つ以上のフィルタ名を パラメータとしてとり、それらはパイプ文字 | で区切られます。

```
<?php
/* これは、文字列 "Hello World"
   に対して rot13 フィルタを適用し、カレントディレクトリの
   example.txt に書き込みます。 */
file_put_contents("php://filter/write=string.rot13/resource=example.txt","Hello World");
?>
```

- /<両方のチェーンに適用するフィルタのリスト> （任意）read= あるいは write= の指定をされていないすべてのフィルタは、 読み込みチェーンと書き込みチェーンの両方に（適宜）適用されます。

php://memory ラッパーは、データをメモリに保存します。 php://temp も同様ですが、メモリの制限（デフォルトは 2 MB です）を超過した際にはテンポラリファイルを使用します。

php://temp ラッパーは、次の 'parameters' を 'path' の一部として受け付けます。

- /maxmemory:<バイト数>（任意） このパラメータは、メモリの制限（データをテンポラリファイルに移動する閾値）のデフォルト値を変更します。

```
<?php
$fiveMBs = 5 * 1024 * 1024;
$fp = fopen("php://temp/maxmemory:$fiveMBs", 'r+');

fputs($fp, "hello¥n");

// 先ほど書き込んだデータを読み込みます。
rewind($fp);
echo stream_get_contents($fp);
?>
```

ラッパーの概要（*php://filter* については、 フィルタされる側のラッパーの概要を参照します）

属性	サポートの有無
allow_url_fopen で制約される	No
allow_url_include で制約される	*php://input*、 *php://stdin*、 *php://memory* および *php://temp* のみ。
読み込み許可	*php://stdin*、 *php://input*、 *php://memory* および *php://temp* のみ。
書き込み許可	*php://stdout*、 *php://stderr*、 *php://output*、 *php://memory* および *php://temp* のみ。
追加許可	*php://stdout*、 *php://stderr*、 *php://output*、 *php://memory* および *php://temp* のみ (書き込みと同じ)。
同時読み書き許可	*php://memory* および *php://temp* のみ。
stat() のサポート	*php://memory* および *php://temp* のみ。
unlink() のサポート	No
rename() のサポート	No
mkdir() のサポート	No
rmdir() のサポート	No

圧縮ストリーム

zlib: PHP 4.0.4 - PHP 4.2.3 (fopencookie をサポートするシステムのみ)

compress.zlib:// および compress.bzip2:// PHP 4.3.0以降

- *zlib:*
- *compress.zlib://*
- *compress.bzip2://*

zlib: は gzopen() と同様に 動作しますが、このストリームは fread() および 他のファイルシステム関数と組み合わせて使用することができるところが 異なります。この機能ではファイル名に ':' 文字が含まれる曖昧さが あるため、PHP 4.3.0 以降では古い機能となっています。代わりに compress.zlib:// を使用してください。

compress.zlib:// および compress.bzip2:// は、それぞれ gzopen() および bzopen() と等価で、 fopencookie をサポートしないシステムの上でも動作します。

ラッパーの概要

属性	サポートの有無
allow_url_fopen で制約される	No
読み込み許可	Yes
書き込み許可	Yes
追加許可	Yes
同時読み書き許可	No
stat() のサポート	No, 圧縮されたファイルの状態を知るには、通常の *file://* ラッパーを使用します。
unlink() のサポート	No, 圧縮されたファイルを unlink するには、通常の *file://* ラッパーを使用します。
rename() のサポート	No
mkdir() のサポート	No
rmdir() のサポート	No

ZIP 拡張モジュール は zip: ラッパーを登録します。

データ (RFC 2397)

data: (» RFC 2397) ストリームラッパーは、 PHP 5.2.0 以降で使用可能です。

Example#1 data:// の内容の表示

```
<?php
// "I love PHP" と表示します
echo file_get_contents('data://text/plain;base64,SSBsb3ZlIFBIUAo=');
?>
```

Example#2 media type の取得

```
<?php
$fp   = fopen('data://text/plain;base64,', 'r');
$meta = stream_get_meta_data($fp);

// "text/plain" と表示します
echo $meta['mediatype'];
?>
```

ラッパーの概要

属性	サポートの有無
allow_url_fopen で制約される	No
allow_url_include で制約される	Yes
読み込み許可	Yes
書き込み許可	No
追加許可	No
同時読み書き許可	No
stat() のサポート	No
unlink() のサポート	No
rename() のサポート	No
mkdir() のサポート	No
rmdir() のサポート	No

SSH2

ssh2.shell:// ssh2.exec:// ssh2.tunnel:// ssh2.sftp:// ssh2.scp:// PHP 4.3.0 以降 (PECL)

- *ssh2.shell://user:pass@example.com:22/xterm*

- *ssh2.exec://user:pass@example.com:22/usr/local/bin/somecmd*
- *ssh2.tunnel://user:pass@example.com:22/192.168.0.1:14*
- *ssh2.sftp://user:pass@example.com:22/path/to/filename*

***注意**: このラッパーはデフォルトでは有効になっていません* *ssh2.*://* ラッパーを使用するには、 *» PECL* から *» SSH2* 拡張モジュールを インストールする必要があります。

ssh2 ラッパーでは、*URL* のホスト部分に接続リソースを渡すことで既にオープン している接続を再利用することが可能です。

***Example#1** アクティブな接続からストリームをオープンする*

```
<?php
$session  = ssh2_connect('example.com', 22);
ssh2_auth_pubkey_file($session, 'username', '/home/username/.ssh/id_rsa.pub',
                                            '/home/username/.ssh/id_rsa', 'secret');
$stream  = fopen("ssh2.tunnel://$session/remote.example.com:1234", 'r');
?>
```

ラッパーの概要

属性	ssh2.shell	ssh2.exec	ssh2.tunnel	ssh2.sftp	ssh2.scp
allow_url_fopen で制約される	Yes	Yes	Yes	Yes	Yes
読み込み許可	Yes	Yes	Yes	Yes	Yes
書き込み許可	Yes	Yes	Yes	Yes	No
追加許可	No	No	No	Yes（サーバがサポートしている場合）	No
同時読み書き許可	Yes	Yes	Yes	Yes	No
stat() のサポート	No	No	No	Yes	No
unlink() のサポート	No	No	No	Yes	No
rename() のサポート	No	No	No	Yes	No
mkdir() のサポート	No	No	No	Yes	No
rmdir() のサポート	No	No	No	Yes	No

コンテキストオプション

名前	使用法	デフォルト
session	再利用する接続済みの ssh2 リソース	
sftp	再利用する割り当て済みの sftp リソース	
methods	Key exchange, hostkey, cipher, compression, および MAC methods	
callbacks		
username	接続するユーザ名	
password	パスワード認証に使用するパスワード	
pubkey_file	認証に使用する公開鍵ファイル	
privkey_file	認証に使用する秘密鍵ファイル	
env	設定する環境変数の連想配列	
term	pty を割り当てる際の端末エミュレート方式	
term_width	pty を割り当てる際の端末の幅	
term_height	pty を割り当てる際の端末の高さ	
term_units	term_width および term_height の単位	**SSH2_TERM_UNIT_CHARS**

オーディオストリーム

ogg:// PHP 4.3.0 以降 (PECL)

- *ogg://soundfile.ogg*
- *ogg:///path/to/soundfile.ogg*
- *ogg://http://www.example.com/path/to/soundstream.ogg*

***注意**: このラッパーはデフォルトでは有効になっていません* *ogg://* ラッパーを使用するには、 *» PECL* から *» OGG/Vorbis* 拡張モジュールを インストールする必要があります。

ogg:// ラッパー経由で読み込みモードでオープンされた ファイルは、*OGG/Vorbis* コーデックでエンコードされた圧縮音声ファイルとして 扱われます。同様に、*ogg://* ラッパー経由で書き込みモード あるいは追記モードでオープンされたファイルは、圧縮音声データとして 書き込まれます。読み込みモードでオープンした *OGG/Vorbis* ファイルに対して *stream_get_meta_data()* を適用した場合、以下のような さまざまな情報を返します。 *vendor* タグ、*comments* 、 多くの *channels* 、サンプリングレート *rate* 、 および以下のパラメータで指定されるエンコーディングレート。 *bitrate_lower* 、*bitrate_upper* 、 *bitrate_nominal* 、*bitrate_window*

ラッパーの概要

属性	サポートの有無
allow_url_fopen で制約される	No
読み込み許可	Yes
書き込み許可	Yes
追加許可	Yes
同時読み書き許可	No
stat() のサポート	No
unlink() のサポート	No
rename() のサポート	No
mkdir() のサポート	No
rmdir() のサポート	No

コンテキストオプション

名前	使用法	デフォルト	モード
pcm_mode	読み込みの際に適用する PCM エンコーディング。以下のうちのひとつ。 **OGGVORBIS_PCM_U8**、 **OGGVORBIS_PCM_S8**、 **OGGVORBIS_PCM_U16_BE**、 **OGGVORBIS_PCM_S16_BE**、 **OGGVORBIS_PCM_U16_LE**、 および **OGGVORBIS_PCM_S16_LE** （8 ビットか 16 ビットか、符号付きか符号なしか、ビッグエンディアンかリトルエンディアンか）。	OGGVORBIS_PCM_S16_LE	Read
rate	入力データのサンプリングレート。Hz 単位。	44100	Write/Append
bitrate	整数値の場合、エンコードの際の固定ビットレート（16000 から 131072）。 浮動小数点値の場合、使用する可変ビットレート（-1.0 から 1.0）。	128000	Write/Append
channels	エンコードする音声チャネル数。一般には 1（モノラル）あるいは 2（ステレオ）。最大 16 まで。	2	Write/Append
comments	エンコード時にトラックヘッダに入れる文字列の配列。		Write/Append

対話的プロセスストリーム

expect:// PHP 4.3.0 以降 (PECL)

- *expect://command*

***注意**: このラッパはデフォルトでは有効となっていません。 expect:// ラッパを使用するには、 » PECL にある » Expect 拡張モジュールを インストールする必要があります。*

expect:// ラッパでオープンしたストリームを使用すると、 プロセスの標準入力・標準出力・標準エラー出力への PTY 経由でのアクセスが 可能となります。

ラッパの概要

属性	サポートの有無
allow_url_fopen で制約される	No
読み込み許可	Yes
書き込み許可	Yes
追加許可	Yes
同時読み書き許可	No
stat() のサポート	No
unlink() のサポート	No
rename() のサポート	No
mkdir() のサポート	No
rmdir() のサポート	No

利用できるフィルタのリスト

目次

以下は、stream_filter_append() で利用できる 組み込みフィルタのリストです。PHP のバージョンによっては このリスト以外のフィルタがあったり、このリストにあるフィルタが存在 しなかったりするかもしれません。

stream_filter_append() と stream_filter_prepend() のちょっとした違いに ついて説明します。 すべての PHP ストリームはその内部に小さな読み込みバッファ を持っており、ファイルシステムもしくは他のリソースから読み込まれた データはこのバッファに格納されます。このことで、データをより効率的に 扱うことができるのです。リソースがストリームの内部バッファに 取り込まれたら、アプリケーション側で準備ができているかどうかに かかわらずすぐにフィルタ処理が行われます。フィルタが append されていた場合、読み込みバッファに はいってきたデータはすぐにフィルタ処理されます。つまり、バッファの 存在をまったく意識せずに扱えるということです。しかし、もしフィルタが prepend されていた場合、読み込みバッファにデータが 入ってきた段階ではフィルタ処理されません。 リソースから次のデータブロックが読み込まれるまで、フィルタ処理を 待機するのです。

あなたが使っている PHP にどのようなフィルタが登録されているかを 知るには、stream_get_filters() を利用します。

文字列フィルタ

これらのフィルタは、まさしくその名が示すとおりの働きをし、PHP 組み込み の文字列処理関数と同じように動作します。これらのフィルタについての より詳しい情報は、対応する関数のマニュアルを参照してください。

string.rot13 (PHP 4.3.0 以降) このフィルタは、すべてのストリームデータに対して str_rot13() 関数を適用するのと同じ動作をします。

Example#1 string.rot13

```
<?php
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'string.rot13');
fwrite($fp, "This is a test.¥n");
/* 出力:      Guvf vf n grfg.    */
?>
```

string.toupper (PHP 5.0.0 以降) このフィルタは、すべてのストリームデータに対して strtoupper() 関数を適用するのと同じ動作をします。

Example#2 string.toupper

```
<?php
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'string.toupper');
fwrite($fp, "This is a test.¥n");
/* Outputs:   THIS IS A TEST.    */
?>
```

string.tolower (PHP 5.0.0 以降) このフィルタは、すべてのストリームデータに対して strtolower() 関数を適用するのと同じ動作をします。

Example#3 string.tolower

```
<?php
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'string.tolower');
fwrite($fp, "This is a test.¥n");
/* 出力:      this is a test.    */
?>
```

string.strip_tags (PHP 5.0.0 以降) このフィルタは、すべてのストリームデータに対して strip_tags() 関数を適用するのと同じ動作をします。 以下の 2 つのうちどちらかの形式でパラメータを渡すことができます。 ひとつは、strip_tags() 関数の第 2 パラメータと同じ 形式でタグを並べた文字列、もうひとつはタグ名の配列です。

Example#4 string.strip_tags

```
<?php
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'string.strip_tags', STREAM_FILTER_WRITE, "<b><i><u>");
fwrite($fp, "<b>bolded text</b> enlarged to a <h1>level 1 heading</h1>¥n");
fclose($fp);
/* 出力:      <b>bolded text</b> enlarged to a level 1 heading    */

$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'string.strip_tags', STREAM_FILTER_WRITE, array('b','i','u'));
fwrite($fp, "<b>bolded text</b> enlarged to a <h1>level 1 heading</h1>¥n");
fclose($fp);
/* 出力:      <b>bolded text</b> enlarged to a level 1 heading    */
?>
```

変換フィルタ

string.* フィルタと同様、convert.* フィルタもその名前と同じような 動作をします。変換フィルタは、PHP 5.0.0 で追加されました。 これらのフィルタについてのより詳しい情報は、対応する関数のマニュアルを 参照してください。

convert.base64-encode と convert.base64-decode このフィルタは、すべてのストリームデータに対してそれぞれ base64_encode() または base64_decode() 関数を適用するのと同じ動作をします。 convert.base64-encode は、パラメータを連想配列形式で 受け取ることができます。line-length が与えられた 場合、base64 出力は line-length 文字単位に 分割されます。分割されたデータは、line-break-chars で指定された文字列で区切って出力されます。 これらのパラメータは、base64_encode() を chunk_split() とともに利用した場合と同じ動作をします。

Example#1 convert.base64-encode と convert.base64-decode

```
<?php
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'convert.base64-encode');
fwrite($fp, "This is a test.¥n");
fclose($fp);
/* 出力:      VGhpcyBpcyBhIHRlc3QuCg==   */

$param = array('line-length' => 8, 'line-break-chars' => "¥r¥n");
```

```
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'convert.base64-encode', STREAM_FILTER_WRITE, $param);
fwrite($fp, "This is a test.¥n");
fclose($fp);
/* 出力:      VGhpcyBp
        :      cyBhIHRl
        :      c3QuCg==  */

$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'convert.base64-decode');
fwrite($fp, "VGhpcyBpcyBhIHRlc3QuCg==");
fclose($fp);
/* 出力:      This is a test.  */
?>
```

convert.quoted-printable-encode と convert.quoted-printable-decode これらのフィルタのうち、デコードフィルタはすべてのストリームデータに 対して quoted_printable_decode() 関数を適用するのと 同じ動作をします。 convert.quoted-printable-encode に対応する関数は存在しません。 convert.quoted-printable-encode は、パラメータを 連想配列形式で受け取ることができます。convert.base64-encode でサポートされているパラメータに加え、convert.quoted-printable-encode は binary と force-encode-first という 2 つの論理型パラメータをサポートしています。 convert.base64-decode は line-break-chars パラメータのみをサポートしており、 これは符号化されたデータを分割する際に用いられます。

Example#2 convert.quoted-printable-encode と convert.quoted-printable-decode

```
<?php
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'convert.quoted-printable-encode');
fwrite($fp, "This is a test.¥n");
/* 出力:     =This is a test.=0A  */
?>
```

圧縮フィルタ

圧縮ストリーム を用いれば ローカルファイルシステム上に gzip や bz2 と互換性のある圧縮ファイルを 作成することができます。しかし、これはネットワーク越しの圧縮機能を 持っておらず、また非圧縮ストリームを圧縮されたストリームに変換することも できません。その点、圧縮フィルタはどんなストリームリソースでもどんな場合 でも適用可能です。

> **注意**: 圧縮フィルタは、gzip のようなコマンドライン ユーティリティで使われるヘッダを生成しません。 これらのフィルタは、（ヘッダ部を除いた）データストリームの本体のみを 圧縮・展開するものです。

zlib.deflate（圧縮）と zlib.inflate（展開）は、 » RFC 1951 で述べられている圧縮方法を 実装したものです。deflate フィルタには、次の 3 つの パラメータを連想配列形式で渡すことができます: level は、圧縮の度合いを 1から9 までで表した ものです。数字が大きいほど圧縮後のサイズが小さくなりますが、そのぶん 処理時間が長くかかります。次の2つの値は特別な意味を持ちます。 0（一切圧縮しない）、そして -1（zlib のデフォルト設定 -- 現在は 6）。 window は、圧縮用ループバックウィンドウのサイズを （2 を底とする）対数で指定します。大きな値（15 -- つまり 32768 バイトまで 引き上げる）を指定するとメモリをふんだんに利用してより小さく圧縮されます。 一方、小さな値（9 -- つまり 512 バイトまで絞り込む）を指定すると、圧縮の 効率は落ちますがメモリの消費量を抑えられます。値を指定しなかった際の window の初期値は、現在 **15** です。 memory は、作業用の一時メモリをどの程度割り当てるかを 指定します。1（最小限）から 9（最大限）の間で指定できます。この値は 圧縮の速度のみに影響し、圧縮後のデータのサイズには影響しません。

> **注意**: level は一番よく使われるパラメータなので、このパラメータについては （配列形式ではなく）直接整数値として設定することも可能にしています。

zlib.* 圧縮フィルタが有効になるのは、PHP バージョン 5.1.0 で zlib サポートが有効な場合です。 また、» PECL から » zlib_filter パッケージをインストールすることで、バージョン 5.0.x でも利用可能になります。これらのフィルタは、PHP 4 では利用できません。

Example#1 zlib.deflate と zlib.inflate

```
<?php
$params = array('level' => 6, 'window' => 15, 'memory' => 9);

$original_text = "This is a test.¥nThis is only a test.¥nThis is not an important string.¥n";
echo "もとのテキストの長さは " . strlen($original_text) . " 文字です。\n";

$fp = fopen('test.deflated', 'w');
stream_filter_append($fp, 'zlib.deflate', STREAM_FILTER_WRITE, $params);
fwrite($fp, $original_text);
fclose($fp);

echo "圧縮後のファイルの大きさは " . filesize('test.deflated') . " バイトです。\n";
echo "もとのテキストは:\n";
/* readfile と zlib.inflate を利用し、メモリ上で展開します */
readfile('php://filter/zlib.inflate/resource=test.deflated');

/* 出力:

もとのテキストの長さは 70 文字です。
圧縮後のファイルの大きさは 56 バイトです。
もとのテキストは:
This is a test.
This is only a test.
This is not an important string.

 */
?>
```

Example#2 zlib.deflate のシンプルな例

```
<?php
$original_text = "This is a test.¥nThis is only a test.¥nThis is not an important string.¥n";
echo "もとのテキストの長さは " . strlen($original_text) . " 文字です。\n";

$fp = fopen('test.deflated', 'w');
/* この "6" は、パラメータ "level" が 6 であるということ */
stream_filter_append($fp, 'zlib.deflate', STREAM_FILTER_WRITE, 6);
fwrite($fp, $original_text);
fclose($fp);

echo "圧縮後のファイルの大きさは " . filesize('test.deflated') . " バイトです。\n";
```

```
/* 出力:

もとのテキストの長さは 70 文字です。
圧縮後のファイルの大きさは 56 バイトです。

  */
?>
```

bzip2.compress と bzip2.decompress は、上で示した zlib フィルタと同じような動作をします。 bzip2.compress フィルタには、次の 2 つの パラメータを連想配列形式で渡すことができます: blocks は 1 から 9 までの整数値を設定します。 これは一時領域として割り当てるメモリのサイズを 100kバイトブロックの 数で示したものです。 work も整数値で、0 から 250 までの値を設定します。 これは、通常の圧縮方法がうまくいかなかった際に、どのくらい再試行した後で 速度の遅い（ただしより確実な）方法に切りかえるかを示します。 このパラメータは圧縮の速度のみに影響します。圧縮後のデータのサイズや 圧縮時のメモリ使用量は変わりません。0 を指定した場合は、bzip ライブラリの 初期設定値が利用されます。 bzip2.decompress フィルタには 1 つのパラメータを 渡すことができます。このパラメータは普通に論理型の値として渡すことも できますし、連想配列形式で small という名前の 要素として渡すこともできます。 small を **TRUE** に設定した場合、bzip ライブラリは メモリ使用量をできるだけ抑えるようになり、その分、処理速度は遅くなります。

bzip2.* 圧縮フィルタが有効になるのは、PHP バージョン 5.1.0 で bz2 サポートが有効な場合です。 また、» PECL から » bz2_filter パッケージをインストールすることで、バージョン 5.0.x でも利用可能になります。これらのフィルタは、PHP 4 では利用できません。

Example#3 bzip2.compress と bzip2.decompress

```
<?php
$param = array('blocks' => 9, 'work' => 0);

echo "もとのファイルの大きさは " . filesize('LICENSE') . " バイトです。\n";

$fp = fopen('LICENSE.compressed', 'w');
stream_filter_append($fp, 'bzip2.compress', STREAM_FILTER_WRITE, $param);
fwrite($fp, file_get_contents('LICENSE'));
fclose($fp);

echo "圧縮後のファイルの大きさは " . filesize('LICENSE.compressed') . " バイトです。\n";

/* 出力:

もとのファイルの大きさは 3288 バイトです。
圧縮後のファイルの大きさは 1488 バイトです。

  */
?>
```

暗号化フィルタ

mcrypt.* と mdecrypt.* は、libmcrypt を用いた暗号化・復号化を行います。これらのフィルタは、 mcrypt 暗号化関数 で利用可能な アルゴリズムをサポートしており、mcrypt.ciphername という名前で利用できます。ciphername の部分は、 mcrypt_module_open() に渡すのと同じアルゴリズム名 です。また、以下の 5 つのパラメータが利用できます。

mcrypt フィルタのパラメータ

パラメータ名	必須？	初期値	値の例
mode	任意	cbc	cbc, cfb, ecb, nofb, ofb, stream
algorithms_dir	任意	ini_get('mcrypt.algorithms_dir')	アルゴリズムモジュールのある場所
modes_dir	任意	ini_get('mcrypt.modes_dir')	モードモジュールのある場所
iv	必須	N/A	通常は 8, 16, あるいは 32 バイトのバイナリデータ。暗号の形式に依存する
key	必須	N/A	通常は 8, 16, あるいは 32 バイトのバイナリデータ。暗号の形式に依存する

Example#1 3DES を用い、ファイルへの出力を暗号化する

```
<?php
$passphrase = 'My secret';

/* 可読形式のパスフレーズを、再現可能な
 * iv/key のペアに変換する
 */
$iv = substr(md5('iv'.$passphrase, true), 0, 8);
$key = substr(md5('pass1'.$passphrase, true) .
              md5('pass2'.$passphrase, true), 0, 24);
$opts = array('iv'=>$iv, 'key'=>$key);

$fp = fopen('secert-file.enc', 'wb');
stream_filter_append($fp, 'mcrypt.tripledes', STREAM_FILTER_WRITE, $opts);
fwrite($fp, 'Secret secret secret data');
fclose($fp);
?>
```

Example#2 暗号化されたファイルを読み込む

```
<?php
$passphrase = 'My secret';

/* 可読形式のパスフレーズを、再現可能な
 * iv/key のペアに変換する
 */
$iv = substr(md5('iv'.$passphrase, true), 0, 8);
$key = substr(md5('pass1'.$passphrase, true) .
              md5('pass2'.$passphrase, true), 0, 24);
$opts = array('iv'=>$iv, 'key'=>$key);

$fp = fopen('secert-file.enc', 'rb');
```

```
stream_filter_append($fp, 'mdecrypt.tripledes', STREAM_FILTER_WRITE, $opts);
$data = rtrim(stream_get_contents($fp));
fclose($fp);

echo $data;
?>
```

サポートされるソケットトランスポートのリスト

目次

以下は、PHP がビルトインで持っている fsockopen() や stream_socket_client() のような ストリームに基づくソケットトランスポートとともに使用できる URL 形式のソケットトランスポートのリストです。 これらの転送は ソケット拡張には 適用されません。

自分が使用している PHP のバージョンのトランスポートのリストを得るには stream_get_transports() を使用してください。

Internet ドメイン: TCP、UDP、SSL、および TLS

PHP 3、PHP 4、PHP 5。 ssl:// & tls:// PHP 4.3.0 以降 sslv2:// & sslv3:// PHP 5.0.2 以降

> **注意**: トランスポートが指定されなければ、tcp:// と仮定されます。

- 127.0.0.1
- fe80::1
- www.example.com
- tcp://127.0.0.1
- tcp://fe80::1
- tcp://www.example.com
- udp://www.example.com
- ssl://www.example.com
- sslv2://www.example.com
- sslv3://www.example.com
- tls://www.example.com

Internet ドメインソケットは、対象のアドレスに加えてポート番号を 受け付けます。fsockopen() の場合、これは 2 番目の パラメータとして指定するので、トランスポートの URL には影響を与えません。 しかし、stream_socket_client() および関連する関数では 伝統的な URL を使用します。この場合、ポート番号はトランスポート URL の 後にコロンで区切ってつなげます。

- tcp://127.0.0.1:80
- tcp://[fe80::1]:80
- tcp://www.example.com:80

> **注意**: **IPv6 数値アドレスとポート番号** IPv4 やホスト名形式の例では、ポート番号はアドレスやホスト名の直後に コロンでつながっていますが、上の 2 番目の例では IPv6 アドレスが 角括弧でかこまれて [fe80::1] となっています。 これは、IPv6 アドレスに使用されるコロンとポート番号を表す際のコロンを 区別するためです。

ssl:// および tls:// のトランスポート （PHP が openssl サポートを含めてコンパイルされている場合のみ有効）は、 tcp:// トランスポートに SSL 暗号化を含めた拡張です。 PHP 4.3.0 以降では OpenSSL サポートは PHP に静的に組み込まれている 必要があります。PHP 5.0.0 以降では静的に組み込むだけではなくモジュールとして コンパイルされていてもよくなりました。

ssl:// は SSL V2 での接続を試みます。 あるいはリモートホストの設定によっては SSL V3 での接続を試みます。 sslv2:// および sslv3:// は、SSL V2 と SSL V3 のどちらのプロトコルを 使用するかを明示的に指定します。

ssl:// および tls:// の トランスポート時のオプション（PHP 4.3.2 以降）

名前	使用法	デフォルト
verify_peer	**TRUE** あるいは **FALSE** 。 SSL サーバ証明書の検証を要求するかどうか。	**FALSE**
allow_self_signed	**TRUE** あるいは **FALSE** 。 自己証明の証明書を許可するかどうか。	**FALSE**
cafile	ローカルファイルシステム上の証明書ファイルの場所。 verify_peer オプションでリモートサーバとの 認証の際に使用する。	
capath	cafile が指定されていなかったりその場所にファイルが 見つからなかったりした場合、capath が指す ディレクトリを検索して認証ファイルを探します。capath は認証ファイルのディレクトリを正確に指している必要があります。	
local_cert	ファイルシステム上のローカル証明書ファイルのパス。 あなたの証明書とプライベートキーを含み、PEM エンコードされた ファイルである必要があります。オプションで、発行者の 認証チェーンを含めることも可能です。	
passphrase	local_cert ファイルをエンコードした際の パスフレーズ。	
CN_match	予期している一般名（CN）。PHP は限定されたワイルドカード検索を 行います。もし一般名がこれにマッチしなかった場合、接続の試行は 失敗します。	

> **注意**: ssl:// は https:// および ftps:// のラッパの 基盤となるものなので、ssl:// に適用可能なオプションは https:// および ftps:// にも 適用可能です。

Unix ドメイン: UNIX および UDG

unix:// PHP 3 以降、 udg:// PHP 5 以降

- unix:///tmp/mysock
- udg:///tmp/mysock

unix:// は、Unix ドメインを使用したソケットストリーム 接続へのアクセスを提供します。udg:// は、UDP を使用した別方式での Unix ドメインソケットのトランスポートを提供します。

Unix ドメインソケットは、Internet ドメインソケットと異なり ポート番号を受け付けません。fsockopen() の場合、 portno パラメータは 0 に設定します。

PHP 型の比較表

下記の表はPHPの型と 比較演算子 の振る舞いについて、緩やかな場合と厳密な場合の両方について 例を示しています。この付録はマニュアルの 型の相互変換 にも関連しています。種々のユーザーコメントと » BlueShoesの働きの おかげです。

この表を活用する前に、型とその意味について理解しておく必要があります。 例えば、"42"は文字列ですが 42は整数です。 **FALSE**はboolean ですが "false"は 文字列です。

> **注意**: HTMLフォームは整数、浮動小数点数、booleanを渡してはくれず、 文字列を渡します。文字が数値であるかどうか確認するには、is_numeric()を使うとよいでしょう。
>
> **注意**: $xが定義されていない状態で単に if ($x)としてしまうと**E_NOTICE** レベルのエラーが発行てしまいます。代わりに、empty()やisset()を使うかあるいは変数を初期化するように してください。

$x PHP関数での$xの比較

Expression	gettype()	empty()	is_null()	isset()	boolean : if($x)
$x = "";	string	TRUE	FALSE	TRUE	FALSE
$x = NULL	NULL	TRUE	TRUE	FALSE	FALSE
var $x;	NULL	TRUE	TRUE	FALSE	FALSE
$x is undefined	NULL	TRUE	TRUE	FALSE	FALSE
$x = array();	array	TRUE	FALSE	TRUE	FALSE
$x = false;	boolean	TRUE	FALSE	TRUE	FALSE
$x = true;	boolean	FALSE	FALSE	TRUE	TRUE
$x = 1;	integer	FALSE	FALSE	TRUE	TRUE
$x = 42;	integer	FALSE	FALSE	TRUE	TRUE
$x = 0;	integer	TRUE	FALSE	TRUE	FALSE
$x = -1;	integer	FALSE	FALSE	TRUE	TRUE
$x = "1";	string	FALSE	FALSE	TRUE	TRUE
$x = "0";	string	TRUE	FALSE	TRUE	FALSE
$x = "-1";	string	FALSE	FALSE	TRUE	TRUE
$x = "php";	string	FALSE	FALSE	TRUE	TRUE
$x = "true";	string	FALSE	FALSE	TRUE	TRUE
$x = "false";	string	FALSE	FALSE	TRUE	TRUE

==による緩やかな比較

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

===による厳密な比較

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
array()	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
"php"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

注意: PHP 3.0 に関する注意 *文字列 "0" は、PHP 3 では空の値とみなされませんでした。 この挙動は PHP 4 で変更され、現在は空の値とみなされます。*

パーサトークンの一覧

PHP 言語の種々の部分は、内部的に T_SR のように表されています。PHP は、 パーサエラーが発生した際に、これらの ID を "Parse error: unexpected T_SR, expecting ',' or ';' in script.php on line 10." のように出力します。

ここでは、T_SR が何を意味するのかを知っていることを仮定しています。 この対応が分からない方のために、以下にこれらの ID、PHP 構文、マニュアルでの 適当な参照先の一覧を示します。

トークン

トークン	構文	参照先
T_ABSTRACT	abstract	クラスの抽象化 (PHP 5.0.0 以降で使用可能)
T_AND_EQUAL	&=	代入演算子
T_ARRAY	array()	array(), array 構文
T_ARRAY_CAST	(array)	型キャスト
T_AS	as	foreach
T_BAD_CHARACTER		ASCII 32以下の全ての文字。\t (0x09), \n (0x0a) , \r (0x0d) は除く
T_BOOLEAN_AND	&&	論理演算子
T_BOOLEAN_OR	¦¦	論理演算子
T_BOOL_CAST	(bool) or (boolean)	型キャスト
T_BREAK	break;	break
T_CASE	case	switch
T_CATCH	catch	例外(exceptions) (PHP 5.0.0 以降で使用可能)
T_CHARACTER		
T_CLASS	class	クラスとオブジェクト
T_CLASS_C	__CLASS__	定数 (PHP 4.3.0 以降で使用可能)
T_CLONE	clone	クラスとオブジェクト (PHP 5.0.0 以降で使用可能)
T_CLOSE_TAG	?> or %>	
T_COMMENT	// or #, and /* */ in PHP 5	コメント
T_CONCAT_EQUAL	.=	代入演算子
T_CONST	const	
T_CONSTANT_ENCAPSED_STRING	"foo" or 'bar'	文字列構文
T_CONTINUE	continue	
T_CURLY_OPEN		
T_DEC	--	可算/減算演算子
T_DECLARE	declare	declare
T_DEFAULT	default	switch
T_DIV_EQUAL	/=	代入演算子
T_DNUMBER	0.12, etc	浮動小数点数
T_DOC_COMMENT	/** */	PHPDoc 形式のコメント (PHP 5.0.0 以降で使用可能)
T_DO	do	do..while
T_DOLLAR_OPEN_CURLY_BRACES	${	complex variable parsed syntax
T_DOUBLE_ARROW	=>	array 構文
T_DOUBLE_CAST	(real), (double) or (float)	型キャスト
T_DOUBLE_COLON	::	以下の **T_PAAMAYIM_NEKUDOTAYIM** を参照ください。
T_ECHO	echo	echo()
T_ELSE	else	else
T_ELSEIF	elseif	elseif
T_EMPTY	empty	empty()
T_ENCAPSED_AND_WHITESPACE		
T_ENDDECLARE	enddeclare	declare, 別の構文
T_ENDFOR	endfor	for, 別の構文
T_ENDFOREACH	endforeach	foreach, 別の構文
T_ENDIF	endif	if, 別の構文
T_ENDSWITCH	endswitch	switch, 別の構文
T_ENDWHILE	endwhile	while, 別の構文
T_END_HEREDOC		heredoc 構文
T_EVAL	eval()	eval()
T_EXIT	exit or die	exit(), die()

トークン	構文	参照先

token_name() も参照ください。

ユーザレベルでの命名の手引き

目次

これは、あなたが書く PHP コード中の識別子によりよい名前をつけるための手引きです。 グローバル名前空間のシンボルを作成するコードでシンボルの名前を決める際には、 以下のガイドラインを考慮することが重要です。これにより、 PHP の将来のバージョンとの衝突を避けることができます。

グローバル名前空間

グローバル名前空間に属する言語構造には、このようなものがあります。

- 関数
- クラス
- インターフェイス
- 定数（クラス定数以外）
- 関数/メソッド の外部で定義された変数

ルール

以下のリストは、PHP プロジェクトで新たな内部識別子を作成する際に どのような基準で名前を決めているのかを おおまかにまとめたものです。完全な規約は、公式の » コーディング規約 を参照ください。

- PHP はトップレベルの名前空間を所有していますが、 きちんとしたわかりやすい名前をつけるようにこころがけ、 衝突が起こらないようにしています。
- 関数名では、単語の間にアンダースコアを使用します。一方、 クラス名の場合はキャメルケースを使用します（昔からあるクラスや関数の中には、例外もあります）。
- PHP の拡張モジュールのグローバルシンボルには、 その拡張モジュールの名前を先頭につけます（過去には、この原則を守っていない例が大量にありました）。 たとえば次のようになります。
  - curl_close()
  - mysql_query()
  - PREG_SPLIT_DELIM_CAPTURE
  - new DOMDocument()
  - strpos()（過去に間違えた例）
  - new SplFileObject()
- しかし、イテレータや例外については、単純に最後に "Iterator" および "Exception" を追加するようにします。例えば次のようになります。
  - ArrayIterator
  - LogicException
- PHP では、__ で始まるシンボルを特殊なものとして予約済みです。 以下にあげるような文書化されている機能を使用する場合を除き、 __ で始まるシンボルを作成しないことを推奨します。
  - **__get()**
  - **__autoload()**

Tips

将来もそのまま使用できるようなコードを書くためにお勧めなのは、次の方法です。 グローバル名前空間に属するシンボル名には、 前または後ろに何らかの文字列を追加するようにします。 追加する文字は 3 文字から 4 文字程度のあまり一般的でない文字列にし、 それをアンダースコアで（前または後ろに）連結します。 他のユーザのコードとの衝突を避けるためにお勧めなのは、次の方法です。 まず、他のプロジェクトでどのような接頭辞（あるいは接尾辞）が使用されているかを調べます。そして、自分たちが選択した接頭辞（あるいは接尾辞）を周りに広めるようにします。 これらを満たす例は、次のようになります。

- MyPx_someFunc()
- Foo_Date
- $asdf_dbh

マニュアルについて

目次

フォーマット

PHP のマニュアルはいくつかのフォーマットで提供されています。 大まかに言うとフォーマットには2つの種類があります。オンラインで 読むことのできるフォーマットとダウンロード可能なパッケージです。

> **注意**: いくつかの出版社は印刷されたマニュアルを提供しています。私たち としてはそれらをお勧めすることは出来ません。何故ならそういった ものはすぐに古いものなってしまいがちだからです。

オンラインマニュアルは » PHP.net Web サイト や、いくつかのミラーサイトで読むことができます。このタイプの フォーマットのマニュアルを快適に読むためには、一番近い場所にある ミラーサイトを選択すると良いでしょう。オンライン版の PHP マニュアルには、現在二種類の CSS スタイルシートが提供されています。それぞれ、 ウェブでの閲覧に適したものと、プリンタでの印刷に適したものです。

オンラインマニュアルがオフライン版より優れている点は、 ユーザノートと 見たいマニュアルのページを速やかに入手できる » URL ショートカットが 統合されていることでしょう。不利な点は、当然ですがマニュアルを見るためには 常にオンラインでいる必要があることです。

オフラインフォーマットにはいくつかのタイプがあるので、使用している OS や読み方にあわせて最適なものを選択してください。これらの異なったフォーマットのドキュメントをどのように生成しているかという 情報については、この付録の 各種フォーマットを生成する方法の項を読んでください。

最もプラットフォーム依存性が少いマニュアルのフォーマットは HTML 版です。このマニュアルは、一つのファイルにまとめられたものと、 章ごとに分割されたファイルのパッケージ（その結果、総数は数千にもなっています)の両方があります。 これらの HTML 版は圧縮された状態で提供されているため、 アーカイブの中のファイルを取得するには解凍用ユーティリティ が必要です。

Windows 上では、Windows HTML Help で Windows HTML Help 版のマニュアルを使用すると非常に便利です。このバージョンは全文検索、完全な目次、 そしてブックマーク機能を提供します。多くの有名な Windows 上での PHP 開発環境ではこのバージョンのマニュアルを統合する機能を備えていて、 簡単にアクセスすることが出来ます。Linux デスクトップ用の CHM ビューワも 利用可能です。 » xCHM または » GnoCHM を参照してください。

» 拡張 CHM 版も使用可能です。 その更新頻度はより少なくなっていますが、より多くの追加機能があります。 これは、ヘルプページを構築する際に使用される技術のために Microsoft Windows でのみ動作します。

ユーザノートについて

ユーザノートは PHP マニュアルを作成していく上で重要な役割を 持っています。マニュアルの読者が実例、警告そしてさらに詳しい 説明をブラウザから提供できるようにすることで、そうした フィードバックをマニュアルの本文に取り込むことが出来るのです。 そして記事が本文に取り入れられるまでの間は、その内容をオンライン、 もしくはいくつかのオフラインフォーマットで参照できるでしょう。

ユーザノートが PHP のサイトに表示される際に、何らかの 査閲がされているわけではない、ということに注意してください。 従って、そこに書かれた内容、実例の質については保証 されません(とはいえ、マニュアルの本文そのものの 質が保証されない、ということではありません)。

> **注意**: 著作権の適用範囲として、ユーザノートは PHP マニュアルの一部と見なされます。 それゆえ、このドキュメンテーションのライセンス(現状は Open Publication License)が適用されます。 詳細は Manual's Copyright をご覧ください。

関数の定義(プロトタイプ)を読むには

各関数についてクイックリファレンスが記述されているので、 マニュアルを読み理解するノウハウは PHP の利用をより簡単にしてくれます。 例示やカットアンドペーストに頼るより、関数の定義（プロトタイプ）を 読む方法を覚えるべきです。やってみましょう:

> **注意**: **前提条件: 型の基本的な理解** PHP は型についてルーズな言語ですが、重要な意味があるので、 型の基本を理解することは重要です。

関数の定義には返り値が どんな型であるかが示されています。最初の例として、 strlen()の定義を考えてみましょう。

```
 strlen

(PHP 3, PHP 4, PHP 5)
 strlen -- 文字列の長さを得る

 説明
int strlen ( string $string )
```

指定した文字列の長さを返します

関数の定義の説明

Part	説明
strlen	関数の名前
(PHP 3, PHP 4, PHP 5)	strlen()は PHP 3 と PHP 4 の両方で使用できる
int	関数が戻す値の型。ここでは整数 (文字列の長さが数字で数えら得ている)
(string $string)	strlen() 関数の最初の（この場合は唯一の）引数が *string* という名前であり それは文字列である

上の関数定義は、一般的な書き方では以下のようにも書けます。

```
returned type    function name    ( parameter type    parameter name )
```

多くの関数は複数の引数を取ります。例えば in_array()。 このプロトタイプは次のようになります:

```
bool in_array ( mixed $needle, array $haystack [, bool $strict])
```

*これは何を意味しているのでしょう？ in_array() は boolean の値を返します。 成功した場合は **TRUE**(needle が haystack の中にある場合)、失敗した場合は **FALSE** (needle が haystack の中にない場合)です。最初の引数は needle と名づけられており、それは多くの違った 型をとることができます。その場合"mixed"と 呼ばれます。 needle (我々が探しているもの)はスカラー値 (文字列,整数, 又は float)でも、 配列でもかまいません。 haystack (対象を探す配列)は二番目の引数です。 三番目のオプションの引数は strict と名づけられています。全てのオプション引数は [ブラケット]の中に表現されます。 strict パラメータはデフォルトでは boolean の **FALSE** であるとマニュアルに述べられています。機能の詳細については 各関数のマニュアルをご覧ください。*

より複雑なバージョンとの関係を有する関数もあります。以下が html_entity_decode() の例です。

```
(PHP 4 >= 4.3.0, PHP 5)
```

これは、この関数は PHP 3 では利用できず、 PHP 4.3.0 以降のバージョンでのみ利用可能であることを意味します。

本マニュアルに記載された PHP のバージョン

このドキュメントには、現在のバージョンおよび将来のバージョンの PHP についての情報が含まれます。また、追加情報として、 PHP 3 に関する移行情報や互換性情報も含まれています。 異なるバージョンの PHP における、動作、引数、 戻り値の違い等についてはマニュアルのノートやインラインテキストに 記述されています。

一部に CVS バージョンの PHP に関するドキュメントも存在します。これは 最新の開発バージョンのことです。まだ公開されておらず、CVS バージョン 管理システムか » スナップショット を用いてのみ取得可能です。つまり、"available in CVS" とされている 機能は、一般的にはまだ使えないということです。 しかし、おそらく次期安定バージョンの PHP では利用可能になるでしょう。 CVS バージョンをダウンロードする方法は » anonymous CVS access page を参照ください。

リリースされていないバージョンの PHP (例えば PHP 6.0.0、 最新の安定版は 5.x.x)に関するドキュメントに出あうかもしれません。 ほとんどの場合、これはドキュメントのミスではありません。 現在の PHP リリースでは有効ではない説明もしばしば追加されるのです。 それらは将来のバージョンで有効になります。 通常、PHP ではメジャーリリースでのみ新しい機能の追加が行われ、 その他のバージョンではバグ修正のみが行われます。 A.B.C の形式でバージョンは記述され、メジャーリリースでは A または B が増加、 マイナーリリースでは C が増加します。 例えば、最新のリリースが PHP x.0.x である時に、PHP x.1.x で利用可能となる 機能が文書化されることはまれではありません。 また、このマニュアルは過去形ではなく現在形で書かれていることにも注意してください。

しばしば PHP マニュアルでは、PHP ディレクティブの"デフォルト値"の一覧が示されます。 これらの値は、php.ini-recommended ではなく、php.ini-dist に基いています。 また、これらの値は、PHP の最新版に基いています。 これらの値と変更の詳細については、 付録:PHPディレクティブを参照してください。

PHP に関する更なる情報を得るには

このマニュアルは一般的なプログラミングの解説を提供しよう としているわけではありません。従って、もしあなたが全く初めて プログラミングを行う、もしくはほとんど初心者である、と言う場合は このマニュアルだけを使って、PHP のプログラミングを習得するのは 難しいでしょう。そうした場合には初心者向けの本を探したほうが よいかもしれません。

PHP に関する様々な側面について活発に議論しているメーリングリストも 多数あります。もし問題にぶつかったら、これらのメーリングリストの 利用を考えてください。メーリングリストを含むサポート情報は » PHP.net サポートページ にあります。さらに» PHP.net リンクページには PHP 関連の記事や 掲示板、コードギャラリーを提供しているサイトの一覧があります。

ドキュメントの改善を手助けするには

ドキュメントを改善していくには 3 つの方法があります。

もし(どの言語でかかれたものでも)マニュアルに間違いが見つかったら » http://bugs.php.net/ にある bug system を使ってその問題を報告してください。バグの種別は "Documentation Problem"にしてください。マニュアルのフォーマットを 含め、ドキュメントに関する問題はすべてここから送信してください。

> ***注意**: bug system に助けを要求することでシステムを無駄に使用しないでください。 かわりに、たくさんの» サポートオプション の中のひとつを利用してください。*

各ページに注釈をつけることで、原文に実例、警告等の更なる説明を 追加することが出来ます。とはいえ、この注釈システムを使って バグレポートを送信しないでください。ユーザノートの詳細については、 About user notesの 項を読んでください。

PHP マニュアルは多くの言語に翻訳されています。英語とその他の外国語が わかる方なら、翻訳チームを手伝うことで PHP マニュアルの改善に協力できます。 新しく翻訳を始めたい場合や、現在翻訳されているバージョンを手伝えおうと 思ったら、» http://doc.php.net/php/dochowto/ を是非読んでください。

PHP ドキュメント作成プロジェクトの IRC チャネルには、マニュアルの 作者の多くが参加しています。irc.efnet.org の #php.doc で、PHP ドキュメントを改善する方法に ついて議論しましょう。

各種フォーマットを生成する方法

このマニュアルは、» DocBook XML DTD を用いて XML で記述されています。 また、» XSLT (Extensible Stylesheet Language Transformations) によって保守やフォーマットを行います。

XML をソースファイルのフォーマットとして 使用することで、一つのソースファイルを管理するだけで、 多くの出力形式を生成することができるようになっています。 オンライン版のマニュアルを作成する際には » xsltproc および » DocBook XSL Stylesheets を使用しています。 HTML と TeX 版で使用している ツールは » James Clark によって 書かれた » Jade と、 » Norman Walsh によって 書かれた » The Modular DocBook Stylesheets です。Windows HTML ヘルプを作成するためには » Microsoft HTML Help Workshop を使用しています。またちょっとした変換処理やフォーマットの 処理にはもちろん PHP を使っています。

PHP マニュアルは多くの言語とフォーマットで作成されています。詳しい 情報は » http://www.php.net/docs.php を参照ください。 このドキュメントの XML ソースコードは、CVS から のダウンロードしたり » http://cvs.php.net/ で閲覧したりできます。ドキュメントは phpdoc モジュールに格納されています。

翻訳

PHP マニュアルはさまざまな形式で読めるだけでなく、さまざまな言語で 読むことができます。マニュアルの文章はまず最初に英語で書かれ、 その後世界中のチームによってそれぞれの母国語に翻訳されます。 もし特定の関数や章がまだ翻訳されていない場合、マニュアル生成システム はその英語版を自動的に挿入します。

翻訳に携わる人は » http://cvs.php.net/ から アクセスできる XML ソースコードを母国語に 翻訳することから始めます。生成されたバージョン（HTML やプレーンテキストなど）を扱うことはありません。 XMLを人間が読める形式に変換する処理をしてくれるのが ビルドシステムです。

> **注意**: もしドキュメントをあなたの母国語に翻訳することを手伝いたいと思ったら、 phpdocメーリングリストに加入してドキュメント／翻訳チームと連絡を取ってください。 » phpdoc-subscribe@lists.php.net に空のメールを送るだけです。 メーリングリストのアドレスは phpdoc@lists.php.net です。 マニュアルの翻訳に関心があることをメッセージに書いてください。 誰かから返事をもらって新しい言語への翻訳をスタートする手助けをしてもらうか、 あるいは、その言語について既に活動している翻訳チームと連絡が取れます。

現在のところ以下の言語において全て又は一部のマニュアルが利用できます。 （ブラジルの）ポルトガル語、中国語（簡体字）、中国語（香港 広東語）、 中国語（繁体字）、チェコ語、オランダ語、フィンランド語、フランス語、 ドイツ語、ヘブライ語、ハンガリー語、イタリア語、日本語、韓国/朝鮮語、 ポーランド語、ルーマニア語、ロシア語、スロバキア語、スロベニア語、 スペイン語、スウェーデン語、そしてトルコ語。

全てここからダウンロードできるでしょう: » http://www.php.net/docs.php

オープン・パブリケーション・ライセンス

目次

v1.0, 8 June 1999

I. 変更の有無にかかわらず要求される条件

オープン・パブリケーションである著作物は、このライセンスの定める条件を 厳守し、加えて本契約書を複製物に掲載するか、あるいは複製物中で(作者か出版者がオプションを選択した場合はそれらと共に)本契約書について 言及する限り、その全体あるいは一部を複製、配布することができる。物理的、電子的な媒体は問わない。

本ライセンスへの言及としてふさわしい形式は以下の通り:

Copyright (c) <year> <作者か被指名人の名前>。この作品は オープン・パブリケーション・ライセンス(vX.Y かそれ以降)で 指定された条件と制約に従う限り配布することができる （ライセンスの最新版は現在のところ » http://www.opencontent.org/openpub/で入手可能である)。

文書の作者と出版者のどちらか、あるいは両方によって選択された オプションは、本契約書へ言及した直後に指定されなければならない (VI節を参照せよ)。 オープン・パブリケーション・ライセンスの下で利用が許可された作品は、 商業的に再配布することができる。 一般的な(紙媒体の)書籍形式で出版する場合には、原作者と原出版者について 列挙することが要求される。出版者と作者の名前を書籍の外側表面すべてに 載せなければならない。書籍の外側表面全体には、原出版者名を著作物の タイトルと同じ大きさで、かつそのタイトルの所有者であることが 分かるように載せなければならない。

II. 著作権

それぞれのオープン・パブリケーションの著作権はその作者か 被指名人が所有する。

III. 本ライセンスの適用範囲

以下の利用条件は、特に文書中で指定されない限り、すべての オープン・パブリケーション著作物で適用される。

オープン・パブリケーションである著作物かその一部を、同一の媒体上に 他の著作物やプログラムと一緒に集めただけならば、本契約書が 他の著作物にまで適用されることはない。しかし、そのような 集積著作物には、オープン・パブリケーションである作品を含んでいる という通知と、適切な著作権表示を含めなければならない。

ライセンスの可分性について。本ライセンスの一部がいかなる 司法的管轄においても施行できない場合であっても、 ライセンスの残りの部分は依然として有効である。

無保証について。オープン・パブリケーション著作物は「あるがまま」で 提供され、利用が許可される。商業適合性の保証や特定の目的への適合性、無侵害の保証など、いかなる種類の保証も、暗黙明示を問わず存在しない。

IV. 変更を加えた場合に要求される条件

翻訳、アンソロジー、編集物、文書の一部分など、 このライセンスによって保護された文書を変更したものに関しては、 以下の条件を満たさなければならない:

1. 変更されたものはそう分かるようになっていなければならない。
2. 変更を加えた人物が特定できなければならず、 変更した日時も記載しておかなければならない。
3. 原作者と原出版者への謝辞は、それが適切なものである限り、 学問の世界での引用慣行に従って保たれなければならない。
4. 変更されていない元の文書の場所が特定できなければならない。
5. 変更して出来た文書を推奨宣伝する目的で、 原作者の名前を許可なしに使ってはならない。

V. 好ましい慣行の勧め

本ライセンスで指定された必要条件に加え、 再配布者は以下の実行を強く推奨、要請される:

1. もしオープン・パブリケーション著作物を印刷物ないし CD-ROM として 配布するならば、作者に対して再配布したいという旨をあなたの原稿か媒体が確定稿になる少なくとも 30 日前までに電子メールで知らせ、 作者が更新された文書を提供する機会を与えること。文書に変更を 行ったならば、この通知でそれについて説明するべきである。
2. 相当量の変更(削除含む)は文書中で明白にマーク付けされるか、 文書の付録において説明されるべきである。
3. 最後に、これは本ライセンスが強制するわけではないが、 オープン・パブリケーションとして利用が許可された著作物の印刷物や CD-ROM の複写を、作者に無料で進呈するのは良い心がけである。

VI. 利用許諾のオプション(選択権)

オープン・パブリケーションとして利用が許可された文書の作者あるいは 出版者は、本ライセンスの複写か本契約書への言及に言葉を追加することにより、 ある種のオプションを選択することができる。これらのオプションは 本ライセンス本文の一部とみなされ、派生物のライセンス(かそれへの言及)にも 含められなければならない。

A. 作者が明確に許可を与えない限り、実質的に変更されたものの配布は 禁止する。「実質的な変更」とは、文書の意味内容の変更として定義され、形式や誤植の修正などは除く。

この権利を行使するには、「この文書を実質的に変更した場合、 その配布を著作権保有者の明確な許可なしに行うことを禁止する」 という一文を本ライセンスの複写ないし本ライセンスへの言及に追加する。

B. 著作権所有者から前もって許可を得ない限り、商業目的で この著作物か派生物の全体または一部分を一般的な(紙媒体の) 書籍形式で出版することを禁止する。

この権利を行使するには、「この著作物かその派生物の一般的な(紙媒体の) 書籍形式での配布は、著作権所有者から前もって許可を得ない限り禁止する」 という一文を本ライセンスの複写か本ライセンスへの言及に追加する。

関数一覧

関数一覧

apache_request_headers()
apache_reset_timeout()
apache_response_headers()
apache_setenv()
apc_add()
apc_cache_info()
apc_clear_cache()
apc_compile_file()
apc_define_constants()
apc_delete()
apc_fetch()
apc_load_constants()
apc_sma_info()
apc_store()
apd_breakpoint()
apd_callstack()
apd_clunk()
apd_continue()
apd_croak()
apd_dump_function_table()
apd_dump_persistent_resources()
apd_dump_regular_resources()
apd_echo()
apd_get_active_symbols()
apd_set_pprof_trace()
apd_set_session()
apd_set_session_trace()
apd_set_socket_session_trace()
array()
array_change_key_case()
array_chunk()
array_combine()
array_count_values()
array_diff()
array_diff_assoc()
array_diff_key()
array_diff_uassoc()
array_diff_ukey()
array_fill()
array_fill_keys()
array_filter()
array_flip()
array_intersect()
array_intersect_assoc()
array_intersect_key()
array_intersect_uassoc()
array_intersect_ukey()
array_key_exists()
array_keys()
array_map()
array_merge()
array_merge_recursive()
array_multisort()
array_pad()
array_pop()
array_product()
array_push()
array_rand()
array_reduce()
array_reverse()
array_search()
array_shift()
array_slice()
array_splice()
array_sum()
array_udiff()
array_udiff_assoc()
array_udiff_uassoc()
array_uintersect()
array_uintersect_assoc()
array_uintersect_uassoc()
array_unique()
array_unshift()
array_values()
array_walk()
array_walk_recursive()
ArrayIterator::current()
ArrayIterator::key()
ArrayIterator::next()
ArrayIterator::rewind()
ArrayIterator::seek()
ArrayIterator::valid()
ArrayObject::__construct()
ArrayObject::append()
ArrayObject::count()
ArrayObject::getIterator()
ArrayObject::offsetExists()
ArrayObject::offsetGet()
ArrayObject::offsetSet()
ArrayObject::offsetUnset()
arsort()
ascii2ebcdic()
asin()
asinh()
asort()
aspell_check()
aspell_check_raw()
aspell_new()
aspell_suggest()
assert()
assert_options()

dbplus_tcl()
dbplus_tremove()
dbplus_undo()
dbplus_undoprepare()
dbplus_unlockrel()
dbplus_unselect()
dbplus_update()
dbplus_xlockrel()
dbplus_xunlockrel()
dbx_close()
dbx_compare()
dbx_connect()
dbx_error()
dbx_escape_string()
dbx_fetch_row()
dbx_query()
dbx_sort()
dcgettext()
dcngettext()
deaggregate()
debug_backtrace()
debug_print_backtrace()
debug_zval_dump()
debugger_off()
debugger_on()
decbin()
dechex()
decoct()
define()
define_syslog_variables()
defined()
deg2rad()
delete()
dgettext()
die()
dio_close()
dio_fcntl()
dio_open()
dio_read()
dio_seek()
dio_stat()
dio_tcsetattr()
dio_truncate()
dio_write()
dir()
DirectoryIterator::__construct()
DirectoryIterator::current()
DirectoryIterator::getATime()
DirectoryIterator::getCTime()
DirectoryIterator::getFilename()
DirectoryIterator::getGroup()
DirectoryIterator::getInode()
DirectoryIterator::getMTime()
DirectoryIterator::getOwner()
DirectoryIterator::getPath()
DirectoryIterator::getPathname()
DirectoryIterator::getPerms()
DirectoryIterator::getSize()
DirectoryIterator::getType()
DirectoryIterator::isDir()
DirectoryIterator::isDot()
DirectoryIterator::isExecutable()
DirectoryIterator::isFile()
DirectoryIterator::isLink()
DirectoryIterator::isReadable()
DirectoryIterator::isWritable()
DirectoryIterator::key()
DirectoryIterator::next()
DirectoryIterator::rewind()
DirectoryIterator::valid()
dirname()
disk_free_space()
disk_total_space()
diskfreespace()
dl()
dngettext()
dns_check_record()
dns_get_mx()
dns_get_record()
dom_import_simplexml()
DOMAttr->__construct()()
DOMAttr->isId()()
DomAttribute->name()
DomAttribute->set_value()
DomAttribute->specified()
DomAttribute->value()
DOMCharacterData->appendData()()
DOMCharacterData->deleteData()()
DOMCharacterData->insertData()()
DOMCharacterData->replaceData()()
DOMCharacterData->substringData()()
DOMComment->__construct()()
DOMDocument->__construct()()
DomDocument->add_root()
DomDocument->create_attribute()
DomDocument->create_cdata_section()
DomDocument->create_comment()
DomDocument->create_element()
DomDocument->create_element_ns()
DomDocument->create_entity_reference()
DomDocument->create_processing_instruction()

DomDocument->create_text_node()
DOMDocument->createAttribute()()
DOMDocument->createAttributeNS()()
DOMDocument->createCDATASection()()
DOMDocument->createComment()()
DOMDocument->createDocumentFragment()()
DOMDocument->createElement()()
DOMDocument->createElementNS()()
DOMDocument->createEntityReference()()
DOMDocument->createProcessingInstruction()()
DOMDocument->createTextNode()()
DomDocument->doctype()
DomDocument->document_element()
DomDocument->dump_file()
DomDocument->dump_mem()
DomDocument->get_element_by_id()
DomDocument->get_elements_by_tagname()
DOMDocument->getElementById()()
DOMDocument->getElementsByTagName()()
DOMDocument->getElementsByTagNameNS()()
DomDocument->html_dump_mem()
DOMDocument->importNode()()
DOMDocument->load()()
DOMDocument->loadHTML()()
DOMDocument->loadHTMLFile()()
DOMDocument->loadXML()()
DOMDocument->normalizeDocument()()
DOMDocument->registerNodeClass()()
DOMDocument->relaxNGValidate()()
DOMDocument->relaxNGValidateSource()()
DOMDocument->save()()
DOMDocument->saveHTML()()
DOMDocument->saveHTMLFile()()
DOMDocument->saveXML()()
DOMDocument->schemaValidate()()
DOMDocument->schemaValidateSource()()
DOMDocument->validate()()
DomDocument->xinclude()
DOMDocument->xinclude()()
DOMDocumentFragment->appendXML()()
DomDocumentType->entities()()
DomDocumentType->internal_subset()()
DomDocumentType->name()()
DomDocumentType->notations()()
DomDocumentType->public_id()()
DomDocumentType->system_id()()
DOMElement->__construct()()
DomElement->get_attribute()()
DomElement->get_attribute_node()()
DomElement->get_elements_by_tagname()()
DOMElement->getAttribute()()
DOMElement->getAttributeNode()()
DOMElement->getAttributeNodeNS()()
DOMElement->getAttributeNS()()
DOMElement->getElementsByTagName()()
DOMElement->getElementsByTagNameNS()()
DomElement->has_attribute()()
DOMElement->hasAttribute()()
DOMElement->hasAttributeNS()()
DomElement->remove_attribute()()
DOMElement->removeAttribute()()
DOMElement->removeAttributeNode()()
DOMElement->removeAttributeNS()()
DomElement->set_attribute()()
DomElement->set_attribute_node()()
DOMElement->setAttribute()()
DOMElement->setAttributeNode()()
DOMElement->setAttributeNodeNS()()
DOMElement->setAttributeNS()()
DOMElement->setIdAttribute()()
DOMElement->setIdAttributeNode()()
DOMElement->setIdAttributeNS()()
DomElement->tagname()()
DOMEntityReference->__construct()()
DOMImplementation->__construct()()
DOMImplementation->createDocument()()
DOMImplementation->createDocumentType()()
DOMImplementation->hasFeature()()
DOMNamedNodeMap->getNamedItem()()
DOMNamedNodeMap->getNamedItemNS()()
DOMNamedNodeMap->item()()
DomNode->add_namespace()
DomNode->append_child()
DomNode->append_sibling()
DOMNode->appendChild()()
DomNode->attributes()
DomNode->child_nodes()
DomNode->clone_node()
DOMNode->cloneNode()()
DomNode->dump_node()
DomNode->first_child()
DomNode->get_content()
DomNode->has_attributes()
DomNode->has_child_nodes()
DOMNode->hasAttributes()()
DOMNode->hasChildNodes()()
DomNode->insert_before()
DOMNode->insertBefore()()
DomNode->is_blank_node()
DOMNode->isDefaultNamespace()()
DOMNode->isSameNode()()

F
fam_cancel_monitor()
fam_close()
fam_monitor_collection()
fam_monitor_directory()
fam_monitor_file()
fam_next_event()
fam_open()
fam_pending()
fam_resume_monitor()
fam_suspend_monitor()
fbsql_affected_rows()
fbsql_autocommit()
fbsql_blob_size()
fbsql_change_user()
fbsql_clob_size()
fbsql_close()
fbsql_commit()
fbsql_connect()
fbsql_create_blob()
fbsql_create_clob()
fbsql_create_db()
fbsql_data_seek()
fbsql_database()
fbsql_database_password()
fbsql_db_query()
fbsql_db_status()
fbsql_drop_db()
fbsql_errno()
fbsql_error()
fbsql_fetch_array()
fbsql_fetch_assoc()
fbsql_fetch_field()
fbsql_fetch_lengths()
fbsql_fetch_object()
fbsql_fetch_row()
fbsql_field_flags()
fbsql_field_len()
fbsql_field_name()
fbsql_field_seek()
fbsql_field_table()
fbsql_field_type()
fbsql_free_result()
fbsql_get_autostart_info()
fbsql_hostname()
fbsql_insert_id()
fbsql_list_dbs()
fbsql_list_fields()
fbsql_list_tables()
fbsql_next_result()
fbsql_num_fields()
fbsql_num_rows()
fbsql_password()
fbsql_pconnect()
fbsql_query()
fbsql_read_blob()
fbsql_read_clob()
fbsql_result()
fbsql_rollback()
fbsql_rows_fetched()
fbsql_select_db()
fbsql_set_characterset()
fbsql_set_lob_mode()
fbsql_set_password()
fbsql_set_transaction()
fbsql_start_db()
fbsql_stop_db()
fbsql_table_name()
fbsql_tablename()
fbsql_username()
fbsql_warnings()
fclose()
fdf_add_doc_javascript()
fdf_add_template()
fdf_close()
fdf_create()
fdf_enum_values()
fdf_errno()
fdf_error()
fdf_get_ap()
fdf_get_attachment()
fdf_get_encoding()
fdf_get_file()
fdf_get_flags()
fdf_get_opt()
fdf_get_status()
fdf_get_value()
fdf_get_version()
fdf_header()
fdf_next_field_name()
fdf_open()
fdf_open_string()
fdf_remove_item()
fdf_save()
fdf_save_string()
fdf_set_ap()
fdf_set_encoding()
fdf_set_file()
fdf_set_flags()
fdf_set_javascript_action()
fdf_set_on_import_javascript()

fdf_set_opt()
fdf_set_status()
fdf_set_submit_form_action()
fdf_set_target_frame()
fdf_set_value()
fdf_set_version()
feof()
fflush()
fgetc()
fgetcsv()
fgets()
fgetss()
file()
file_exists()
file_get_contents()
file_put_contents()
fileatime()
filectime()
filegroup()
fileinode()
filemtime()
fileowner()
fileperms()
filepro()
filepro_fieldcount()
filepro_fieldname()
filepro_fieldtype()
filepro_fieldwidth()
filepro_retrieve()
filepro_rowcount()
filesize()
filetype()
filter_has_var()
filter_id()
filter_input()
filter_input_array()
filter_list()
filter_var()
filter_var_array()
FilterIterator::current()
FilterIterator::getInnerIterator()
FilterIterator::key()
FilterIterator::next()
FilterIterator::rewind()
FilterIterator::valid()
finfo->__construct()()
finfo_buffer()
finfo_close()
finfo_file()
finfo_open()
finfo_set_flags()
floatval()
flock()
floor()
flush()
fmod()
fnmatch()
fopen()
fpassthru()
fprintf()
fputcsv()
fputs()
fread()
frenchtojd()
fribidi_log2vis()
fscanf()
fseek()
fsockopen()
fstat()
ftell()
ftok()
ftp_alloc()
ftp_cdup()
ftp_chdir()
ftp_chmod()
ftp_close()
ftp_connect()
ftp_delete()
ftp_exec()
ftp_fget()
ftp_fput()
ftp_get()
ftp_get_option()
ftp_login()
ftp_mdtm()
ftp_mkdir()
ftp_nb_continue()
ftp_nb_fget()
ftp_nb_fput()
ftp_nb_get()
ftp_nb_put()
ftp_nlist()
ftp_pasv()
ftp_put()
ftp_pwd()
ftp_quit()
ftp_raw()
ftp_rawlist()
ftp_rename()
ftp_rmdir()
ftp_set_option()

G

HaruDoc::setCurrentEncoder()
HaruDoc::setEncryptionMode()
HaruDoc::setInfoAttr()
HaruDoc::setInfoDateAttr()
HaruDoc::setOpenAction()
HaruDoc::setPageLayout()
HaruDoc::setPageMode()
HaruDoc::setPagesConfiguration()
HaruDoc::setPassword()
HaruDoc::setPermission()
HaruDoc::useCNSEncodings()
HaruDoc::useCNSFonts()
HaruDoc::useCNTEncodings()
HaruDoc::useCNTFonts()
HaruDoc::useJPEncodings()
HaruDoc::useJPFonts()
HaruDoc::useKREncodings()
HaruDoc::useKRFonts()

haruencoder()

HaruEncoder::getByteType()
HaruEncoder::getType()
HaruEncoder::getUnicode()
HaruEncoder::getWritingMode()

haruexception()

harufont()

HaruFont::getAscent()
HaruFont::getCapHeight()
HaruFont::getDescent()
HaruFont::getEncodingName()
HaruFont::getFontName()
HaruFont::getTextWidth()
HaruFont::getUnicodeWidth()
HaruFont::getXHeight()
HaruFont::measureText()

haruimage()

HaruImage::getBitsPerComponent()
HaruImage::getColorSpace()
HaruImage::getHeight()
HaruImage::getSize()
HaruImage::getWidth()
HaruImage::setColorMask()
HaruImage::setMaskImage()

haruoutline()

HaruOutline::setDestination()
HaruOutline::setOpened()

harupage()

HaruPage::arc()
HaruPage::beginText()
HaruPage::circle()
HaruPage::closePath()
HaruPage::concat()
HaruPage::createDestination()
HaruPage::createLinkAnnotation()
HaruPage::createTextAnnotation()
HaruPage::createURLAnnotation()
HaruPage::curveTo()
HaruPage::curveTo2()
HaruPage::curveTo3()
HaruPage::drawImage()
HaruPage::ellipse()
HaruPage::endPath()
HaruPage::endText()
HaruPage::eofill()
HaruPage::eoFillStroke()
HaruPage::fill()
HaruPage::fillStroke()
HaruPage::getCharSpace()
HaruPage::getCMYKFill()
HaruPage::getCMYKStroke()
HaruPage::getCurrentFont()
HaruPage::getCurrentFontSize()
HaruPage::getCurrentPos()
HaruPage::getCurrentTextPos()
HaruPage::getDash()
HaruPage::getFillingColorSpace()
HaruPage::getFlatness()
HaruPage::getGMode()
HaruPage::getGrayFill()
HaruPage::getGrayStroke()
HaruPage::getHeight()
HaruPage::getHorizontalScaling()
HaruPage::getLineCap()
HaruPage::getLineJoin()
HaruPage::getLineWidth()
HaruPage::getMiterLimit()
HaruPage::getRGBFill()
HaruPage::getRGBStroke()
HaruPage::getStrokingColorSpace()
HaruPage::getTextLeading()
HaruPage::getTextMatrix()
HaruPage::getTextRenderingMode()
HaruPage::getTextRise()
HaruPage::getTextWidth()
HaruPage::getTransMatrix()
HaruPage::getWidth()
HaruPage::getWordSpace()
HaruPage::lineTo()
HaruPage::measureText()
HaruPage::moveTextPos()
HaruPage::moveTo()
HaruPage::moveToNextLine()

HaruPage::rectangle()
HaruPage::setCharSpace()
HaruPage::setCMYKFill()
HaruPage::setCMYKStroke()
HaruPage::setDash()
HaruPage::setFlatness()
HaruPage::setFontAndSize()
HaruPage::setGrayFill()
HaruPage::setGrayStroke()
HaruPage::setHeight()
HaruPage::setHorizontalScaling()
HaruPage::setLineCap()
HaruPage::setLineJoin()
HaruPage::setLineWidth()
HaruPage::setMiterLimit()
HaruPage::setRGBFill()
HaruPage::setRGBStroke()
HaruPage::setRotate()
HaruPage::setSize()
HaruPage::setSlideShow()
HaruPage::setTextLeading()
HaruPage::setTextMatrix()
HaruPage::setTextRenderingMode()
HaruPage::setTextRise()
HaruPage::setWidth()
HaruPage::setWordSpace()
HaruPage::showText()
HaruPage::showTextNextLine()
HaruPage::stroke()
HaruPage::textOut()
HaruPage::textRect()
hash()
hash_algos()
hash_file()
hash_final()
hash_hmac()
hash_hmac_file()
hash_init()
hash_update()
hash_update_file()
hash_update_stream()
header()
headers_list()
headers_sent()
hebrev()
hebrevc()
hexdec()
highlight_file()
highlight_string()
html_entity_decode()
htmlentities()
htmlspecialchars()
htmlspecialchars_decode()
http_build_cookie()
http_build_query()
http_build_str()
http_build_url()
http_cache_etag()
http_cache_last_modified()
http_chunked_decode()
http_date()
http_deflate()
http_get()
http_get_request_body()
http_get_request_body_stream()
http_get_request_headers()
http_head()
http_inflate()
http_match_etag()
http_match_modified()
http_match_request_header()
http_negotiate_charset()
http_negotiate_content_type()
http_negotiate_language()
http_parse_cookie()
http_parse_headers()
http_parse_message()
http_parse_params()
http_persistent_handles_clean()
http_persistent_handles_count()
http_persistent_handles_ident()
http_post_data()
http_post_fields()
http_put_data()
http_put_file()
http_put_stream()
http_redirect()
http_request()
http_request_body_encode()
http_request_method_exists()
http_request_method_name()
http_request_method_register()
http_request_method_unregister()
http_send_content_disposition()
http_send_content_type()
http_send_data()
http_send_file()
http_send_last_modified()
http_send_status()
http_send_stream()
http_support()

http_throttle()
httpdeflatestream()
HttpDeflateStream::__construct()
HttpDeflateStream::factory()
HttpDeflateStream::finish()
HttpDeflateStream::flush()
HttpDeflateStream::update()
httpinflatestream()
HttpInflateStream::__construct()
HttpInflateStream::factory()
HttpInflateStream::finish()
HttpInflateStream::flush()
HttpInflateStream::update()
httpmessage()
HttpMessage::__construct()
HttpMessage::addHeaders()
HttpMessage::detach()
HttpMessage::factory()
HttpMessage::fromEnv()
HttpMessage::fromString()
HttpMessage::getBody()
HttpMessage::getHeader()
HttpMessage::getHeaders()
HttpMessage::getHttpVersion()
HttpMessage::getParentMessage()
HttpMessage::getRequestMethod()
HttpMessage::getRequestUrl()
HttpMessage::getResponseCode()
HttpMessage::getResponseStatus()
HttpMessage::getType()
HttpMessage::guessContentType()
HttpMessage::prepend()
HttpMessage::reverse()
HttpMessage::send()
HttpMessage::setBody()
HttpMessage::setHeaders()
HttpMessage::setHttpVersion()
HttpMessage::setRequestMethod()
HttpMessage::setRequestUrl()
HttpMessage::setResponseCode()
HttpMessage::setResponseStatus()
HttpMessage::setType()
HttpMessage::toMessageTypeObject()
HttpMessage::toString()
httpquerystring()
HttpQueryString::__construct()
HttpQueryString::get()
HttpQueryString::mod()
HttpQueryString::set()
HttpQueryString::singleton()
HttpQueryString::toArray()
HttpQueryString::toString()
HttpQueryString::xlate()
httprequest()
HttpRequest::__construct()
HttpRequest::addCookies()
HttpRequest::addHeaders()
HttpRequest::addPostFields()
HttpRequest::addPostFile()
HttpRequest::addPutData()
HttpRequest::addQueryData()
HttpRequest::addRawPostData()
HttpRequest::addSslOptions()
HttpRequest::clearHistory()
HttpRequest::enableCookies()
HttpRequest::getContentType()
HttpRequest::getCookies()
HttpRequest::getHeaders()
HttpRequest::getHistory()
HttpRequest::getMethod()
HttpRequest::getOptions()
HttpRequest::getPostFields()
HttpRequest::getPostFiles()
HttpRequest::getPutData()
HttpRequest::getPutFile()
HttpRequest::getQueryData()
HttpRequest::getRawPostData()
HttpRequest::getRawRequestMessage()
HttpRequest::getRawResponseMessage()
HttpRequest::getRequestMessage()
HttpRequest::getResponseBody()
HttpRequest::getResponseCode()
HttpRequest::getResponseCookies()
HttpRequest::getResponseData()
HttpRequest::getResponseHeader()
HttpRequest::getResponseInfo()
HttpRequest::getResponseMessage()
HttpRequest::getResponseStatus()
HttpRequest::getSslOptions()
HttpRequest::getUrl()
HttpRequest::resetCookies()
HttpRequest::send()
HttpRequest::setContentType()
HttpRequest::setCookies()
HttpRequest::setHeaders()
HttpRequest::setMethod()
HttpRequest::setOptions()
HttpRequest::setPostFields()
HttpRequest::setPostFiles()
HttpRequest::setPutData()
HttpRequest::setPutFile()

HttpRequest::setQueryData()
HttpRequest::setRawPostData()
HttpRequest::setSslOptions()
HttpRequest::setUrl()
httprequestpool()
HttpRequestPool::__construct()
HttpRequestPool::__destruct()
HttpRequestPool::attach()
HttpRequestPool::detach()
HttpRequestPool::getAttachedRequests()
HttpRequestPool::getFinishedRequests()
HttpRequestPool::reset()
HttpRequestPool::send()
HttpRequestPool::socketPerform()
HttpRequestPool::socketSelect()
httpresponse()
HttpResponse::capture()
HttpResponse::getBufferSize()
HttpResponse::getCache()
HttpResponse::getCacheControl()
HttpResponse::getContentDisposition()
HttpResponse::getContentType()
HttpResponse::getData()
HttpResponse::getETag()
HttpResponse::getFile()
HttpResponse::getGzip()
HttpResponse::getHeader()
HttpResponse::getLastModified()
HttpResponse::getRequestBody()
HttpResponse::getRequestBodyStream()
HttpResponse::getRequestHeaders()
HttpResponse::getStream()
HttpResponse::getThrottleDelay()
HttpResponse::guessContentType()
HttpResponse::redirect()
HttpResponse::send()
HttpResponse::setBufferSize()
HttpResponse::setCache()
HttpResponse::setCacheControl()
HttpResponse::setContentDisposition()
HttpResponse::setContentType()
HttpResponse::setData()
HttpResponse::setETag()
HttpResponse::setFile()
HttpResponse::setGzip()
HttpResponse::setHeader()
HttpResponse::setLastModified()
HttpResponse::setStream()
HttpResponse::setThrottleDelay()
HttpResponse::status()
hw_api->checkin()
hw_api->checkout()
hw_api->children()
hw_api->content()
hw_api->copy()
hw_api->dbstat()
hw_api->dcstat()
hw_api->dstanchors()
hw_api->dstofsrcanchor()
hw_api->find()
hw_api->ftstat()
hw_api->hwstat()
hw_api->identify()
hw_api->info()
hw_api->insert()
hw_api->insertanchor()
hw_api->insertcollection()
hw_api->insertdocument()
hw_api->link()
hw_api->lock()
hw_api->move()
hw_api->object()
hw_api->objectbyanchor()
hw_api->parents()
hw_api->remove()
hw_api->replace()
hw_api->setcommittedversion()
hw_api->srcanchors()
hw_api->srcsofdst()
hw_api->unlock()
hw_api->user()
hw_api->userlist()
hw_api_attribute()
hw_api_attribute->key()
hw_api_attribute->langdepvalue()
hw_api_attribute->value()
hw_api_attribute->values()
hw_api_content()
hw_api_content->mimetype()
hw_api_content->read()
hw_api_error->count()
hw_api_error->reason()
hw_api_object()
hw_api_object->assign()
hw_api_object->attreditable()
hw_api_object->count()
hw_api_object->insert()
hw_api_object->remove()
hw_api_object->title()
hw_api_object->value()
hw_api_reason->description()

Imagick::chopImage()
Imagick::clear()
Imagick::clipImage()
Imagick::clipPathImage()
Imagick::clone()
Imagick::coalesceImages()
Imagick::colorFloodfillImage()
Imagick::colorizeImage()
Imagick::combineImages()
Imagick::commentImage()
Imagick::compareImageChannels()
Imagick::compareImageLayers()
Imagick::compositeImage()
Imagick::contrastImage()
Imagick::contrastStretchImage()
Imagick::convolveImage()
Imagick::cropImage()
Imagick::cropThumbnailImage()
Imagick::current()
Imagick::cycleColormapImage()
Imagick::deconstructImages()
Imagick::despeckleImage()
Imagick::destroy()
Imagick::displayImage()
Imagick::displayImages()
Imagick::drawImage()
Imagick::edgeImage()
Imagick::embossImage()
Imagick::enhanceImage()
Imagick::equalizeImage()
Imagick::evaluateImage()
Imagick::flattenImages()
Imagick::flipImage()
Imagick::flopImage()
Imagick::frameImage()
Imagick::fxImage()
Imagick::gammaImage()
Imagick::gaussianBlurImage()
Imagick::getCompression()
Imagick::getCompressionQuality()
Imagick::getCopyright()
Imagick::getFilename()
Imagick::getFormat()
Imagick::getHomeURL()
Imagick::getImage()
Imagick::getImageBackgroundColor()
Imagick::getImageBlob()
Imagick::getImageBluePrimary()
Imagick::getImageBorderColor()
Imagick::getImageChannelDepth()
Imagick::getImageChannelDistortion()
Imagick::getImageChannelExtrema()
Imagick::getImageChannelMean()
Imagick::getImageChannelStatistics()
Imagick::getImageColormapColor()
Imagick::getImageColors()
Imagick::getImageColorspace()
Imagick::getImageCompose()
Imagick::getImageDelay()
Imagick::getImageDepth()
Imagick::getImageDispose()
Imagick::getImageDistortion()
Imagick::getImageExtrema()
Imagick::getImageFilename()
Imagick::getImageFormat()
Imagick::getImageGamma()
Imagick::getImageGeometry()
Imagick::getImageGreenPrimary()
Imagick::getImageHeight()
Imagick::getImageHistogram()
Imagick::getImageIndex()
Imagick::getImageInterlaceScheme()
Imagick::getImageInterpolateMethod()
Imagick::getImageIterations()
Imagick::getImageMatte()
Imagick::getImageMatteColor()
Imagick::getImagePage()
Imagick::getImagePixelColor()
Imagick::getImageProfile()
Imagick::getImageProperty()
Imagick::getImageRedPrimary()
Imagick::getImageRegion()
Imagick::getImageRenderingIntent()
Imagick::getImageResolution()
Imagick::getImageScene()
Imagick::getImageSignature()
Imagick::getImageSize()
Imagick::getImageTicksPerSecond()
Imagick::getImageTotalInkDensity()
Imagick::getImageType()
Imagick::getImageUnits()
Imagick::getImageVirtualPixelMethod()
Imagick::getImageWhitePoint()
Imagick::getImageWidth()
Imagick::getInterlaceScheme()
Imagick::getNumberImages()
Imagick::getOption()
Imagick::getPackageName()
Imagick::getPage()
Imagick::getPixelIterator()
Imagick::getPixelRegionIterator()

Imagick::getQuantumDepth()
Imagick::getQuantumRange()
Imagick::getReleaseDate()
Imagick::getResource()
Imagick::getResourceLimit()
Imagick::getSamplingFactors()
Imagick::getSize()
Imagick::getSizeOffset()
Imagick::getVersion()
Imagick::hasNextImage()
Imagick::hasPreviousImage()
Imagick::identifyImage()
Imagick::implodeImage()
Imagick::labelImage()
Imagick::levelImage()
Imagick::linearStretchImage()
Imagick::magnifyImage()
Imagick::matteFloodfillImage()
Imagick::medianFilterImage()
Imagick::minifyImage()
Imagick::modulateImage()
Imagick::montageImage()
Imagick::morphImages()
Imagick::mosaicImages()
Imagick::motionBlurImage()
Imagick::negateImage()
Imagick::newImage()
Imagick::newPseudoImage()
Imagick::nextImage()
Imagick::normalizeImage()
Imagick::oilPaintImage()
Imagick::optimizeImageLayers()
Imagick::paintOpaqueImage()
Imagick::paintTransparentImage()
Imagick::pingImage()
Imagick::pingImageBlob()
Imagick::pingImageFile()
Imagick::polaroidImage()
Imagick::posterizeImage()
Imagick::previousImage()
Imagick::profileImage()
Imagick::queryFontMetrics()
Imagick::queryFonts()
Imagick::queryFormats()
Imagick::radialBlurImage()
Imagick::raiseImage()
Imagick::randomThresholdImage()
Imagick::readImage()
Imagick::readImageBlob()
Imagick::readImageFile()
Imagick::reduceNoiseImage()
Imagick::removeImage()
Imagick::removeImageProfile()
Imagick::render()
Imagick::resampleImage()
Imagick::resizeImage()
Imagick::rollImage()
Imagick::rotateImage()
Imagick::roundCorners()
Imagick::sampleImage()
Imagick::scaleImage()
Imagick::separateImageChannel()
Imagick::sepiaToneImage()
Imagick::setBackgroundColor()
Imagick::setCompression()
Imagick::setCompressionQuality()
Imagick::setFilename()
Imagick::setFirstIterator()
Imagick::setFormat()
Imagick::setImageBackgroundColor()
Imagick::setImageBias()
Imagick::setImageBluePrimary()
Imagick::setImageBorderColor()
Imagick::setImageChannelDepth()
Imagick::setImageColormapColor()
Imagick::setImageColorspace()
Imagick::setImageCompose()
Imagick::setImageCompression()
Imagick::setImageDelay()
Imagick::setImageDepth()
Imagick::setImageDispose()
Imagick::setImageExtent()
Imagick::setImageFilename()
Imagick::setImageFormat()
Imagick::setImageGamma()
Imagick::setImageGreenPrimary()
Imagick::setImageIndex()
Imagick::setImageInterlaceScheme()
Imagick::setImageInterpolateMethod()
Imagick::setImageIterations()
Imagick::setImageMatte()
Imagick::setImageMatteColor()
Imagick::setImagePage()
Imagick::setImageProfile()
Imagick::setImageProperty()
Imagick::setImageRedPrimary()
Imagick::setImageRenderingIntent()
Imagick::setImageResolution()
Imagick::setImageScene()
Imagick::setImageTicksPerSecond()
Imagick::setImageType()

Imagick::setImageUnits()
Imagick::setImageVirtualPixelMethod()
Imagick::setImageWhitePoint()
Imagick::setInterlaceScheme()
Imagick::setOption()
Imagick::setPage()
Imagick::setResolution()
Imagick::setResourceLimit()
Imagick::setSamplingFactors()
Imagick::setSize()
Imagick::setSizeOffset()
Imagick::setType()
Imagick::shadeImage()
Imagick::shadowImage()
Imagick::sharpenImage()
Imagick::shaveImage()
Imagick::shearImage()
Imagick::sigmoidalContrastImage()
Imagick::sketchImage()
Imagick::solarizeImage()
Imagick::spliceImage()
Imagick::spreadImage()
Imagick::steganoImage()
Imagick::stereoImage()
Imagick::stripImage()
Imagick::swirlImage()
Imagick::textureImage()
Imagick::thresholdImage()
Imagick::thumbnailImage()
Imagick::tintImage()
Imagick::transverseImage()
Imagick::trimImage()
Imagick::uniqueImageColors()
Imagick::unsharpMaskImage()
Imagick::valid()
Imagick::vignetteImage()
Imagick::waveImage()
Imagick::whiteThresholdImage()
Imagick::writeImage()
Imagick::writeImages()
ImagickDraw::__construct()
ImagickDraw::affine()
ImagickDraw::annotation()
ImagickDraw::arc()
ImagickDraw::bezier()
ImagickDraw::circle()
ImagickDraw::clear()
ImagickDraw::clone()
ImagickDraw::color()
ImagickDraw::comment()
ImagickDraw::composite()
ImagickDraw::destroy()
ImagickDraw::ellipse()
ImagickDraw::getClipPath()
ImagickDraw::getClipRule()
ImagickDraw::getClipUnits()
ImagickDraw::getFillColor()
ImagickDraw::getFillOpacity()
ImagickDraw::getFillRule()
ImagickDraw::getFont()
ImagickDraw::getFontFamily()
ImagickDraw::getFontSize()
ImagickDraw::getFontStyle()
ImagickDraw::getFontWeight()
ImagickDraw::getGravity()
ImagickDraw::getStrokeAntialias()
ImagickDraw::getStrokeColor()
ImagickDraw::getStrokeDashArray()
ImagickDraw::getStrokeDashOffset()
ImagickDraw::getStrokeLineCap()
ImagickDraw::getStrokeLineJoin()
ImagickDraw::getStrokeMiterLimit()
ImagickDraw::getStrokeOpacity()
ImagickDraw::getStrokeWidth()
ImagickDraw::getTextAlignment()
ImagickDraw::getTextAntialias()
ImagickDraw::getTextDecoration()
ImagickDraw::getTextEncoding()
ImagickDraw::getTextUnderColor()
ImagickDraw::getVectorGraphics()
ImagickDraw::line()
ImagickDraw::matte()
ImagickDraw::pathClose()
ImagickDraw::pathCurveToAbsolute()
ImagickDraw::pathCurveToQuadraticBezierAbsolute()
ImagickDraw::pathCurveToQuadraticBezierRelative()
ImagickDraw::pathCurveToQuadraticBezierSmoothAbsolute()
ImagickDraw::pathCurveToQuadraticBezierSmoothRelative()
ImagickDraw::pathCurveToRelative()
ImagickDraw::pathCurveToSmoothAbsolute()
ImagickDraw::pathCurveToSmoothRelative()
ImagickDraw::pathEllipticArcAbsolute()
ImagickDraw::pathEllipticArcRelative()
ImagickDraw::pathFinish()
ImagickDraw::pathLineToAbsolute()
ImagickDraw::pathLineToHorizontalAbsolute()
ImagickDraw::pathLineToHorizontalRelative()
ImagickDraw::pathLineToRelative()
ImagickDraw::pathLineToVerticalAbsolute()
ImagickDraw::pathLineToVerticalRelative()
ImagickDraw::pathMoveToAbsolute()

mailparse_msg_extract_whole_part_file()
mailparse_msg_free()
mailparse_msg_get_part()
mailparse_msg_get_part_data()
mailparse_msg_get_structure()
mailparse_msg_parse()
mailparse_msg_parse_file()
mailparse_rfc822_parse_addresses()
mailparse_stream_encode()
mailparse_uudecode_all()
main()
max()
maxdb()()
maxdb->affected_rows()
maxdb->auto_commit()
maxdb->change_user()
maxdb->character_set_name()
maxdb->close()
maxdb->close_long_data()
maxdb->commit()
maxdb->disable_reads_from_master()
maxdb->errno()
maxdb->fetch_assoc()
maxdb->field_count()
maxdb->get_host_info()
maxdb->info()
maxdb->insert_id()
maxdb->kill()
maxdb->more_results()
maxdb->multi_query()
maxdb->next_result()
maxdb->options()
maxdb->ping()
maxdb->prepare()
maxdb->protocol_version()
maxdb->query()
maxdb->real_connect()
maxdb->real_escape_string()
maxdb->real_query()
maxdb->rollback()
maxdb->rpl_query_type()
maxdb->select_db()
maxdb->send_query()
maxdb->server_info()
maxdb->sqlstate()
maxdb->ssl_set()
maxdb->stat()
maxdb->stmt_init()
maxdb->store_result()
maxdb->store_result()
maxdb->thread_id()
maxdb->use_result()
maxdb->warning_count()
maxdb_affected_rows()
maxdb_autocommit()
maxdb_bind_param()
maxdb_bind_result()
maxdb_change_user()
maxdb_character_set_name()
maxdb_client_encoding()
maxdb_close()
maxdb_close_long_data()
maxdb_commit()
maxdb_connect()
maxdb_connect_errno()
maxdb_connect_error()
maxdb_data_seek()
maxdb_debug()
maxdb_disable_reads_from_master()
maxdb_disable_rpl_parse()
maxdb_dump_debug_info()
maxdb_embedded_connect()
maxdb_enable_reads_from_master()
maxdb_enable_rpl_parse()
maxdb_errno()
maxdb_error()
maxdb_escape_string()
maxdb_execute()
maxdb_fetch()
maxdb_fetch_array()
maxdb_fetch_assoc()
maxdb_fetch_field()
maxdb_fetch_field_direct()
maxdb_fetch_fields()
maxdb_fetch_lengths()
maxdb_fetch_object()
maxdb_fetch_row()
maxdb_field_count()
maxdb_field_seek()
maxdb_field_tell()
maxdb_free_result()
maxdb_get_client_info()
maxdb_get_client_version()
maxdb_get_host_info()
maxdb_get_metadata()
maxdb_get_proto_info()
maxdb_get_server_info()
maxdb_get_server_version()
maxdb_info()
maxdb_init()
maxdb_insert_id()

mb_strripos()
mb_strrpos()
mb_strstr()
mb_strtolower()
mb_strtoupper()
mb_strwidth()
mb_substitute_character()
mb_substr()
mb_substr_count()
mcal_append_event()
mcal_close()
mcal_create_calendar()
mcal_date_compare()
mcal_date_valid()
mcal_day_of_week()
mcal_day_of_year()
mcal_days_in_month()
mcal_delete_calendar()
mcal_delete_event()
mcal_event_add_attribute()
mcal_event_init()
mcal_event_set_alarm()
mcal_event_set_category()
mcal_event_set_class()
mcal_event_set_description()
mcal_event_set_end()
mcal_event_set_recur_daily()
mcal_event_set_recur_monthly_mday()
mcal_event_set_recur_monthly_wday()
mcal_event_set_recur_none()
mcal_event_set_recur_weekly()
mcal_event_set_recur_yearly()
mcal_event_set_start()
mcal_event_set_title()
mcal_expunge()
mcal_fetch_current_stream_event()
mcal_fetch_event()
mcal_is_leap_year()
mcal_list_alarms()
mcal_list_events()
mcal_next_recurrence()
mcal_open()
mcal_popen()
mcal_rename_calendar()
mcal_reopen()
mcal_snooze()
mcal_store_event()
mcal_time_valid()
mcal_week_of_year()
mcrypt_cbc()
mcrypt_cfb()
mcrypt_create_iv()
mcrypt_decrypt()
mcrypt_ecb()
mcrypt_enc_get_algorithms_name()
mcrypt_enc_get_block_size()
mcrypt_enc_get_iv_size()
mcrypt_enc_get_key_size()
mcrypt_enc_get_modes_name()
mcrypt_enc_get_supported_key_sizes()
mcrypt_enc_is_block_algorithm()
mcrypt_enc_is_block_algorithm_mode()
mcrypt_enc_is_block_mode()
mcrypt_enc_self_test()
mcrypt_encrypt()
mcrypt_generic()
mcrypt_generic_deinit()
mcrypt_generic_end()
mcrypt_generic_init()
mcrypt_get_block_size()
mcrypt_get_cipher_name()
mcrypt_get_iv_size()
mcrypt_get_key_size()
mcrypt_list_algorithms()
mcrypt_list_modes()
mcrypt_module_close()
mcrypt_module_get_algo_block_size()
mcrypt_module_get_algo_key_size()
mcrypt_module_get_supported_key_sizes()
mcrypt_module_is_block_algorithm()
mcrypt_module_is_block_algorithm_mode()
mcrypt_module_is_block_mode()
mcrypt_module_open()
mcrypt_module_self_test()
mcrypt_ofb()
md5()
md5_file()
mdecrypt_generic()
Memcache::add()
Memcache::addServer()
Memcache::close()
Memcache::connect()
Memcache::decrement()
Memcache::delete()
Memcache::flush()
Memcache::get()
Memcache::getExtendedStats()
Memcache::getServerStatus()
Memcache::getStats()
Memcache::getVersion()
Memcache::increment()

Memcache::pconnect()
Memcache::replace()
Memcache::set()
Memcache::setCompressThreshold()
Memcache::setServerParams()
memcache_debug()
memory_get_peak_usage()
memory_get_usage()
metaphone()
method_exists()
mhash()
mhash_count()
mhash_get_block_size()
mhash_get_hash_name()
mhash_keygen_s2k()
microtime()
mime_content_type()
min()
ming_keypress()
ming_setcubicthreshold()
ming_setscale()
ming_setswfcompression()
ming_useconstants()
ming_useswfversion()
mkdir()
mktime()
money_format()
move_uploaded_file()
msession_connect()
msession_count()
msession_create()
msession_destroy()
msession_disconnect()
msession_find()
msession_get()
msession_get_array()
msession_get_data()
msession_inc()
msession_list()
msession_listvar()
msession_lock()
msession_plugin()
msession_randstr()
msession_set()
msession_set_array()
msession_set_data()
msession_timeout()
msession_uniq()
msession_unlock()
msg_get_queue()
msg_receive()
msg_remove_queue()
msg_send()
msg_set_queue()
msg_stat_queue()
msql()
msql_affected_rows()
msql_close()
msql_connect()
msql_create_db()
msql_createdb()
msql_data_seek()
msql_db_query()
msql_dbname()
msql_drop_db()
msql_error()
msql_fetch_array()
msql_fetch_field()
msql_fetch_object()
msql_fetch_row()
msql_field_flags()
msql_field_len()
msql_field_name()
msql_field_seek()
msql_field_table()
msql_field_type()
msql_fieldflags()
msql_fieldlen()
msql_fieldname()
msql_fieldtable()
msql_fieldtype()
msql_free_result()
msql_list_dbs()
msql_list_fields()
msql_list_tables()
msql_num_fields()
msql_num_rows()
msql_numfields()
msql_numrows()
msql_pconnect()
msql_query()
msql_regcase()
msql_result()
msql_select_db()
msql_tablename()
mssql_bind()
mssql_close()
mssql_connect()
mssql_data_seek()
mssql_execute()
mssql_fetch_array()

mssql_fetch_assoc()
mssql_fetch_batch()
mssql_fetch_field()
mssql_fetch_object()
mssql_fetch_row()
mssql_field_length()
mssql_field_name()
mssql_field_seek()
mssql_field_type()
mssql_free_result()
mssql_free_statement()
mssql_get_last_message()
mssql_guid_string()
mssql_init()
mssql_min_error_severity()
mssql_min_message_severity()
mssql_next_result()
mssql_num_fields()
mssql_num_rows()
mssql_pconnect()
mssql_query()
mssql_result()
mssql_rows_affected()
mssql_select_db()
mt_getrandmax()
mt_rand()
mt_srand()
muscat_close()
muscat_get()
muscat_give()
muscat_setup()
muscat_setup_net()
mysql_affected_rows()
mysql_change_user()
mysql_client_encoding()
mysql_close()
mysql_connect()
mysql_create_db()
mysql_data_seek()
mysql_db_name()
mysql_db_query()
mysql_drop_db()
mysql_errno()
mysql_error()
mysql_escape_string()
mysql_fetch_array()
mysql_fetch_assoc()
mysql_fetch_field()
mysql_fetch_lengths()
mysql_fetch_object()
mysql_fetch_row()
mysql_field_flags()
mysql_field_len()
mysql_field_name()
mysql_field_seek()
mysql_field_table()
mysql_field_type()
mysql_free_result()
mysql_get_client_info()
mysql_get_host_info()
mysql_get_proto_info()
mysql_get_server_info()
mysql_info()
mysql_insert_id()
mysql_list_dbs()
mysql_list_fields()
mysql_list_processes()
mysql_list_tables()
mysql_num_fields()
mysql_num_rows()
mysql_pconnect()
mysql_ping()
mysql_query()
mysql_real_escape_string()
mysql_result()
mysql_select_db()
mysql_set_charset()
mysql_stat()
mysql_tablename()
mysql_thread_id()
mysql_unbuffered_query()
mysqli->__construct()()
mysqli->affected_rows()
mysqli->autocommit()()
mysqli->change_user()()
mysqli->character_set_name()()
mysqli->close()()
mysqli->commit()()
mysqli->debug()()
mysqli->disable_reads_from_master()()
mysqli->dump_debug_info()()
mysqli->errno()
mysqli->fetch_assoc()()
mysqli->field_count()()
mysqli->host_info()
mysqli->info()
mysqli->insert_id()
mysqli->kill()()
mysqli->more_results()
mysqli->multi_query()()
mysqli->next_result()()

ncurses_insertln()
ncurses_insstr()
ncurses_instr()
ncurses_isendwin()
ncurses_keyok()
ncurses_keypad()
ncurses_killchar()
ncurses_longname()
ncurses_meta()
ncurses_mouse_trafo()
ncurses_mouseinterval()
ncurses_mousemask()
ncurses_move()
ncurses_move_panel()
ncurses_mvaddch()
ncurses_mvaddchnstr()
ncurses_mvaddchstr()
ncurses_mvaddnstr()
ncurses_mvaddstr()
ncurses_mvcur()
ncurses_mvdelch()
ncurses_mvgetch()
ncurses_mvhline()
ncurses_mvinch()
ncurses_mvvline()
ncurses_mvwaddstr()
ncurses_napms()
ncurses_new_panel()
ncurses_newpad()
ncurses_newwin()
ncurses_nl()
ncurses_nocbreak()
ncurses_noecho()
ncurses_nonl()
ncurses_noqiflush()
ncurses_noraw()
ncurses_pair_content()
ncurses_panel_above()
ncurses_panel_below()
ncurses_panel_window()
ncurses_pnoutrefresh()
ncurses_prefresh()
ncurses_putp()
ncurses_qiflush()
ncurses_raw()
ncurses_refresh()
ncurses_replace_panel()
ncurses_reset_prog_mode()
ncurses_reset_shell_mode()
ncurses_resetty()
ncurses_savetty()
ncurses_scr_dump()
ncurses_scr_init()
ncurses_scr_restore()
ncurses_scr_set()
ncurses_scrl()
ncurses_show_panel()
ncurses_slk_attr()
ncurses_slk_attroff()
ncurses_slk_attron()
ncurses_slk_attrset()
ncurses_slk_clear()
ncurses_slk_color()
ncurses_slk_init()
ncurses_slk_noutrefresh()
ncurses_slk_refresh()
ncurses_slk_restore()
ncurses_slk_set()
ncurses_slk_touch()
ncurses_standend()
ncurses_standout()
ncurses_start_color()
ncurses_termattrs()
ncurses_termname()
ncurses_timeout()
ncurses_top_panel()
ncurses_typeahead()
ncurses_ungetch()
ncurses_ungetmouse()
ncurses_update_panels()
ncurses_use_default_colors()
ncurses_use_env()
ncurses_use_extended_names()
ncurses_vidattr()
ncurses_vline()
ncurses_waddch()
ncurses_waddstr()
ncurses_wattroff()
ncurses_wattron()
ncurses_wattrset()
ncurses_wborder()
ncurses_wclear()
ncurses_wcolor_set()
ncurses_werase()
ncurses_wgetch()
ncurses_whline()
ncurses_wmouse_trafo()
ncurses_wmove()
ncurses_wnoutrefresh()
ncurses_wrefresh()
ncurses_wstandend()

ora_numcols()
ora_numrows()
ora_open()
ora_parse()
ora_plogon()
ora_rollback()
orbitenum()
orbitobject()
orbitstruct()
ord()
output_add_rewrite_var()
output_reset_rewrite_vars()
overload()
override_function()
ovrimos_close()
ovrimos_commit()
ovrimos_connect()
ovrimos_cursor()
ovrimos_exec()
ovrimos_execute()
ovrimos_fetch_into()
ovrimos_fetch_row()
ovrimos_field_len()
ovrimos_field_name()
ovrimos_field_num()
ovrimos_field_type()
ovrimos_free_result()
ovrimos_longreadlen()
ovrimos_num_fields()
ovrimos_num_rows()
ovrimos_prepare()
ovrimos_result()
ovrimos_result_all()
ovrimos_rollback()

P

pack()
ParentIterator::getChildren()
ParentIterator::hasChildren()
ParentIterator::next()
ParentIterator::rewind()
parse_ini_file()
parse_str()
parse_url()
parsekit_compile_file()
parsekit_compile_string()
parsekit_func_arginfo()
passthru()
pathinfo()
pclose()
pcntl_alarm()
pcntl_exec()
pcntl_fork()
pcntl_getpriority()
pcntl_setpriority()
pcntl_signal()
pcntl_wait()
pcntl_waitpid()
pcntl_wexitstatus()
pcntl_wifexited()
pcntl_wifsignaled()
pcntl_wifstopped()
pcntl_wstopsig()
pcntl_wtermsig()
pdf_activate_item()
pdf_add_annotation()
pdf_add_bookmark()
pdf_add_launchlink()
pdf_add_locallink()
pdf_add_nameddest()
pdf_add_note()
pdf_add_outline()
pdf_add_pdflink()
pdf_add_table_cell()
pdf_add_textflow()
pdf_add_thumbnail()
pdf_add_weblink()
pdf_arc()
pdf_arcn()
pdf_attach_file()
pdf_begin_document()
pdf_begin_font()
pdf_begin_glyph()
pdf_begin_item()
pdf_begin_layer()
pdf_begin_page()
pdf_begin_page_ext()
pdf_begin_pattern()
pdf_begin_template()
pdf_begin_template_ext()
pdf_circle()
pdf_clip()
pdf_close()
pdf_close_image()
pdf_close_pdi()
pdf_close_pdi_page()
pdf_closepath()
pdf_closepath_fill_stroke()
pdf_closepath_stroke()
pdf_concat()
pdf_continue_text()

pdf_create_3dview()
pdf_create_action()
pdf_create_annotation()
pdf_create_bookmark()
pdf_create_field()
pdf_create_fieldgroup()
pdf_create_gstate()
pdf_create_pvf()
pdf_create_textflow()
pdf_curveto()
pdf_define_layer()
pdf_delete()
pdf_delete_pvf()
pdf_delete_table()
pdf_delete_textflow()
pdf_encoding_set_char()
pdf_end_document()
pdf_end_font()
pdf_end_glyph()
pdf_end_item()
pdf_end_layer()
pdf_end_page()
pdf_end_page_ext()
pdf_end_pattern()
pdf_end_template()
pdf_endpath()
pdf_fill()
pdf_fill_imageblock()
pdf_fill_pdfblock()
pdf_fill_stroke()
pdf_fill_textblock()
pdf_findfont()
pdf_fit_image()
pdf_fit_pdi_page()
pdf_fit_table()
pdf_fit_textflow()
pdf_fit_textline()
pdf_get_apiname()
pdf_get_buffer()
pdf_get_errmsg()
pdf_get_errnum()
pdf_get_font()
pdf_get_fontname()
pdf_get_fontsize()
pdf_get_image_height()
pdf_get_image_width()
pdf_get_majorversion()
pdf_get_minorversion()
pdf_get_parameter()
pdf_get_pdi_parameter()
pdf_get_pdi_value()
pdf_get_value()
pdf_info_font()
pdf_info_matchbox()
pdf_info_table()
pdf_info_textflow()
pdf_info_textline()
pdf_initgraphics()
pdf_lineto()
pdf_load_3ddata()
pdf_load_font()
pdf_load_iccprofile()
pdf_load_image()
pdf_makespotcolor()
pdf_moveto()
pdf_new()
pdf_open_ccitt()
pdf_open_file()
pdf_open_gif()
pdf_open_image()
pdf_open_image_file()
pdf_open_jpeg()
pdf_open_memory_image()
pdf_open_pdi()
pdf_open_pdi_page()
pdf_open_tiff()
pdf_pcos_get_number()
pdf_pcos_get_stream()
pdf_pcos_get_string()
pdf_place_image()
pdf_place_pdi_page()
pdf_process_pdi()
pdf_rect()
pdf_restore()
pdf_resume_page()
pdf_rotate()
pdf_save()
pdf_scale()
pdf_set_border_color()
pdf_set_border_dash()
pdf_set_border_style()
pdf_set_char_spacing()
pdf_set_duration()
pdf_set_gstate()
pdf_set_horiz_scaling()
pdf_set_info()
pdf_set_info_author()
pdf_set_info_creator()
pdf_set_info_keywords()
pdf_set_info_subject()
pdf_set_info_title()

pdf_set_layer_dependency()
pdf_set_leading()
pdf_set_parameter()
pdf_set_text_matrix()
pdf_set_text_pos()
pdf_set_text_rendering()
pdf_set_text_rise()
pdf_set_value()
pdf_set_word_spacing()
pdf_setcolor()
pdf_setdash()
pdf_setdashpattern()
pdf_setflat()
pdf_setfont()
pdf_setgray()
pdf_setgray_fill()
pdf_setgray_stroke()
pdf_setlinecap()
pdf_setlinejoin()
pdf_setlinewidth()
pdf_setmatrix()
pdf_setmiterlimit()
pdf_setpolydash()
pdf_setrgbcolor()
pdf_setrgbcolor_fill()
pdf_setrgbcolor_stroke()
pdf_shading()
pdf_shading_pattern()
pdf_shfill()
pdf_show()
pdf_show_boxed()
pdf_show_xy()
pdf_skew()
pdf_stringwidth()
pdf_stroke()
pdf_suspend_page()
pdf_translate()
pdf_utf16_to_utf8()
pdf_utf32_to_utf16()
pdf_utf8_to_utf16()
PDO->__construct()()
PDO->beginTransaction()()
PDO->commit()()
PDO->errorCode()()
PDO->errorInfo()()
PDO->exec()()
PDO->getAttribute()()
PDO->getAvailableDrivers()()
PDO->lastInsertId()()
PDO->prepare()()
PDO->query()()
PDO->quote()()
PDO->rollBack()()
PDO->setAttribute()()
PDO->sqliteCreateAggregate()()
PDO->sqliteCreateFunction()()
PDO::pgsqlLOBCreate()
PDO::pgsqlLOBOpen()
PDO::pgsqlLOBUnlink()
PDOStatement->bindColumn()()
PDOStatement->bindParam()()
PDOStatement->bindValue()()
PDOStatement->closeCursor()()
PDOStatement->columnCount()()
PDOStatement->errorCode()()
PDOStatement->errorInfo()()
PDOStatement->execute()()
PDOStatement->fetch()()
PDOStatement->fetchAll()()
PDOStatement->fetchColumn()()
PDOStatement->fetchObject()()
PDOStatement->getAttribute()()
PDOStatement->getColumnMeta()()
PDOStatement->nextRowset()()
PDOStatement->rowCount()()
PDOStatement->setAttribute()()
PDOStatement->setFetchMode()()
pfpro_cleanup()
pfpro_init()
pfpro_process()
pfpro_process_raw()
pfpro_version()
pfsockopen()
pg_affected_rows()
pg_cancel_query()
pg_client_encoding()
pg_close()
pg_connect()
pg_connection_busy()
pg_connection_reset()
pg_connection_status()
pg_convert()
pg_copy_from()
pg_copy_to()
pg_dbname()
pg_delete()
pg_end_copy()
pg_escape_bytea()
pg_escape_string()
pg_execute()
pg_fetch_all()

PharFileInfo::__construct()
php_check_syntax()
php_ini_scanned_files()
php_logo_guid()
php_sapi_name()
php_strip_whitespace()
php_uname()
phpcredits()
phpinfo()
phpversion()
pi()
png2wbmp()
popen()
pos()
posix_access()
posix_ctermid()
posix_get_last_error()
posix_getcwd()
posix_getegid()
posix_geteuid()
posix_getgid()
posix_getgrgid()
posix_getgrnam()
posix_getgroups()
posix_getlogin()
posix_getpgid()
posix_getpgrp()
posix_getpid()
posix_getppid()
posix_getpwnam()
posix_getpwuid()
posix_getrlimit()
posix_getsid()
posix_getuid()
posix_initgroups()
posix_isatty()
posix_kill()
posix_mkfifo()
posix_mknod()
posix_setegid()
posix_seteuid()
posix_setgid()
posix_setpgid()
posix_setsid()
posix_setuid()
posix_strerror()
posix_times()
posix_ttyname()
posix_uname()
pow()
preg_grep()
preg_last_error()
preg_match()
preg_match_all()
preg_quote()
preg_replace()
preg_replace_callback()
preg_split()
prev()
print()
print_r()
printer_abort()
printer_close()
printer_create_brush()
printer_create_dc()
printer_create_font()
printer_create_pen()
printer_delete_brush()
printer_delete_dc()
printer_delete_font()
printer_delete_pen()
printer_draw_bmp()
printer_draw_chord()
printer_draw_elipse()
printer_draw_line()
printer_draw_pie()
printer_draw_rectangle()
printer_draw_roundrect()
printer_draw_text()
printer_end_doc()
printer_end_page()
printer_get_option()
printer_list()
printer_logical_fontheight()
printer_open()
printer_select_brush()
printer_select_font()
printer_select_pen()
printer_set_option()
printer_start_doc()
printer_start_page()
printer_write()
printf()
proc_close()
proc_get_status()
proc_nice()
proc_open()
proc_terminate()
property_exists()
ps_add_bookmark()
ps_add_launchlink()

ps_add_locallink()
ps_add_note()
ps_add_pdflink()
ps_add_weblink()
ps_arc()
ps_arcn()
ps_begin_page()
ps_begin_pattern()
ps_begin_template()
ps_circle()
ps_clip()
ps_close()
ps_close_image()
ps_closepath()
ps_closepath_stroke()
ps_continue_text()
ps_curveto()
ps_delete()
ps_end_page()
ps_end_pattern()
ps_end_template()
ps_fill()
ps_fill_stroke()
ps_findfont()
ps_get_buffer()
ps_get_parameter()
ps_get_value()
ps_hyphenate()
ps_include_file()
ps_lineto()
ps_makespotcolor()
ps_moveto()
ps_new()
ps_open_file()
ps_open_image()
ps_open_image_file()
ps_open_memory_image()
ps_place_image()
ps_rect()
ps_restore()
ps_rotate()
ps_save()
ps_scale()
ps_set_border_color()
ps_set_border_dash()
ps_set_border_style()
ps_set_info()
ps_set_parameter()
ps_set_text_pos()
ps_set_value()
ps_setcolor()
ps_setdash()
ps_setflat()
ps_setfont()
ps_setgray()
ps_setlinecap()
ps_setlinejoin()
ps_setlinewidth()
ps_setmiterlimit()
ps_setoverprintmode()
ps_setpolydash()
ps_shading()
ps_shading_pattern()
ps_shfill()
ps_show()
ps_show2()
ps_show_boxed()
ps_show_xy()
ps_show_xy2()
ps_string_geometry()
ps_stringwidth()
ps_stroke()
ps_symbol()
ps_symbol_name()
ps_symbol_width()
ps_translate()
pspell_add_to_personal()
pspell_add_to_session()
pspell_check()
pspell_clear_session()
pspell_config_create()
pspell_config_data_dir()
pspell_config_dict_dir()
pspell_config_ignore()
pspell_config_mode()
pspell_config_personal()
pspell_config_repl()
pspell_config_runtogether()
pspell_config_save_repl()
pspell_new()
pspell_new_config()
pspell_new_personal()
pspell_save_wordlist()
pspell_store_replacement()
pspell_suggest()
putenv()
px_close()
px_create_fp()
px_date2string()
px_delete()
px_delete_record()

px_get_field()
px_get_info()
px_get_parameter()
px_get_record()
px_get_schema()
px_get_value()
px_insert_record()
px_new()
px_numfields()
px_numrecords()
px_open_fp()
px_put_record()
px_retrieve_record()
px_set_blob_file()
px_set_parameter()
px_set_tablename()
px_set_targetencoding()
px_set_value()
px_timestamp2string()
px_update_record()

Q
qdom_error()
qdom_tree()
quoted_printable_decode()
quotemeta()

R
rad2deg()
radius_acct_open()
radius_add_server()
radius_auth_open()
radius_close()
radius_config()
radius_create_request()
radius_cvt_addr()
radius_cvt_int()
radius_cvt_string()
radius_demangle()
radius_demangle_mppe_key()
radius_get_attr()
radius_get_vendor_attr()
radius_put_addr()
radius_put_attr()
radius_put_int()
radius_put_string()
radius_put_vendor_addr()
radius_put_vendor_attr()
radius_put_vendor_int()
radius_put_vendor_string()
radius_request_authenticator()
radius_send_request()
radius_server_secret()
radius_strerror()
rand()
range()
Rar::extract()
Rar::getAttr()
Rar::getCrc()
Rar::getFileTime()
Rar::getHostOs()
Rar::getMethod()
Rar::getName()
Rar::getPackedSize()
Rar::getUnpackedSize()
Rar::getVersion()
rar_close()
rar_entry_get()
rar_list()
rar_open()
rawurldecode()
rawurlencode()
read_exif_data()
readdir()
readfile()
readgzfile()
readline()
readline_add_history()
readline_callback_handler_install()
readline_callback_handler_remove()
readline_callback_read_char()
readline_clear_history()
readline_completion_function()
readline_info()
readline_list_history()
readline_on_new_line()
readline_read_history()
readline_redisplay()
readline_write_history()
readlink()
realpath()
recode()
recode_file()
recode_string()
RecursiveDirectoryIterator::getChildren()
RecursiveDirectoryIterator::hasChildren()
RecursiveDirectoryIterator::key()
RecursiveDirectoryIterator::next()
RecursiveDirectoryIterator::rewind()
RecursiveIteratorIterator::current()
RecursiveIteratorIterator::getDepth()

RecursiveIteratorIterator::getSubIterator()
RecursiveIteratorIterator::key()
RecursiveIteratorIterator::next()
RecursiveIteratorIterator::rewind()
RecursiveIteratorIterator::valid()
register_shutdown_function()
register_tick_function()
rename()
rename_function()
reset()
resources()
restore_error_handler()
restore_exception_handler()
restore_include_path()
result->current_field()
result->current_field()
result->data_seek()
result->data_seek()()
result->fetch_array()
result->fetch_array()()
result->fetch_field()
result->fetch_field()()
result->fetch_field_direct()
result->fetch_field_direct()()
result->fetch_fields()
result->fetch_fields()()
result->fetch_object()
result->fetch_object()()
result->fetch_row()
result->fetch_row()()
result->field_count()
result->field_count()
result->field_seek()
result->field_seek()()
result->free()
result->free()()
result->lengths()
result->lengths()()
result->num_rows()
rewind()
rewinddir()
rmdir()
round()
rpm_close()
rpm_get_tag()
rpm_is_valid()
rpm_open()
rpm_version()
rsort()
rtrim()
runkit_class_adopt()
runkit_class_emancipate()
runkit_constant_add()
runkit_constant_redefine()
runkit_constant_remove()
runkit_function_add()
runkit_function_copy()
runkit_function_redefine()
runkit_function_remove()
runkit_function_rename()
runkit_import()
runkit_lint()
runkit_lint_file()
runkit_method_add()
runkit_method_copy()
runkit_method_redefine()
runkit_method_remove()
runkit_method_rename()
runkit_return_value_used()
runkit_sandbox()
runkit_sandbox_output_handler()
runkit_sandbox_parent()
runkit_superglobals()

S
SAMConnection->__construct()()
SAMConnection->commit()()
SAMConnection->connect()()
SAMConnection->disconnect()()
SAMConnection->errno()
SAMConnection->error()
SAMConnection->isConnected()()
SAMConnection->peek()()
SAMConnection->peekAll()()
SAMConnection->receive()()
SAMConnection->remove()()
SAMConnection->rollback()()
SAMConnection->send()()
SAMConnection->subscribe()()
SAMConnection->unsubscribe()()
SAMConnection::setDebug()()
SAMMessage->__construct()()
SAMMessage->body()
SAMMessage->header()
satellite_caught_exception()
satellite_exception_id()
satellite_exception_value()
satellite_get_repository_id()
satellite_load_idl()
satellite_object_to_string()
SCA::createDataObject()

SCA::getService()
SCA_LocalProxy::createDataObject()
SCA_SoapProxy::createDataObject()
scandir()
SDO_DAS_ChangeSummary::beginLogging()
SDO_DAS_ChangeSummary::endLogging()
SDO_DAS_ChangeSummary::getChangedDataObjects()
SDO_DAS_ChangeSummary::getChangeType()
SDO_DAS_ChangeSummary::getOldContainer()
SDO_DAS_ChangeSummary::getOldValues()
SDO_DAS_ChangeSummary::isLogging()
SDO_DAS_DataFactory::addPropertyToType()
SDO_DAS_DataFactory::addType()
SDO_DAS_DataFactory::getDataFactory()
SDO_DAS_DataObject::getChangeSummary()
SDO_DAS_Relational::__construct()
SDO_DAS_Relational::applyChanges()
SDO_DAS_Relational::createRootDataObject()
SDO_DAS_Relational::executePreparedQuery()
SDO_DAS_Relational::executeQuery()
SDO_DAS_Setting::getListIndex()
SDO_DAS_Setting::getPropertyIndex()
SDO_DAS_Setting::getPropertyName()
SDO_DAS_Setting::getValue()
SDO_DAS_Setting::isSet()
SDO_DAS_XML::addTypes()
SDO_DAS_XML::create()
SDO_DAS_XML::createDataObject()
SDO_DAS_XML::createDocument()
SDO_DAS_XML::loadFile()
SDO_DAS_XML::loadString()
SDO_DAS_XML::saveFile()
SDO_DAS_XML::saveString()
SDO_DAS_XML_Document::getRootDataObject()
SDO_DAS_XML_Document::getRootElementName()
SDO_DAS_XML_Document::getRootElementURI()
SDO_DAS_XML_Document::setEncoding()
SDO_DAS_XML_Document::setXMLDeclaration()
SDO_DAS_XML_Document::setXMLVersion()
SDO_DataFactory::create()
SDO_DataObject::clear()
SDO_DataObject::createDataObject()
SDO_DataObject::getContainer()
SDO_DataObject::getSequence()
SDO_DataObject::getTypeName()
SDO_DataObject::getTypeNamespaceURI()
SDO_Exception::getCause()
SDO_List::insert()
SDO_Model_Property::getContainingType()
SDO_Model_Property::getDefault()
SDO_Model_Property::getName()
SDO_Model_Property::getType()
SDO_Model_Property::isContainment()
SDO_Model_Property::isMany()
SDO_Model_ReflectionDataObject::__construct()
SDO_Model_ReflectionDataObject::export()
SDO_Model_ReflectionDataObject::getContainmentProperty()
SDO_Model_ReflectionDataObject::getInstanceProperties()
SDO_Model_ReflectionDataObject::getType()
SDO_Model_Type::getBaseType()
SDO_Model_Type::getName()
SDO_Model_Type::getNamespaceURI()
SDO_Model_Type::getProperties()
SDO_Model_Type::getProperty()
SDO_Model_Type::isAbstractType()
SDO_Model_Type::isDataType()
SDO_Model_Type::isInstance()
SDO_Model_Type::isOpenType()
SDO_Model_Type::isSequencedType()
SDO_Sequence::getProperty()
SDO_Sequence::insert()
SDO_Sequence::move()
sem_acquire()
sem_get()
sem_release()
sem_remove()
serialize()
sesam_affected_rows()
sesam_commit()
sesam_connect()
sesam_diagnostic()
sesam_disconnect()
sesam_errormsg()
sesam_execimm()
sesam_fetch_array()
sesam_fetch_result()
sesam_fetch_row()
sesam_field_array()
sesam_field_name()
sesam_free_result()
sesam_num_fields()
sesam_query()
sesam_rollback()
sesam_seek_row()
sesam_settransaction()
session_cache_expire()
session_cache_limiter()
session_commit()
session_decode()
session_destroy()
session_encode()

session_get_cookie_params()
session_id()
session_is_registered()
session_module_name()
session_name()
session_pgsql_add_error()
session_pgsql_get_error()
session_pgsql_get_field()
session_pgsql_reset()
session_pgsql_set_field()
session_pgsql_status()
session_regenerate_id()
session_register()
session_save_path()
session_set_cookie_params()
session_set_save_handler()
session_start()
session_unregister()
session_unset()
session_write_close()
set_error_handler()
set_exception_handler()
set_file_buffer()
set_include_path()
set_magic_quotes_runtime()
set_time_limit()
setcookie()
setlocale()
setrawcookie()
settype()
sha1()
sha1_file()
shell_exec()
shm_attach()
shm_detach()
shm_get_var()
shm_put_var()
shm_remove()
shm_remove_var()
shmop_close()
shmop_delete()
shmop_open()
shmop_read()
shmop_size()
shmop_write()
show_source()
shuffle()
similar_text()
simplexml_import_dom()
simplexml_load_file()
simplexml_load_string()
SimpleXMLElement->__construct()()
SimpleXMLElement->addAttribute()()
SimpleXMLElement->addChild()()
SimpleXMLElement->asXML()()
SimpleXMLElement->attributes()()
SimpleXMLElement->children()()
SimpleXMLElement->getDocNamespaces()()
SimpleXMLElement->getName()()
SimpleXMLElement->getNamespaces()()
SimpleXMLElement->registerXPathNamespace()()
SimpleXMLElement->xpath()()
SimpleXMLIterator::current()
SimpleXMLIterator::getChildren()
SimpleXMLIterator::hasChildren()
SimpleXMLIterator::key()
SimpleXMLIterator::next()
SimpleXMLIterator::rewind()
SimpleXMLIterator::valid()
sin()
sinh()
sizeof()
sleep()
snmp_get_quick_print()
snmp_get_valueretrieval()
snmp_read_mib()
snmp_set_enum_print()
snmp_set_oid_numeric_print()
snmp_set_oid_output_format()
snmp_set_quick_print()
snmp_set_valueretrieval()
snmpget()
snmpgetnext()
snmprealwalk()
snmpset()
snmpwalk()
snmpwalkoid()
SoapClient->__call()()
SoapClient->__construct()()
SoapClient->__doRequest()()
SoapClient->__getFunctions()()
SoapClient->__getLastRequest()()
SoapClient->__getLastRequestHeaders()()
SoapClient->__getLastResponse()()
SoapClient->__getLastResponseHeaders()()
SoapClient->__getTypes()()
SoapClient->__setCookie()()
SoapClient->__soapCall()()
SoapFault->__construct()()
SoapHeader->__construct()()
SoapParam->__construct()()

SoapServer->__construct()()
SoapServer->addFunction()()
SoapServer->fault()()
SoapServer->getFunctions()()
SoapServer->handle()()
SoapServer->setClass()()
SoapServer->setPersistence()()
SoapVar->__construct()()
socket_accept()
socket_bind()
socket_clear_error()
socket_close()
socket_connect()
socket_create()
socket_create_listen()
socket_create_pair()
socket_get_option()
socket_get_status()
socket_getpeername()
socket_getsockname()
socket_last_error()
socket_listen()
socket_read()
socket_recv()
socket_recvfrom()
socket_select()
socket_send()
socket_sendto()
socket_set_block()
socket_set_blocking()
socket_set_nonblock()
socket_set_option()
socket_set_timeout()
socket_shutdown()
socket_strerror()
socket_write()
sort()
soundex()
spl_autoload()
spl_autoload_call()
spl_autoload_extensions()
spl_autoload_functions()
spl_autoload_register()
spl_autoload_unregister()
spl_classes()
spl_object_hash()
split()
spliti()
sprintf()
sql_regcase()
sqlite_array_query()
sqlite_busy_timeout()
sqlite_changes()
sqlite_close()
sqlite_column()
sqlite_create_aggregate()
sqlite_create_function()
sqlite_current()
sqlite_error_string()
sqlite_escape_string()
sqlite_exec()
sqlite_factory()
sqlite_fetch_all()
sqlite_fetch_array()
sqlite_fetch_column_types()
sqlite_fetch_object()
sqlite_fetch_single()
sqlite_fetch_string()
sqlite_field_name()
sqlite_has_more()
sqlite_has_prev()
sqlite_key()
sqlite_last_error()
sqlite_last_insert_rowid()
sqlite_libencoding()
sqlite_libversion()
sqlite_next()
sqlite_num_fields()
sqlite_num_rows()
sqlite_open()
sqlite_popen()
sqlite_prev()
sqlite_query()
sqlite_rewind()
sqlite_seek()
sqlite_single_query()
sqlite_udf_decode_binary()
sqlite_udf_encode_binary()
sqlite_unbuffered_query()
sqlite_valid()
SQLiteDatabase->arrayQuery()
SQLiteDatabase->busyTimeout()
SQLiteDatabase->changes()
SQLiteDatabase->createAggregate()
SQLiteDatabase->createFunction()
SQLiteDatabase->exec()
SQLiteDatabase->fetchColumnTypes()
SQLiteDatabase->lastError()
SQLiteDatabase->lastInsertRowid()
SQLiteDatabase->query()
SQLiteDatabase->singleQuery()

SQLiteDatabase->unbufferedQuery()
SQLiteResult->column()
SQLiteResult->current()
SQLiteResult->fetch()
SQLiteResult->fetchAll()
SQLiteResult->fetchObject()
SQLiteResult->fetchSingle()
SQLiteResult->fieldName()
SQLiteResult->hasPrev()
SQLiteResult->key()
SQLiteResult->next()
SQLiteResult->numFields()
SQLiteResult->numRows()
SQLiteResult->prev()
SQLiteResult->rewind()
SQLiteResult->seek()
SQLiteResult->valid()
SQLiteUnbuffered->column()
SQLiteUnbuffered->current()
SQLiteUnbuffered->fetch()
SQLiteUnbuffered->fetchAll()
SQLiteUnbuffered->fetchObject()
SQLiteUnbuffered->fetchSingle()
SQLiteUnbuffered->fieldName()
SQLiteUnbuffered->next()
SQLiteUnbuffered->numFields()
SQLiteUnbuffered->valid()
sqrt()
srand()
sscanf()
ssh2_auth_hostbased_file()
ssh2_auth_none()
ssh2_auth_password()
ssh2_auth_pubkey_file()
ssh2_connect()
ssh2_exec()
ssh2_fetch_stream()
ssh2_fingerprint()
ssh2_methods_negotiated()
ssh2_publickey_add()
ssh2_publickey_init()
ssh2_publickey_list()
ssh2_publickey_remove()
ssh2_scp_recv()
ssh2_scp_send()
ssh2_sftp()
ssh2_sftp_lstat()
ssh2_sftp_mkdir()
ssh2_sftp_readlink()
ssh2_sftp_realpath()
ssh2_sftp_rename()
ssh2_sftp_rmdir()
ssh2_sftp_stat()
ssh2_sftp_symlink()
ssh2_sftp_unlink()
ssh2_shell()
ssh2_tunnel()
stat()
stats_absolute_deviation()
stats_cdf_beta()
stats_cdf_binomial()
stats_cdf_cauchy()
stats_cdf_chisquare()
stats_cdf_exponential()
stats_cdf_f()
stats_cdf_gamma()
stats_cdf_laplace()
stats_cdf_logistic()
stats_cdf_negative_binomial()
stats_cdf_noncentral_chisquare()
stats_cdf_noncentral_f()
stats_cdf_poisson()
stats_cdf_t()
stats_cdf_uniform()
stats_cdf_weibull()
stats_covariance()
stats_den_uniform()
stats_dens_beta()
stats_dens_cauchy()
stats_dens_chisquare()
stats_dens_exponential()
stats_dens_f()
stats_dens_gamma()
stats_dens_laplace()
stats_dens_logistic()
stats_dens_negative_binomial()
stats_dens_normal()
stats_dens_pmf_binomial()
stats_dens_pmf_hypergeometric()
stats_dens_pmf_poisson()
stats_dens_t()
stats_dens_weibull()
stats_harmonic_mean()
stats_kurtosis()
stats_rand_gen_beta()
stats_rand_gen_chisquare()
stats_rand_gen_exponential()
stats_rand_gen_f()
stats_rand_gen_funiform()
stats_rand_gen_gamma()
stats_rand_gen_ibinomial()

stats_rand_gen_ibinomial_negative()
stats_rand_gen_int()
stats_rand_gen_ipoisson()
stats_rand_gen_iuniform()
stats_rand_gen_noncenral_chisquare()
stats_rand_gen_noncentral_f()
stats_rand_gen_noncentral_t()
stats_rand_gen_normal()
stats_rand_gen_t()
stats_rand_get_seeds()
stats_rand_phrase_to_seeds()
stats_rand_ranf()
stats_rand_setall()
stats_skew()
stats_standard_deviation()
stats_stat_binomial_coef()
stats_stat_correlation()
stats_stat_gennch()
stats_stat_independent_t()
stats_stat_innerproduct()
stats_stat_noncentral_t()
stats_stat_paired_t()
stats_stat_percentile()
stats_stat_powersum()
stats_variance()
stmt->bind_param()
stmt->bind_param()()
stmt->bind_result()
stmt->bind_result()()
stmt->close_long_data()
stmt->data_seek()
stmt->data_seek()()
stmt->execute()
stmt->execute()()
stmt->fetch()
stmt->fetch()()
stmt->free_result()
stmt->free_result()()
stmt->num_rows()
stmt->num_rows()
stmt->param_count()
stmt->param_count()
stmt->prepare()
stmt->prepare()()
stmt->reset()
stmt->reset()()
stmt->result_metadata()()
stmt->send_long_data()
stmt->send_long_data()()
str_getcsv()
str_ireplace()
str_pad()
str_repeat()
str_replace()
str_rot13()
str_shuffle()
str_split()
str_word_count()
strcasecmp()
strchr()
strcmp()
strcoll()
strcspn()
stream_bucket_append()
stream_bucket_make_writeable()
stream_bucket_new()
stream_bucket_prepend()
stream_context_create()
stream_context_get_default()
stream_context_get_options()
stream_context_set_option()
stream_context_set_params()
stream_copy_to_stream()
stream_encoding()
stream_filter_append()
stream_filter_prepend()
stream_filter_register()
stream_filter_remove()
stream_get_contents()
stream_get_filters()
stream_get_line()
stream_get_meta_data()
stream_get_transports()
stream_get_wrappers()
stream_register_wrapper()
stream_resolve_include_path()
stream_select()
stream_set_blocking()
stream_set_timeout()
stream_set_write_buffer()
stream_socket_accept()
stream_socket_client()
stream_socket_enable_crypto()
stream_socket_get_name()
stream_socket_pair()
stream_socket_recvfrom()
stream_socket_sendto()
stream_socket_server()
stream_socket_shutdown()
stream_wrapper_register()
stream_wrapper_restore()

swf_getfontinfo()
swf_getframe()
swf_labelframe()
swf_lookat()
swf_modifyobject()
swf_mulcolor()
swf_nextid()
swf_oncondition()
swf_openfile()
swf_ortho()
swf_ortho2()
swf_perspective()
swf_placeobject()
swf_polarview()
swf_popmatrix()
swf_posround()
swf_pushmatrix()
swf_removeobject()
swf_rotate()
swf_scale()
swf_setfont()
swf_setframe()
swf_shapearc()
swf_shapecurveto()
swf_shapecurveto3()
swf_shapefillbitmapclip()
swf_shapefillbitmaptile()
swf_shapefilloff()
swf_shapefillsolid()
swf_shapelinesolid()
swf_shapelineto()
swf_shapemoveto()
swf_showframe()
swf_startbutton()
swf_startdoaction()
swf_startshape()
swf_startsymbol()
swf_textwidth()
swf_translate()
swf_viewport()
swfaction()
SWFAction->__construct()()
swfbitmap()
SWFBitmap->__construct()()
SWFBitmap->getHeight()()
SWFBitmap->getWidth()()
swfbutton()
SWFButton->__construct()()
SWFButton->addAction()()
SWFButton->addASound()()
SWFButton->addShape()()
SWFButton->setAction()()
SWFButton->setDown()()
SWFButton->setHit()()
SWFButton->setMenu()()
SWFButton->setOver()()
SWFButton->setUp()()
swfdisplayitem()
SWFDisplayItem->addAction()()
SWFDisplayItem->addColor()()
SWFDisplayItem->endMask()()
SWFDisplayItem->getRot()()
SWFDisplayItem->getX()()
SWFDisplayItem->getXScale()()
SWFDisplayItem->getXSkew()()
SWFDisplayItem->getY()()
SWFDisplayItem->getYScale()()
SWFDisplayItem->getYSkew()()
SWFDisplayItem->move()()
SWFDisplayItem->moveTo()()
SWFDisplayItem->multColor()()
SWFDisplayItem->remove()()
SWFDisplayItem->rotate()()
SWFDisplayItem->rotateTo()()
SWFDisplayItem->scale()()
SWFDisplayItem->scaleTo()()
SWFDisplayItem->setDepth()()
SWFDisplayItem->setMaskLevel()()
SWFDisplayItem->setMatrix()()
SWFDisplayItem->setName()()
SWFDisplayItem->setRatio()()
SWFDisplayItem->skewX()()
SWFDisplayItem->skewXTo()()
SWFDisplayItem->skewY()()
SWFDisplayItem->skewYTo()()
swffill()
SWFFill->moveTo()()
SWFFill->rotateTo()()
SWFFill->scaleTo()()
SWFFill->skewXTo()()
SWFFill->skewYTo()()
swffont()
SWFFont->__construct()()
SWFFont->getAscent()()
SWFFont->getDescent()()
SWFFont->getLeading()()
SWFFont->getShape()()
SWFFont->getUTF8Width()()
SWFFont->getWidth()()
swffontchar()
SWFFontChar->addChars()()

SWFFontChar->addUTF8Chars()()
swfgradient()
SWFGradient->__construct()()
SWFGradient->addEntry()()
swfmorph()
SWFMorph->__construct()()
SWFMorph->getShape1()()
SWFMorph->getShape2()()
swfmovie()
SWFMovie->__construct()()
SWFMovie->add()()
SWFMovie->addExport()()
SWFMovie->addFont()()
SWFMovie->importChar()()
SWFMovie->importFont()()
SWFMovie->labelFrame()()
SWFMovie->nextFrame()()
SWFMovie->output()()
SWFMovie->remove()()
SWFMovie->save()()
SWFMovie->saveToFile()()
SWFMovie->setbackground()()
SWFMovie->setDimension()()
SWFMovie->setFrames()()
SWFMovie->setRate()()
SWFMovie->startSound()()
SWFMovie->stopSound()()
SWFMovie->streamMP3()()
SWFMovie->writeExports()()
swfprebuiltclip()
SWFPrebuiltClip->__construct()()
swfshape()
SWFShape->__construct()()
SWFShape->addFill()()
SWFShape->drawArc()()
SWFShape->drawCircle()()
SWFShape->drawCubic()()
SWFShape->drawCubicTo()()
SWFShape->drawCurve()()
SWFShape->drawCurveTo()()
SWFShape->drawGlyph()()
SWFShape->drawLine()()
SWFShape->drawLineTo()()
SWFShape->movePen()()
SWFShape->movePenTo()()
SWFShape->setLeftFill()()
SWFShape->setLine()()
SWFShape->setRightFill()()
swfsound()
swfsound()
swfsoundinstance()
SWFSoundInstance->loopCount()()
SWFSoundInstance->loopInPoint()()
SWFSoundInstance->loopOutPoint()()
SWFSoundInstance->noMultiple()()
swfsprite()
SWFSprite->__construct()()
SWFSprite->add()()
SWFSprite->labelFrame()()
SWFSprite->nextFrame()()
SWFSprite->remove()()
SWFSprite->setFrames()()
SWFSprite->startSound()()
SWFSprite->stopSound()()
swftext()
SWFText->__construct()()
SWFText->addString()()
SWFText->addUTF8String()()
SWFText->getAscent()()
SWFText->getDescent()()
SWFText->getLeading()()
SWFText->getUTF8Width()()
SWFText->getWidth()()
SWFText->moveTo()()
SWFText->setColor()()
SWFText->setFont()()
SWFText->setHeight()()
SWFText->setSpacing()()
swftextfield()
SWFTextField->__construct()()
SWFTextField->addChars()()
SWFTextField->addString()()
SWFTextField->align()()
SWFTextField->setBounds()()
SWFTextField->setColor()()
SWFTextField->setFont()()
SWFTextField->setHeight()()
SWFTextField->setIndentation()()
SWFTextField->setLeftMargin()()
SWFTextField->setLineSpacing()()
SWFTextField->setMargins()()
SWFTextField->setName()()
SWFTextField->setPadding()()
SWFTextField->setRightMargin()()
swfvideostream()
SWFVideoStream->__construct()()
SWFVideoStream->getNumFrames()()
SWFVideoStream->setDimension()()
Swish->getMetaList()
Swish->getPropertyList()
Swish->prepare()

- MS SQL Server (PDO) — Microsoft SQL Server および Sybase 関数 (PDO_DBLIB)
- Msession — Mohawk Software セッションハンドラ関数
- mSQL — mSQL 関数
- マルチバイト文字列 — マルチバイト文字列関数 (mbstring)
- muscat — muscat 関数
- MySQL — MySQL 関数
- MySQL (PDO) — MySQL 関数 (PDO_MYSQL)
- mysqli — MySQL 改良版拡張サポート (mysqli)
- Ncurses — Ncurses 端末画面制御関数
- ネットワーク — ネットワーク関数
- Newt — Newt 関数
- NSAPI — NSAPI用関数
- オブジェクトの集約 — オブジェクトの集約/合成関数
- オブジェクトのオーバーロード — オブジェクトプロパティとメソッドコールのオーバーロード
- OCI8 — Oracle 関数
- ODBC — Unified ODBC 関数
- ODBC and DB2 (PDO) — ODBC および DB2 関数 (PDO_ODBC)
- OGG/Vorbis — oggvorbis
- openal — OpenAL 音声バインディング
- OpenSSL — OpenSSL 関数
- Oracle — Oracle 関数 [推奨されません]
- Oracle (PDO) — Oracle 関数 (PDO_OCI)
- 出力制御 — 出力制御関数(output control)
- OvrimosSQL — Ovrimos SQL 関数
- Paradox — Paradox ファイルアクセス
- Parsekit — Parsekit 関数
- PCNTL — プロセス制御関数
- PCRE — 正規表現関数 (Perl 互換)
- PDF — PDF 関数
- PDO — PDO 関数
- phar — Phar アーカイブストリームおよびクラス
- PHP オプション/情報 — PHP オプションと情報(info)
- POSIX — POSIX 関数
- POSIX Regex — 正規表現(regex)関数 (POSIX拡張サポート)
- PostgreSQL — PostgreSQL 関数
- PostgreSQL (PDO) — PostgreSQL 関数 (PDO_PGSQL)
- Printer — プリンタ関数
- プログラム実行 — プログラム実行関数
- PS — PostScript ドキュメントの作成
- Pspell — Pspell 関数
- qtdom — qtdom 関数
- radius — Radius
- Rar — Rar 関数
- Readline — GNU Readline
- Recode — GNU Recode 関数
- RPMReader — RPM ヘッダ読み込み関数
- runkit — runkit 関数
- SAM — SAM - Simple Asynchronous Messaging: 単純な非同期メッセージング
- Satellite — Satellite CORBA クライアント拡張 [推奨されません]
- SCA — SCA 関数
- SDO — SDO 関数
- SDO DAS XML — SDO XML データアクセスサービス関数
- SDO-DAS-Relational — SDO リレーショナルデータアクセスサービス関数
- Semaphore — セマフォ・共有メモリおよび IPC 関数(semaphore)
- SESAM — SESAM データベース関数
- Session PgSQL — PostgreSQL セッション保存ハンドラ
- セッション — セッション処理関数(session)
- shmop — 共有メモリ関数(shmop)
- SimpleXML — SimpleXML関数
- SNMP — SNMP 関数
- SOAP — SOAP関数
- ソケット — ソケット関数
- SPL — Standard PHP Library (SPL) 関数
- SQLite — SQLite 関数
