プログラミング言語PHP

作成日:2009年7月20日

PHPの歴史

1994年　　PHP(Personal Home Page Tools)
1995年　　PHP／FI(Personal Home Page / Form Interpreter)
1997年　　PHP／FI　2.0
1998年6月　PHP(PHP:Hypertext Preprocessor)　3
2000年5月　PHP 4
2004年7月　PHP 5

PHPの正式名称は「PHP:Hypertext Preprocessor」
現在の最新版はPHP 5.3.0
(PHP6はまだ開発中)

対応しているプラットフォーム

- Linux
- Windows
- AS/400
- Mac OS X
- Novell NetWare
- OS/2
- RISC OS
- SGI IRIX 6.5.x
- Solaris (SPARC, INTEL)
- Solaris OpenCSW packages

PHPの特徴

1. サーバ側で実行するHTML埋め込み型のスクリプト言語。
2. 文法はC言語に似ている。
3. MicrosoftのASPより便利な関数が多い。
4. ほとんどのデータベースに対応。
5. PerlのCPANのようなライブラリ(PEAR[ペア])がある。
6. テンプレートエンジン(Smarty[スマーティー]など)
   を使用することで、ロジックとデザインの分離が
   可能(最初の読み出し時だけPHPファイルに変換する
   処理が実行されるので遅い！)
7. 多数のWebアプリケーションフレームワークがある。
   (Maple, CakePHP, Ethna, symfony, Piece Framework,
   ZendFrameworkなど)

PHPを使用しているサイト

PHPは小・中規模だけでなく、大規模案件の稼働実績があります。
(例)
- Yahoo
- 楽天(※)
- ぐるなび
- GREE(グリー)

※楽天のサイトは複数のプログラム言語(Java, Ruby On Rails など)で構築されており、一部分でPHPが使用されています。

PHPで作られている
Webアプリケーション(一部)

- phpMyAdmin　・・・ MySQL用
- phpPgAdmin　・・・ PostgreSQL用
- ibWebAdmin　・・・ Firebird用
- XOOPS　・・・ コンテンツ管理
- XOOPS Cube　・・・ コンテンツ管理
- NetCommons　・・・ コンテンツ管理
- OpenPNE　・・・ SNSサイト運営用
- PHPNuke　・・・ コンテンツ管理
- phpBB　・・・ 掲示板
- EC-CUBE　・・・ ECパッケージ
- PukiWiki　・・・ PHPで作られたWiki
- osCommerce　・・・ ECサイト構築システム

PHPの環境構築(Windows)

・インストール　XAMPP(ザンプ)

(1) XAMPP for Windowsのダウンロード
XAMPPのサイト(http://www.apachefriends.org/jp/xampp.html)より、XAMPP for Windowsを
ダウンロードしてください。

(2) XAMPP for Windowsのインストール
・ダウンロードしたファイルをダブルクリックしてください。
・セキュリティの警告ダイアログが表示されますが、気にせずに「実行」をクリックしてください。
・言語の設定画面が表示されたら「Japanese」を選択して「OK」をクリックしてください。
・英語で注意事項の画面が表示された場合は内容を読んでから「OK」をクリックしてください。
・インストーラ画面が表示されますので、「次へ」をクリックしてください。
・インストール先設定画面が表示されますので、特別な理由がなければデフォルトのまま「次へ」を
クリックしてください。
・オプション項目の設定画面が表示されますので、デフォルトのまま「次へ」をクリックしてください。
・インストールが完了すると「セットアップウィザードは完了しました」という画面が表示されますので、「完了」を
クリックしてください。
・コントロールパネルを起動するかどうかを確認するダイアログが表示されますので、「はい」をクリック
してください。
・コントロールパネルが起動したら「Apache」と「MySQL」の右側にある「開始」ボタンをクリックしてください。

(3) Apacheの動作を確認
・Webブラウザを起動し「http://localhost/」にアクセスしてください。
・画面中央に「XAMPP」と表示されている場合は成功です。
※「日本語」のリンクをクリックするとXAMPPのステータス表示やツールにアクセスできる画面が表示します。

プログラムの開始と終了

<?php ?>
<? ?>
<% %>
<script language=”php”> </script>

上記の4つの記述方法がありますが、一般的には1番上の
書き方の記述方法が多く使用されています。

変数、定数

・変数

(1) 変数は宣言しなくても使用できます。
(2) 変数名の先頭の1文字目は必ず「$」を指定する必要があります。
(3) 変数に使える文字は数字、英字、アンダースコア「_」のみ。
(4) 変数の始まりには数字は使えません。
(5) 変数名の大文字と小文字は区別されます。

・定数

≪構文≫

```
define(【定数名】,【値】);
```

≪使用例≫

```
define(ZEIRITSU,0.05);
print  ZEIRITSU;
```

コメント

・一行コメント

「//」または「#」より後の文字がコメントとして認識されます。

≪構文≫

```
//【コメント】
#【コメント】
```

・複数行コメント

「/*」と「*/」で囲んだブロックがコメントとして認識されます。

≪構文≫

```
/*
   【コメント1】
   【コメント2】
*/
```

スーパーグローバル
すべてのスコープで使用できる組み込みの変数

・$GLOBALS　— グローバルスコープで使用可能なすべての変数への参照
・$_SERVER　— サーバ情報および実行時の環境情報
・$_GET　— HTTP GET 変数
・$_POST　— HTTP POST 変数
・$_FILES　— HTTP ファイルアップロード変数
・$_REQUEST　— HTTP リクエスト変数
・$_SESSION　— セッション変数
・$_ENV　— 環境変数
・$_COOKIE　— HTTP クッキー

≪参考情報≫
定義済の変数
http://www.php.net/manual/ja/reserved.variables.php

シングルクオートとダブルクオートの扱いの違い

・ダブルクォートは文字列中の変数を展開する。
・シングルクォートは文字列中の変数を展開しない。

変数の展開を使わないときはシングルクォートを使ったほうが
処理速度が速くなります。
ダブルクォートは必要なとき以外は使わないほうがよいでしょう。

≪使用例≫

```
<?php
$fruit = "みかん";
echo "果物では $fruit が好きです。";
echo '果物では $fruit が好きです。';
?>
```

≪出力結果≫

```
果物では みかん が好きです。
果物では $fruit が好きです。
```

算術演算子(代数演算子) : 1 ／ 2

算術計算を行う場合に使用します。

記述	意味
$a + $b	$a と $b を足す
$a - $b	$a から $b を引く
$a * $b	$a と $b を掛ける
$a / $b	$a を $b で割る
$a % $b	$a を $b で割った余り

算術演算子(代数演算子) :２／２

≪使用例≫

```
<?php
    print '5 + 2 = ' . (5 + 2) . '<br>';
    print '5 - 2 = ' . (5 - 2) . '<br>';
    print '5 * 2 = ' . (5 * 2) . '<br>';
    print '5 / 2 = ' . (5 / 2) . '<br>';
    print '5 % 2 = ' . (5 % 2) . '<br>';
?>
```

≪出力結果≫

5 + 2 = 7
5 - 2 = 3
5 * 2 = 10
5 / 2 = 2.5
5 % 2 = 1

代入演算子：１／２
変数に値を代入する時に使用します。

記述	意味
$a = $b	$a に $b を代入する
$a += $b	$a に $a + $b を代入する
$a -= $b	$a に $a - $b を代入する
$a *= $b	$a に $a * $b を代入する
$a /= $b	$a に $a / $b を代入する
$a %= $b	$a に $a % $b を代入する
$a .= $b	$a に $b の値を連結して代入する

代入演算子:2／2

≪使用例≫

```
<?php
    print '$a = 5  : ' . ($a = 5) . '<br>';
    print '$a += 2 : ' . ($a += 2) . '<br>';
    print '$a -= 2 : ' . ($a -= 2) . '<br>';
    print '$a *= 2 : ' . ($a *= 2) . '<br>';
    print '$a /= 2 : ' . ($a /= 2) . '<br>';
    print '$a %= 2 : ' . ($a %= 2) . '<br>';
    print '$a .= 2 : ' . ($a .= 2) . '<br>';
?>
```

≪出力結果≫

$a = 5 : 5
$a += 2 : 7
$a -= 2 : 5
$a *= 2 : 10
$a /= 2 : 5
$a %= 2 : 1
$a .= 2 : 12

関係演算子：１／２

値の比較を行うための演算子です。TRUE(真)またはFALSE(偽)を返します。

記述	意味
$a == $b	$a と $b が等しい時にTRUE
$a === $b	$a と $b が等しく同じ型である場合にTRUE
$a != $b	$a と $b が等しくない場合にTRUE
$a <> $b	$a と $b が等しくない場合にTRUE
$a !== $b	$a と $b が等しくないか、同じ型でない場合にTRUE
$a < $b	$a が $b より小さい時にTRUE
$a > $b	$a が $b より大きい時にTRUE
$a <= $b	$a が $b より小さいか等しい時にTRUE
$a >= $b	$a が $b より大きいか等しい時にTRUE

関係演算子:2／2

≪使用例≫

```
<?php
    print '5 == 5      : ' . (5 == 5) . '<br>';
    print '5 === 5     : ' . (5 === 5) . '<br>';
    print '5 != 5      : ' . (5 != 5) . '<br>';
    print '5 <> 5      : ' . (5 <> 5) . '<br>';
    print '5 !== 5     : ' . (5 !== 5) . '<br>';
    print '5 < 5       : ' . (5 < 5) . '<br>';
    print '5 > 5       : ' . (5 > 5) . '<br>';
    print '5 <= 5      : ' . (5 <= 5) . '<br>';
    print '5 >= 5      : ' . (5 >= 5) . '<br>';
?>
```

≪出力結果≫

5 == 5 : 1
5 === 5 : 1
5 != 5 :
5 <> 5 :
5 !== 5 :
5 < 5 :
5 > 5 :
5 <= 5 : 1
5 >= 5 : 1

インクリメント／デクリメント演算子：１／２
変数の値を1つずつ増減させる時に使用する演算子です。

記述	意味
$a++	$a の値を返してから $a の値に 1 を加える。
++$a	$a の値に 1 を加えてから $a の値を返す。
$a--	$a の値を返してから $a の値から 1 を引く。
--$a	$a の値から 1 を引いてから $a の値を返す。

インクリメント／デクリメント演算子：２／２

≪使用例≫

```
<?php
    $a = 5;
    // 変数の値を表示してから1増やす
    print 'a : ' . $a++ . '<br>';
    // 変数を1増やしてから値を表示する
    print 'a : ' . ++$a . '<br>';
    // 変数の値を表示してから1減らす
    print 'a : ' . $a-- . '<br>';
    // 変数を1減らしてから値を表示する
    print 'a : ' . --$a . '<br>';
?>
```

≪出力結果≫

a : 5
a : 7
a : 7
a : 5

ビット演算子：1／2

記述	意味
$a & $b	$aと$bの論理積(AND)
a | b	$aと$bの論理和(OR)
$a ^ $b	排他的論理和(XOR: exclusive OR)
~$a	$aの否定(NOT)
a << n	aを0埋めしながらnビット左にシフトさせる
a >> n	aを符号を保持したままnビット右にシフトさせる

ビット演算子：2／2

≪使用例≫

```
<?php
   $a = 1;
   $b = 0;
   print '$a & $b  : ' . ($a & $b) . '<br>';
   print '$a | $b  : ' . ($a | $b) . '<br>';
   print '$a ^ $b  : ' . ($a ^ $b) . '<br>';
   print '~$a      : ' . (~$a) . '<br>';
   print '$a << $b : ' . ($a << $b) .'<br>';
   print '$a >> $b : ' . ($a >> $b) .'<br>';
?>
```

≪出力結果≫

```
  $a & $b : 0
  $a | $b : 1
  $a ^ $b : 1
  ~$a : -2
  $a << $b : 1
  $a >> $b : 1
```

論理演算子：1／2

記述	意味
$a && $b	$aと$bの論理積
$a and $b	$aと$bの論理積
$a || $b	$aと$bの論理和
$a or $b	$aと$bの論理和
$a xor $b	排他的論理和(XOR: exclusive OR)
! $a	$aの否定(NOT)

論理演算子:2／2

≪使用例≫

```
<?php
	$a = 1;
	$b = 0;
	// 論理積
	print '$a && $b  : ' . ($a && $b) . '<br>';
	// 論理和
	print '$a || $b  : ' . ($a || $b) . '<br>';
	// 排他的論理和
	print '$a xor $b : ' . ($a xor $b) . '<br>';
	// 否定
	print '!$a       : ' . (!$a) . '<br>';
?>
```

≪出力結果≫

```
$a && $b :
$a || $b : 1
$a xor $b : 1
!$a :
```

条件分岐：1／2

- if

if-else文は、条件に応じて処理を分岐させ、プログラムの流れをコントロール
することができます。

文法の説明

変数$aの値が「1」の場合は、処理1を実行し、変数$aの値が「2」の場合には、
処理2を実行します。
変数$aの値が「1」、「2」のどちらでもない場合は、処理3を実行します。

```
if ($a == 1) {
        処理1;
} elseif ($a == 2) {
        処理2;
} else {
        処理3;
}
```

条件分岐：2／2

- switch

式の値に応じて多数の分岐を実現できます。

文法の説明

評価しても一致するものが無い場合は、defaultキーワード後の処理を実行します。
break文は、switch文を抜ける役割をしますので、break文が無い場合はcase
キーワード内の処理実行後もswitch文を抜けません。

```
switch(式) {
    case 定数式1:
        処理1;
        break;
    case 定数式2:
        処理2;
        break;
    default:
        処理n;
}
```

繰り返し：１／４

・for
指定した回数だけ処理を繰り返す場合に用います。

文法の説明

```
for (初期状態; 終了条件; 継続処理) {
    処理;
}
```

繰り返し：２／４

・while
ある条件が真である間、ループを繰り返します。

文法の説明

```
while (条件式) {
    処理;
}
```

繰り返し：３／４

・do-while

do-while文は条件式の評価がループの最後に行われる為、最初の1回が必ず実行されます。

文法の説明

```
do {
    処理;
} while (条件式);
```

繰り返し：４／４

・foreach

・配列を扱う場合

```
foreach ( $配列 as $変数 ) {
    【繰り返す処理】;
}
```

・連想配列を扱う場合

```
foreach ( $配列 as $キー変数 => $値変数 ) {
    【繰り返す処理】;
}
```

・break, continueについて

break文が実行されると、ループが終了していなくてもループから抜けます。
continue文が実行されると、いったんループの先頭に戻ります。

配列

配列は、複数の値をまとめて扱える変数です。

文法の説明（一部）

・初期値を指定して配列を作成

$配列名 = array(値1,値2,値3・・・);

・配列の要素の値の利用

$配列[番号] = 値;
$変数 = $配列[番号];

・配列の要素数を取得

$変数 = count($配列);

include文とrequire文

include()
　引数に指定した外部ファイルでエラーが発生した場合でも、呼び出し元の
　（つまりinclude()関数を記述したファイル）後続処理は続行される。

require()
　外部ファイルでエラーが発生しても、呼び出し元の後続処理は停止する。

include_once()
　呼び出し元のスクリプトが一回読み込まれていれば、その後、何回呼び出し元
　のスクリプトが実行されてもinclude_once()関数の引数に指定されたスクリプトは
　一回しか実行されない。
　つまり、関数の再定義や変数の再計算は行われない。include()と同様で、引数の
　スクリプトでエラーが発生しても呼び出し元の後続処理は続行される。

require_once()
　処理はinclude_onceと同様で、require_once()関数の引数に指定したスクリプトで
　エラーが発生した場合は後続処理を停止する。

関数

・function

引数はカンマ「,」区切りで複数指定できます。
戻り値を指定する場合は「return」を使用します。

≪構文≫

```
function【関数名】(【引数】)
{
    処理

    return【戻り値】;
}
```

データベース
基本的PDOを使うことを推奨します。

・**データベース毎の関数**
dBase、DB++、FrontBase、ilePro、Firebird/InterBase、nformix、IBM DB2 (IBM DB2、Cloudscape および Apache Derby)、ngres、MaxDB、mSQL、Mssql (Microsoft SQL Server)、MySQL、Mysqli (MySQL 改良版拡張モジュール)、OCI8 (Oracle OCI8)、Ovrimos SQL、Paradox (Paradox ファイルアクセス)、PostgreSQL、SQLite、SQLite3、Sybase)

・**PDO**
≪参考情報≫
PDOでサクサクDB開発
http://codezine.jp/article/detail/433?p=1

・**ADOdb**
≪参考情報≫
ADOdbでサクサクDBアクセス
http://codezine.jp/article/detail/48

・**PEAR::DB、PEAR::MDB2**
≪参考情報≫
PEAR MDB2でPHPからデータベースを操作する
http://codezine.jp/article/detail/2480?p=1

例外処理

≪構文≫

```
try
{
    // 処理
}
catch (【例外クラス名】【変数】)
{
   // 例外発生時の処理
}
```

≪使用例≫

```
<?php
  function inverse($x) {
    if (!$x) {
        throw new Exception('ゼロによる除算。');
    }
    else return 1/$x;
  }

  try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
  } catch (Exception $e) {
    echo '捕捉した例外: ',  $e->getMessage(), "\n";
  }

  // 実行は継続される
  echo 'Hello World';
?>
```

≪出力結果≫

```
 0.2
 捕捉した例外: ゼロによる除算。
 Hello World
```

セッション

・セッション関数

session_cache_expire — 現在のキャッシュの有効期限を返す
session_cache_limiter — 現在のキャッシュリミッタを取得または設定する
session_commit — session_write_close のエイリアス
session_decode — 文字列からセッションデータをデコードする
session_destroy — セッションに登録されたデータを全て破棄する
session_encode — 現在のセッションデータを文字列としてエンコードする
session_get_cookie_params — セッションクッキーのパラメータを得る
session_id — カレントのセッション ID を取得または設定する
session_is_registered — 変数がセッションに登録されているかどうかを調べる
session_module_name — 現在のセッションモジュールを取得または設定する
session_name — 現在のセッション名を取得または設定する
session_regenerate_id — 現在のセッションIDを新しく生成したものと置き換える
session_register — 現在のセッションに1つ以上の変数を登録する
session_save_path — 現在のセッションデータ保存パスを取得または設定する
session_set_cookie_params — セッションクッキーパラメータを設定する
session_set_save_handler — ユーザ定義のセッション保存関数を設定する
session_start — セッションデータを初期化する
session_unregister — 現在のセッションから変数の登録を削除する
session_unset — 全てのセッション変数を開放する
session_write_close — セッションデータを書き込んでセッションを終了する

クロスサイトスクリプティング（XSS）対策

HTMLをエスケープする関数に**htmlspecialchars**と**htmlentities**があります。
意図通りにエスケープ処理を実行する為には第2、第3引数を指定する必要があります。
これらの関数の違いは変換する文字数です。**htmlspecialchars**はデフォルトで4つの文字（<、>、&、"）をHTML の表現形式に変換します。対して、**htmlentities**はデフォルトで先の4つを含む100個の文字を変換します。より厳密にエスケープしたいなら、**htmlentities**を使う必要があります。
第3引数に**mb_internal_encoding()**を使うときはちゃんとサーバの **mbstring.internal_encoding** の値を確認することが必要です。
なお、環境によっては第3引数にを直接文字コードを設定する必要な場合もあります。
(PHP5.3 / 6からhtmlspecialcharsとhtmlentitiesの仕様が変わるらしいです。)

≪使用例1≫

```
<?php
  $str = $_POST['sample'];
?>
<input type='text' name='sample'
 value='<?php echo htmlentities($str, ENT_QUOTES, mb_internal_encoding()) ?>' />
```

≪使用例2≫

```
<?php
  $str = $_POST['sample'];
?>
<input type='text' name='sample'
 value='<?php echo htmlentities($str, ENT_QUOTES, 'UTF-8') ?>' />
```

SQLインジェクション対策

SQLインジェクションはすべての変数をエスケープするか，すべてのクエリをプリペアードクエリとして実行すれば100%防げる脆弱性です。
いまだに「**シングルクォートだけ対応すれば問題無い！**」とか「**文字列だけやれば問題無い！**」と自信満々に言っている人が多いのも事実です。みなさんはこのような人にならないようにしてください。

≪どうしても一部の変数だけの対応となる場合≫
問題が起こった場合に、対象のプログラムに対してその場しのぎの対応を行う必要があることをはっきりと言う必要があります。

マルチバイト関数：1／4

関数名	処理
mb_check_encoding	文字列が、指定したエンコーディングで有効なものかどうかを調べる
mb_convert_case	文字列に対してケースフォルディングを行う
mb_convert_encoding	文字エンコーディングを変換する
mb_convert_kana	カナを("全角かな"、"半角かな"等に)変換する
mb_convert_variables	変数の文字コードを変換する
mb_decode_mimeheader	MIME ヘッダフィールドの文字列をデコードする
mb_decode_numericentity	HTML 数値エンティティを文字にデコードする
mb_detect_encoding	文字エンコーディングを検出する
mb_detect_order	文字エンコーディング検出順序を設定あるいは取得する
mb_encode_mimeheader	MIMEヘッダの文字列をエンコードする
mb_encode_numericentity	文字を HTML 数値エンティティにエンコードする
mb_ereg_match	マルチバイト文字列が正規表現に一致するか調べる
mb_ereg_replace	マルチバイト文字列が正規表現に一致するか調べる

マルチバイト関数：2／4

関数名	処理
mb_ereg_replace	マルチバイト文字列に正規表現による置換を行う
mb_ereg_search_getpos	次の正規表現検索を開始する位置を取得する
mb_ereg_search_getregs	マルチバイト文字列が正規表現に一致する部分があるか調べる
mb_ereg_search_init	マルチバイト正規表現検索用の文字列と正規表現を設定する
mb_ereg_search_pos	指定したマルチバイト文字列が正規表現に一致する部分の位置と長さを返す
mb_ereg_search_regs	指定したマルチバイト文字列が正規表現に一致する部分を取得する
mb_ereg_search_setpos	次の正規表現検索を開始する位置を設定する
mb_ereg_search	指定したマルチバイト文字列が正規表現に一致するか調べる
mb_ereg	マルチバイト文字列に正規表現マッチを行う
mb_eregi_replace	マルチバイト文字列に大文字小文字を区別せずに正規表現による置換を行う
mb_eregi	マルチバイト文字列に大文字小文字を区別しない正規表現マッチを行う
mb_get_info	mbstring の内部設定値を取得する
mb_http_input	HTTP 入力文字エンコーディングを検出する

マルチバイト関数：3／4

関数名	処理
mb_http_output	HTTP 出力文字エンコーディングを設定あるいは取得する
mb_internal_encoding	内部文字エンコーディングを設定あるいは取得する
mb_language	現在の言語を設定あるいは取得する
mb_list_encodings	サポートするすべてのエンコーディングの配列を返す
mb_output_handler	出力バッファ内で文字エンコーディングを変換するコールバック関数
mb_parse_str	GET/POST/COOKIE データをパースし、グローバル変数を設定する
mb_preferred_mime_name	MIME 文字設定を文字列で得る
mb_regex_encoding	現在の正規表現用のエンコーディングを文字列として返す
mb_regex_set_options	マルチバイト正規表現関数のデフォルトオプションを取得または設定する
mb_send_mail	エンコード変換を行ってメールを送信する
mb_split	マルチバイト文字列を正規表現により分割する
mb_strcut	文字列の一部を得る
mb_strimwidth	指定した幅で文字列を丸める

マルチバイト関数：4／4

関数名	処理
mb_stripos	大文字小文字を区別せず、文字列の中で指定した文字列が最初に現れる位置を探す
mb_stristr	大文字小文字を区別せず、文字列の中で指定した文字列が最初に現れる位置を探す
mb_strlen	文字列の長さを得る
mb_strpos	文字列の中に指定した文字列が最初に現れる位置を見つける
mb_strrchr	別の文字列の中で、ある文字が最後に現れる場所を見つける
mb_strrichr	大文字小文字を区別せず、別の文字列の中である文字が最後に現れる場所を探す
mb_strripos	大文字小文字を区別せず、文字列の中で指定した文字列が最後に現れる位置を探す
mb_strrpos	文字列の中に指定した文字列が最後に現れる位置を見つける
mb_strstr	文字列の中で、指定した文字列が最初に現れる位置を見つける
mb_strtolower	文字列を小文字にする
mb_strtoupper	文字列を大文字にする
mb_strwidth	文字列の幅を返す
mb_substitute_character	置換文字を設定あるいは取得する
mb_substr_count	部分文字列の出現回数を数える
mb_substr	文字列の一部を得る

オブジェクト指向(1)

・**クラス**

実際のプログラミングでは、クラスを基に「インスタンス」を生成します。
また、既存のクラスをもとに新しいクラスを定義することができます。
ただし、多重継承はサポートしていません。
Javaなどの他の言語で使用できるオーバーロードはできないようです。

≪構文≫

```
[abstract | final] class【クラス名】[extends【親クラス名】]
{
    処理
}
[abstract | final] class【クラス名】[implements【インターフェイス名】]
{
    処理
}
```

キーワード	説明
abstract	抽象クラスの指定
final	継承不可であることの指定
class	クラス定義の開始
extends	継承元のクラス指定
implements	実装するインターフェイス名の指定

オブジェクト指向(2)

・**アクセス修飾子**

キーワード	説明
public	公開されている
protected	継承されたクラスのみ(子クラス)に公開
private	非公開：同じクラス内のみ参照可能

・**インターフェイス**

≪構文≫

```
interface【インターフェイス名】
{
  【アクセス修飾子】function【関数名】();
  【アクセス修飾子】function【関数名】(【引数】);
}
```

オブジェクト指向(3)
マジックメソッドの種類

マジックメソッド名	概要
__construct	コンストラクタ
__destruct	デストラクタ
__call	メソッドのオーバーライド
__get	メンバー変数の値の取得
__set	メンバー変数の値の設定
__isset	isset関数に渡された時の動作を定義
__unset	unset関数に渡された時の動作を定義
__sleep	serialize関数でシリアライズされる際の動作を定義
__wakeup	unserialize関数でシリアライズされる際の動作を定義
__toString	クラスが文字列に変換される際の動作を定義
__set_state	var_export関数に渡された時の動作を定義
__clone	オブジェクトのコピー

オブジェクト指向(4)

・**プロパティ**

オブジェクト内に保持する変数を定義するものです。
アクセス修飾子のprivate / protected / publicが使用可能。
$this変数を使うことによって、プロパティと
ローカル変数を明確に区別することができます。

≪構文≫

【アクセス修飾子】 $【変数】;

・**メソッド**

メソッドは、オブジェクトを操作するものです。
アクセス修飾子のprivate / protected / publicが使用可能。
引数はカンマ「,」区切りで複数指定できます。
通常の関数との違いは以下の通りです。

・クラススコープを持つ
・オブジェクト自身を指す$this変数が定義されている。
・アクセスするには、オブジェクト経由または「クラス名::メソッド名]の文法を利用する。

≪構文≫

```
【アクセス修飾子】 function 【メソッド名】 (【引数】) {
    処理
}
```

オブジェクト指向(5)

・**コンストラクタ**

オブジェクトを生成する時点で、オブジェクトに対する初期処理を行うことができます。

≪構文≫

```
// コンストラクタ (constructの前の「_」[アンダーバー]は2つです。)
function __construct (【引数】) {
    処理
}
```

・**デストラクタ**

オブジェクトが消滅する際に自動的に呼ばれますので、オブジェクトが消滅する際に必ず実行したい処理がある場合に使用します。

≪構文≫

```
// デストラクタ (constructの前の「_」[アンダーバー]は2つです。)
function __destructor (【引数】) {
    処理
}
```

テンプレートエンジン

デザインとロジックの完全な分離が可能
(テンプレートエンジンを使わないとできないわけではありません。)
最初の読み込み時はPHPのコードに変換する処理が行われる関係で
若干表示が遅いという問題がある。
(JavaのJSPのような感じであると思います。)

≪テンプレートエンジン≫
・Smarty(スマーティー)
http://www.smarty.net/
≪書籍≫
Smarty入門(翔泳社、2,800円+税)

フレームワーク

・Zend Framework(ゼンド　フレームワーク)
PHP開発元のZend Technologies社が中心となって開発。書籍が多数あり(翔泳社など)。
http://framework.zend.com/manual/ja/

・CakePHP(ケイクピーエイチピー)
書籍が多数ある(技術評論社など)。Ruby On Railsの影響をかなり受けている。
http://cakephp.jp/

・symfony(シンフォニー)
書籍あり(技術評論社)。
http://symfony.jp/

・Seasar.PHP(シーサードットピーエイチピー)
Maple(メイプル), S2Container.PHP5, S2Dao.PHP5, S2AnA.PHP5, S2Base.PHP5
http://www.seasar.org/#seasar.php

・Ethna(えすな)
GREEの構築で使用されている。書籍あり(技術評論社)。
http://ethna.jp/
EthnaをGREEに!
http://labs.gree.jp/Top/Document/20060518.html

・CodeIgniter(コードイグナイタ/イグニター)
書籍あり(翔泳社)。
http://www.codeigniter.jp/

・Piece Framework(ピース　フレームワーク)
http://piece-framework.com/ja/index.html

最後に

PHPはプログラミングの基礎知識があれば難しくありません。
そのためセキュリティを考慮していないアプリケーションが多いのも事実です。
個人的には、PHPのWebアプリケーションに脆弱性が多い原因の1つとして書店で入手できるPHPの本がセキュリティのことを考慮していない物があまりにも多いことと関係があると思っています。

(補足)PHP関連書籍

・PHPポケットリファレンス(技術評論社、2,380円+税)
・入門PHPセキュリティ(オライリージャパン、1,800円+税)
・Webアプリセキュリティ対策入門(技術評論社、2,980円+税)
・ウェブアプリケーションセキュリティ(データハウス、4,600円+税)
・はじめてのPHP言語プログラミング入門(技術評論社、2,800円+税)
・はじめてのPHP5 (オライリージャパン、2,800円+税)
・プログラミングPHP 第2版(オライリージャパン、3,800円+税)
・PHP4→PHP5移行ガイド(ソフトバンク、2,800円+税)
・Smarty入門(翔泳社、2,800円+税)
・SQLite入門(翔泳社、2,800円+税)
・Ethna X PHP(技術評論社、1,980円+税)
・symfony X PHP(技術評論社、1,980円+税)
・Fast PHP(技術評論社、1,980円+税)
・CakePHPガイドブック(マイコミ、3,200円+税)
・PHPフレームワークZendFramework入門(ソーテック、2,780円+税)
・PHPよるデザインパターン入門(秀和システム、3,000円+税)
・PHPライブラリコレクション(翔泳社、2,400+税)
・PHP Hacks(オライリージャパン、3,600円+税)

(補足)PHP関連サイト

- PHP.net
  http://php.net/
- PHPマニュアル
  http://www.php.net/manual/ja/
- XAMPP
  http://www.apachefriends.org/jp/xampp.html
- PEAR
  http://pear.php.net/index.php
- Smarty
  http://www.smarty.net/
- PHPユーザー会
  http://www.php.gr.jp/
- PHPプロ
  http://www.phppro.jp/

(補足) PHPの開発ツール

- VS.Php
  http://www.asial.co.jp/vsphp/
- Delphi for PHP
  http://www.codegear.com/jp/products/delphi/php
- ZendStdio for Eclipse
  http://www.zend.co.jp/
- PHPエディタ (フリーソフト)
  http://phpspot.net/php/phpeditor.html