

# テンプレートの基礎

trustBee 山本 勇

<http://www.trustbee.com/>  
[yamamoto@trustbee.com](mailto:yamamoto@trustbee.com)



# 本セッションについて

- PHP の書き方
- テンプレートとは
- 簡単な自作テンプレート
- 色々なテンプレートエンジンの紹介
- 便利なライブラリの紹介



# PHP の書き方について

- PHP は HTML に埋め込めるスクリプト言語。
  - PHP で echo 文や printf 関数を使用して、HTML を出力する。
  - HTML に PHP を埋め込む形で記述する。



# コードサンプル

PHP スクリプトで HTML を出力    HTML に埋め込んだ PHP スクリプト

```
<?php
$title = "Hello, World!";
echo "<html>\n";
echo "<head>\n";
echo "<title>$title</title>\n";
echo "</head>\n";
echo "<body>\n";
echo "<h3>$title</h3>";
echo "</body>\n";
echo "</html>\n";
?>
```

```
<html>
<head>
<title>Hello, World!</title>
</head>
<body>
<h3>Hello, World!</h3>
</body>
</html>
```

```
<html>
<head>
<?php $title = "Hello, World!"; ?>
<title>
<?php
echo $title;
?>
</title>
</head>
<body>
<h3><?= $title ?></h3>
</body>
</html>
```

出力された HTML



# 一つの PHP スクリプトで HTML を出力するメリットとデメリット

- メリット

- 一つのスクリプトで、プログラム処理と HTML 出力を管理できる。

- デメリット

- 出力形式（デザイン）の変更に柔軟に対応できない。

# テンプレートとは

- テンプレートとはロジック（ビジネスロジック）とデザイン（プレゼンテーション）の分離を行う手法の一つ。

- → コンポーネント指向

PHP スクリプト

```
<?php
$title = "テンプレートの基礎";
$author = "Isamu Yamamoto";
$mail = "yamamoto@trustbee.com";
$date = "2003/08/30";
include("template.php");
?>
```

テンプレート

```
<html>
<head>
<title><?= $title ?></title>
</head>
<body>
<h3><?= $title ?></h3>
<div align='right'>
```

テンプレートファイルを読み込み

HTML

```
<html>
<head>
<title>テンプレートの基礎</title>
</head>
<body>
<h3>テンプレートの基礎</h3>
<div align='right'>
```



# ロジックとデザイン

- ロジック: データベースに対する処理など、各種の操作を実行するプログラム部分。プログラマの担当部分
  - → PHP スクリプト
- デザイン: プログラムの実行結果を、HTML などの形でユーザに対して出力する部分。デザイナーの担当部分
  - → テンプレート

# なぜテンプレートを使う必要があるのか？

- 中規模程度のシステムに、PHP が使われ始めている。  
規模が大きくなると、作業の効率化のためにロジック開発とデザインを分業する必要が出てくる。  
そこで、ソースを分離するためにテンプレートを使用する。
  - テンプレートを使わないと
    - デザインの HTML が決定しないと、スクリプトが完成しない => NG
  - テンプレートを使うと
    - 事前に使用する変数名などを決め、並行して開発を進める => Good



# 簡単なテンプレートの例 その1

- PHP による変数呼び出し

ロジック部分

```
<?php  
  
$title = "テンプレートの基礎";  
$author = "Isamu Yamamoto";  
$mail = "yamamoto@trustbee.com";  
$date = "2003/08/30";  
  
include("template.inc");  
?>
```



テンプレート部分

```
<html>  
<head>  
<title><?= $title ?></title>  
</head>  
<body>  
<h3><?= $title ?></h3>  
<div align='right'>  
By <a href='mailto:<?= $mail ?>'>  
<?= $author ?></a><br>  
<i><?= $date ?></i>  
</div>  
</body>  
</html>
```

実行結果



# 簡単なテンプレートの例 その2

- 繰り返される情報の出力

- データベースの情報の出力など、繰り返される形式の出力例。

## ロジック部分

```
<?php
// 配列に値を代入
$data = array(
    'りんご'=>8,
    'みかん'=>12,
    'グレープフルーツ'=>6);

// 合計を取得
$sum = array_sum($data);

// テンプレートの読み込み
require("template.inc");
?>
```

## テンプレート部分

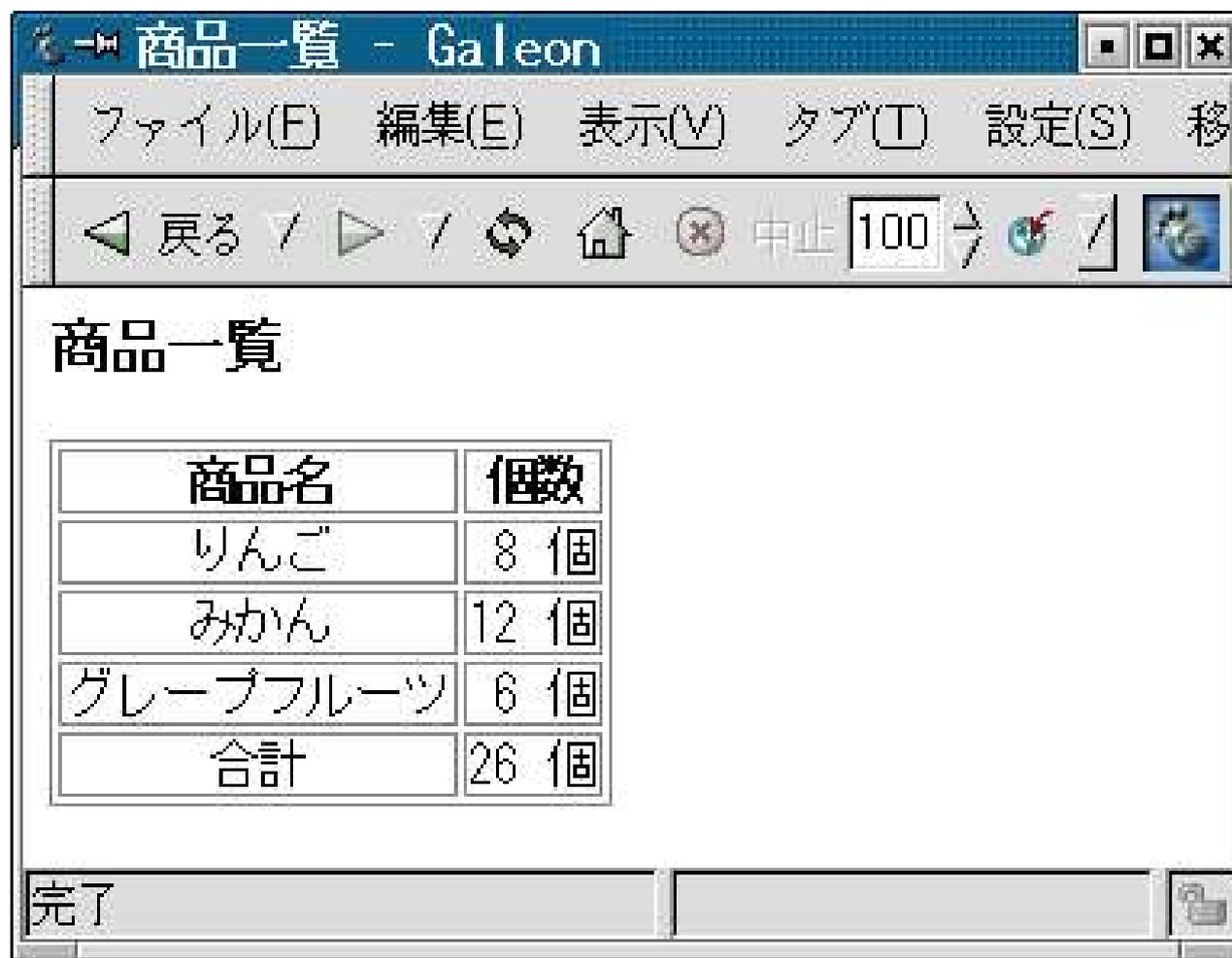
```
<h3>商品一覧</h3>
<table border=' 1'>
<?php
foreach ($data as $key => $val) {
    echo " <tr>¥n";
    echo " <td align=' center'>$key</td>¥n";
    echo " <td align=' right'>$val 個</td>¥n";
    echo " </tr>¥n";
}
?>
<tr>
    <td align=' center'>合計</td>
    <td align=' right'><?= $sum ?> 個</td>
</tr>
</table>
```

配列をループ



# 簡単なテンプレートの例 その2

## 実行結果



# 簡単なテンプレートの例 その3

- 共通のヘッダ、フッタ

- メニューボタンなど、複数のスクリプトで共通のデザインの実出力例。

## ヘッダ (header.inc)

```
<html>
<head>
<title><?= $title ?></title>
</head>
<body>
<div align='right'>
<img src='php.gif'>
</div>
<hr>
```

## メインテンプレート (template.inc)

```
<!-- Header -->
<?php require('header.inc'); ?>
<!-- Main -->
<h3><?= $title ?></h3>
<?= $msg ?>
<!-- Footer -->
<?php require('footer.inc'); ?>
```

## フッタ (footer.inc)

```
<hr>
<div align='right'>
Written by Isamu Yamamoto
</div>
</html>
```

## ロジック部分

```
<?php
$title = 'ロジック A';
$msg = 'これはロジック Aの実行結果です。';
require("template.inc");
?>
```



# 簡単なテンプレートの例 その3

## 実行結果



# 簡単なテンプレートの例 その4

- 正規表現による逐次変数置換

```
<?php
$title = "テンプレートの基礎";
$author = "Isamu Yamamoto";
$mail = "yamamoto@trustbee.com";
$date = "2003/08/30";

$fp = fopen("template.inc", "r") or die("File open error");
while ($buf = fgets($fp, 1024)) {
    $buf = preg_replace ("/{(\w+)}/e", "' '$a¥¥1'", $buf);
    eval ("¥$buf = ¥"$buf¥");
    echo $buf;
}
fclose($fp);
?>
```

ロジック部分

{title} を \$title に置換

\$buf = "<title>\$title</title>";  
を実行

```
<html>
<head>
<title> {title} </title>
</head>
<body>
<h3> {title} </h3>
<div align='right' >
By <a href='mailto: {mail}' > {author} </a> <br>
<i> {date} </i>
</div>
</body>
</html>
```

テンプレート



# テンプレートエンジンを使ってみる

- 様々な高機能なテンプレートエンジンが公開されている。  
Smarty もその一つ。  
ほかにも HTML\_Template\_IT や  
HTML\_Template\_Sigma などがある。
- 国内でも、多くの人々が独自のテンプレートライブラリを公開している。

# Smarty の紹介

- 本家サイト

- <http://smarty.php.net/>

- マニュアルの日本語訳

- <http://sunset.freespace.jp/smarty/>



# Smarty のインストール

- インストール

- ソースを展開する。

```
$ tar xzf Smarty-2.5.0.tar.gz
```

- Smarty-2.5.0/libs/ 以下をインクルードパスのディレクトリなどに移動する。

使用時は、Smarty.class.php を呼び出して使用する。

```
require 'Smarty-2.5.0/libs/Smarty.class.php';
```

- Smarty のディレクトリ（デフォルトの場合）

- configs            設定ファイル格納ディレクトリ。（必須ではない）
- templates        テンプレート格納ディレクトリ。
- templates\_c      コンパイル済みテンプレート格納ディレクトリ。  
Web サーバの実行ユーザでの書き込み権限を与える必要がある。



# Smarty の使用例（その 1）

- 簡単なテンプレートの例（その 1）を、Smarty を使って書いてみる。  
templates および templates\_c ディレクトリを作成し、templates\_c ディレクトリの権限を変更する。

```
$ mkdir templates templates_c  
$ chmod 777 templates_c
```

## ロジック

```
<?php  
// Smarty ライブラリの読み込み  
require 'Smarty-2.5.0/libs/Smarty.class.php';  
  
// オブジェクトの宣言  
$smarty = new Smarty;  
  
// 変数の割り当て  
$smarty->assign("title", "Hello, World!");  
  
// テンプレートの読み込みと表示  
$smarty->display('template.tpl');  
?>
```

## テンプレート

```
<html>  
<head>  
<title>{$title}</title>  
</head>  
<body>  
<h3>{$title|upper}</h3>  
</body>  
</html>
```

`title` を大文字に変換



# Smarty の使用例 ( その 1 ) の キャッシュ

- template\_c ディレクトリに生成された  
キャッシュファイル

```
<?php /* Smarty version 2.5.0, created on 2003-08-24 05:45:31
      compiled from 3-1.tpl */ ?>
<?php $this->_load_plugins(array(
array('modifier', 'upper', 'template.tpl', 6, false),)); ?><html>
<head>
<title><?php echo $this->_tpl_vars['title']; ?>
</title>
</head>
<body>
<h3><?php echo $this->_run_mod_handler('upper', true, $this->_tpl_vars['title']); ?>
</h3>
</body>
</html>
```



# Smarty の使用例（その2）

- 簡単なテンプレートの例（その2）を、Smarty を使って書いてみる。

ロジック

```
<?php
$data = array('りんご'=>8,
              'みかん'=>12,
              'グレープフルーツ'=>6);
$sum = array_sum($data);

// Smarty ライブラリの読み込み
require 'Smarty-2.5.0/libs/Smarty.class.php';

// オブジェクトの宣言
$smarty = new Smarty;

// 変数の割り当て
$smarty->assign("data", $data);
$smarty->assign("sum", $sum);

// テンプレートの読み込みと表示
$smarty->display('template.tpl');
?>
```

テンプレート（一部抜粋）

```
<table border='1'>
<tr>
  <th>商品名</th>
  <th>個数</th>
</tr>
{foreach from=$data key=key item=num}
<tr>
  <td align='center'>{$key}</td>
  <td align='right'>{$num} 個</td>
</tr>
{/foreach}
<tr>
  <td align='center'>合計</td>
  <td align='right'>{$sum} 個</td>
</tr>
</table>
```



# Smarty の使用例 ( その 2 ) の キャッシュ

- template\_c ディレクトリにできたキャッシュファイル  
(一部抜粋)

```
<table border=' 1' >
  <tr>
    <th> 商品名 </th>
    <th> 個数 </th>
  </tr>
  <?php if (count((array)$this->_tpl_vars[' data' ])) :
    foreach ((array)$this->_tpl_vars[' data' ] as
      $this->_tpl_vars[' key' ] => $this->_tpl_vars[' num' ]):
  ?>
  <tr>
    <td align=' center' ><?php echo $this->_tpl_vars[' key' ]; ?>
  </td>
    <td align=' right' ><?php echo $this->_tpl_vars[' num' ]; ?>
    個 </td>
  </tr>
  <?php endforeach; endif; ?>
```

配列の個数をチェックし、  
foreach でループしている



# Smarty のテンプレートのエラー表示の例

- テンプレートのエラー表示例

- テンプレート内の {/foreach} を {/foreac} と間違った場合のエラー表示例

Parse error: parse error in

/path/**templates\_c/%%471/%%471519675/template.tpl** on line 33

- テンプレート自身ではなく、コンパイル済みのテンプレートにおける行数で表示されてしまい、エラー原因の箇所が特定できない。

実際のエラー原因:

```
<?php endforeach; endif; ?> と変換されるべきなのに、  
<?php echo $this->_plugins['function']['/foreac'][0]  
(array(), $this) ; ?> と
```

なってしまう、foreachループ、if文が正常に閉じられていないため。

- Smarty のデバッグコンソール

- `$smarty->debugging = true;`

- JavaScript によりデバッグコンソールが表示され、格納した変数の値などを確認することが出来る。しかし、上記のエラーの原因は分からない。



# AwesomeTemplateEngine

- <http://www.pinkgoblin.com/index.php?view=scripts>
- クラス本体は 10 行にも満たない、非常に軽いテンプレート。  
クラス自体では単にテンプレートをインクルードしているだけ。  
テンプレート内での PHP コードで、テーブルの行のループなどを処理している。  
単に `include("テンプレートファイル")` とするのと違いはない。



# AwesomeTemplateEngine の本体

- AwesomeTemplateEngine のソース

```
<?php
/* AwesomeTemplateEngine.class.php */
class AwesomeTemplateEngine {
    var $templatePath;
    function AwesomeTemplateEngine($templatePath) {
        $this->templatePath=$templatePath;
    }
    function parseTemplate($data, $template) {
        include($this->templatePath.$template);
    }
}

/* Awesome isn't it? */
?>
```

テンプレートパスの保存用に変数を宣言

テンプレートパス設定関数

テンプレート読み込み関数

include 文でテンプレートを読み込む



# その他のテンプレート

- HTML\_Template\_IT
  - テーブル形式など、繰り返す形の出力に適しているテンプレート。キャッシュ機能は無い。
- HTML\_Template\_Sigma
  - テンプレートの書式は、HTML\_Template\_ITと互換性がある。  
設定により、キャッシュ機能も使用できる。  
ただ、キャッシュの書式はPHPスクリプトではなく**独自形式**で保存しているので、キャッシュによる速度向上の効果は少ない。  
むしろ、キャッシュファイル生成の時間がかからない分、キャッシュ不使用の方が速い場合もあった。

独自形式：変数部分をシリアル化して書き込む形式



# その他のテンプレート（続き）

- HTML\_Template\_Xipe

- テンプレート内に PHP タグ (<?php ?>) が入るのを避けるのが元々の目的。  
単純に、“{ }”を“<?php ?>”に置換している。  
置換済みのファイルをキャッシュする。  
制御構造として、TagLib という機能を実装している。



# 各種テンプレートエンジンの違い

- (1) 変数呼び出しの書式
  - %var% {var} など
- (2) 独自の制御構造
  - if 文や for 文の実装
- (3) コンパイル済みテンプレートのキャッシュ
  - 変数置換後のテンプレートをキャッシュする機能

# 各種テンプレートエンジンの違い (ツツコミ)

- (1) 変数呼び出しの書式

- %var% {var} など

- 書式の違いにどれほどの意味があるのか？
    - テンプレートエンジン間で互換性が無いため、デザイナーはそれぞれのテンプレートの形式で書かなければならない。

- (2) 独自の制御構造

- if 文や for 文の実装

- PHP の制御構造とは別に、テンプレートエンジン独自の制御構造を覚えなければならない。
    - エラー時のメッセージが適切でないため、デバッグが困難になる。



# 各種テンプレートエンジンの違い (ツッコミ)(続き)

- (3) コンパイル済みテンプレートのキャッシュ
  - 変数置換後のテンプレートをキャッシュする機能
    - キャッシュとはいえ、単なる PHP スクリプト。  
初回の実行時にテンプレートを読み込み、PHP 形式のキャッシュファイルを生成する。  
2 回目以降はキャッシュファイルを読み込むことで時間の短縮になる。
    - 始めから、PHP 形式のテンプレートを読み込むのと何が違う？

# 逐次変数置換型テンプレート エンジンの欠点

- HTML\_Template\_IT, HTML\_Template\_Sigma など
- 実行処理に時間がかかる。
  - テンプレートを使わず PHP スクリプトして処理した場合と比較すると、HTML\_Template\_IT では 100 行のテーブル出力で約 12 倍、1000 行では約 26 倍の時間がかかった。  
HTML\_Template\_Sigma でも、1000 行で 10 倍の時間がかかる
- 制御構造を使えない。
  - if 文などの制御構造が使えない場合、エラーメッセージはあらかじめ変数に格納する必要がある。



# ・コンパイル型のテンプレートエンジンの欠点

- Smarty, HTML\_Template\_Xipe など
- 処理に時間がかかる
  - テンプレートエンジンの読み込み・処理時間に加え、初回実行時はキャッシュファイル生成の時間が必要になる。
- テンプレートのエラー時のデバッグが困難
  - エラーが発生するのが、コンパイル後のスクリプト内のため、エラーメッセージや行が参考にならない。
- 新たな文法を覚えなければならない
  - 独自の制御構造を持つ場合、PHP とは別の文法を覚えなければならない。



# テンプレートエンジンの比較

## • テンプレートエンジン比較表

テンプレートエンジン	変数置換	キャッシュ	速さ
AwesomeTemplateEngine	×	×	◎
Smarty	×	○	○
HTML_Template_IT	○	×	×
HTML_Template_Sigma	○	△	×
HTML_Template_Xipe	×	○	○

### 変数置換:

実行時(キャッシュ機能がある場合は、二回目以降)の処理にて  
逐次変数置換が行なわれるか

○: 逐次変数置換される × 逐次: 変数置換されない

### キャッシュ:

キャッシュファイルを生成するか

○: キャッシュファイルが必須 △: オプションで選択可能

×: キャッシュファイルを生成しない

### 速さ:

実行時(キャッシュ機能がある場合は、二回目以降)の速さ

◎: とても速い ○: 速い ×: 遅い



# テンプレートエンジンの使用と 選択のポイント

- デザインとロジックの分離を徹底する必要があるか？
  - テンプレートを使うメリットと使わないメリットを考慮する。
- デザイナーが PHP コードを修正しても問題が無いか？
  - 「テンプレート」とは、PHP コードを一切の含まないものであるべきなのか。
  - あらかじめ決めた変数名だけ使用し、デザイナーとプログラマーが、お互いの領域に一切足を踏み入れない状態にすべきか。



# テンプレートエンジンの使用と 選択のポイント（続き）

- WYSIWYG な HTML エディタで編集可能である必要があるか？
  - ヘッダやフッタが別ファイルの場合は、編集できないエディタもある。  
まして、ページ全体が一つのテーブルの場合はさらに別ファイルに分けるのが困難になる。

「WYSIWYG」（ういじういぐ）  
What You See Is What You Get の略。  
直訳では「見たものが、手に入るもの」の意味で、  
ディスプレイ画面に表示された状態のものが、  
そのまま印刷や画面出力されること。



# HTML\_QuickForm

- PEAR からインストール可能。
- フォームの自動生成を行う便利なライブラリ。
- 入力値のチェック機能もあり、JavaScript によるクライアントサイドのチェックも行える。  
チェック関数は、自作関数も使用可能。



# HTML\_QuickForm の使用例

```
<?php
// クラスの読み込み
require_once ("HTML/QuickForm.php");

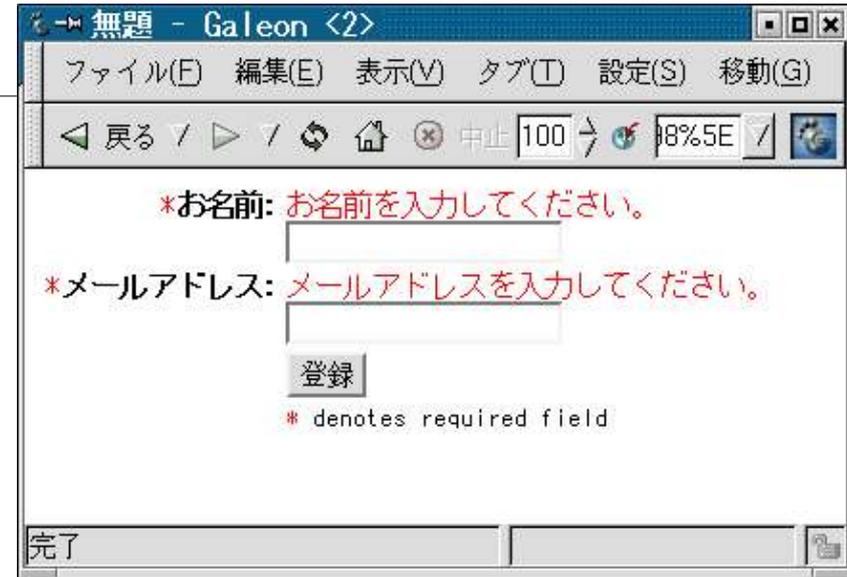
// オブジェクトの宣言
$form = new HTML_QuickForm('form.php', 'get');

// エレメントの定義
$form->addElement('text', 'name', 'お名前:');
$form->addElement('text', 'mail', 'メールアドレス:');
$form->addElement('submit', 'submit', '登録');

// ルールの指定
$form->addRule('name', 'お名前を入力してください。', 'required');
$form->addRule('name', '3文字以上で入力してください。', 'minlength', 3);
$form->addRule('mail', 'メールアドレスを入力してください。', 'required');
$form->addRule('mail', 'メールアドレスが正しくありません。', 'email');

// 入力値のチェック
if ($form->validate()) {
    // 正しく入力された時の処理
}

// フォーム出力
$form->display();
?>
```



実行例

スクリプト



# Pager\_Sliding

- PEAR からインストール可能。
- ページ移動用のリンクを生成するライブラリ。
- 生成されたリンク用の HTML 文は、配列としても取得できるのでテンプレートでも使用できる。

# Pager\_Sliding の使用例

## • 2 の階乗の表示例

```
<?php
require_once 'Pager/Sliding.php';

// データを生成
for ($i=0; $i<=256; $i++)
    $data[] = pow(2, $i);

// ページリンクの設定
$params = array('itemData' => $data);
$pager = &new Pager_Sliding($params);
$data = $pager->getPageData();
$links = $pager->getLinks();

// データの出力
echo "<h3>2 の階乗</h3>";
echo "乗数 ... 値<br>";
foreach ($data as $exp => $value) {
    echo "<br>$exp ... ".number_format($value);
}
echo "<br><br>";

// ページリンクの出力
echo $links['all'];
?>
```



実行例

スクリプト

