Smarty勉強会
at 東京FORT

2005/08/20

By ishii(info@spanstyle.com)

この勉強会について

目的

参加者の皆さんが帰り道で「もうsmartyを使った開発ができるかも？」と思えるようになること

対象者

PHPをある程度使ったことがある、もしくはそれ同等のスキルを有する人で、Smartyテンプレートシステムに興味がある人

テンプレート
システムって？

身近なテンプレートシステム

テンプレート
システムを使うと、
どんなメリットが？

テンプレートシステムを使わない開発

単一のPHPファイル内にロジック(PHP)とデザイン(HTML)が入ったソースになってしまうため、

- **プログラマとデザイナが同時作業しにくい**
- **ロジックとデザインが混在していてわかりづらい**

```
<?php
 $memo = $Container->GetData(DEF_FIELD_REPORT_MEMO);
 $pamphlet_pdf = $Container-
>GetData(DEF_FIELD_REPORT_PAMPHLET_PDF);
 // ID、更新ボタン
 $strId = "未登録";
 $pamphlet = NULL;
 if($nStatus != DEF_RECORD_NEW)
 {
  $strId = "<strong>".$id."</strong>";
$pamphlet = <<< EOF
 <tr>
  <td class="attr"><strong>パンフレット</strong></td>
  <td class="value">
   {$pamplhet_msg}<br />
   <input type="checkbox" name="nopamphlet" value="1"
id="nopamphlet"><label for="nopamphlet">パンフレットを削除する</label>
  </td>
 </tr>

EOF;
  }
  else {
   $pamphlet = <<< EOF
?>
```

テンプレートシステムを使った開発

PHPファイルとテンプレートファイルに分かれ、PHPファイルにはロジック(PHP)を、テンプレートファイルにはデザイン(HTML)を書くようになるため、

- プログラマとデザイナが同時作業しやすい
- ロジック用とデザイン用にファイルが分かれるので、見通しのいいソースになる

```
<?php
// * $_POST['__mode']の値で処理を分岐
// */
if (isset($_POST['__mode'])) {
if ($_POST['__mode'] == 'query') {
/*
* 絞込み中フラグを立てる
*/
$queryParam['query'] = true;
} elseif ($_POST['__mode'] == 'clear') {
}
?>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=euc-jp" />
<title>{$title|default:"ようこそ"}</title>


<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Cache-Control" content="no-cache" />
<meta http-equiv="Expires" content="Thu, 01 Dec 1994 16:00:00 GMT"
/>


<link rel="stylesheet" type="text/css" charset="euc-jp"
href="/manager.css" />
<script type="text/javascript" src="manager.js"></script>
```

テンプレートシステムを利用するメリット

プログラマ側

デザインは気にすることなく、ひたすらロジックの組み立てに集中できてｳﾏｰ

デザイナ側

難解なロジックを目にすることなく、HTML/CSSの組み立てに集中できてｳﾏｰ

結論

みんなウマー

Smartyって何よ？

- **国内では最もメジャーなPHPのテンプレートシステム(テンプレートエンジン)※**
- **必要条件：PHP4.0.6以降**

  (Smarty自体がPHPスクリプトでできているため、他には特に条件はない)

※googleのヒット件数を見る限りでは、日本では一番使われてるっぽい

Smartyのすごいところ

プログラマ側

- キャッシュ機能でサーバ負荷軽減 ※
- 独自プラグイン機能で好き勝手拡張し放題
- 独自フィルタ機能で好き勝手フィルタし放題

デザイナ側

- 修飾子で変数を自由に整形※
- 組み込み関数で表示制御※
- カスタム関数で工数削減※

今回は※の箇所をご紹介します。

Smartyの特徴

- 非常に高速
- 下仕事はPHPパーサが行うので能率的
- コンパイルは一度だけ行われるので、テンプレートのパースによるオーバーヘッドが無い
- 変更されたテンプレートファイルのみ再コンパイルを行うのでスマート
- カスタム関数及び 変数の修正子をカスタム定義する事によって、テンプレート言語を強力に拡張する事が可能
- テンプレート言語の開始と終端を表すデリミタタグの記法を変更可能 (例: { }, {{ }}, <!--{ }-->, 等)
- if/elseif/else/endif ステートメントはPHPパーサに渡されて処理されるので、{if ...}の条件式にはシンプルな式から複雑な式まで自由に指定可能
- section, if等は無制限にネスト可能
- テンプレートエンジンはカスタマイズできるので必要ない(又は推奨されない)かもしれないが、テンプレートファイルにPHPコードを埋め込む事が可能
- キャッシュ機能をサポート
- テンプレートリソースのサポート
- カスタムキャッシュハンドラ関数
- プラグイン構造

出展元：Smarty日本語 マニュアル

コンパイル？

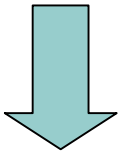
コンパイルって何よ？

テンプレートファイルにおけるSmartyの独自記述部分をパース(解析)し、実行可能なPHPスクリプトへ変換する作業のこと。

コンパイルは通常Smartyによって自動的に行われるので、利用者側では特に意識する必要はない。

テンプレートファイル(index.tpl)：

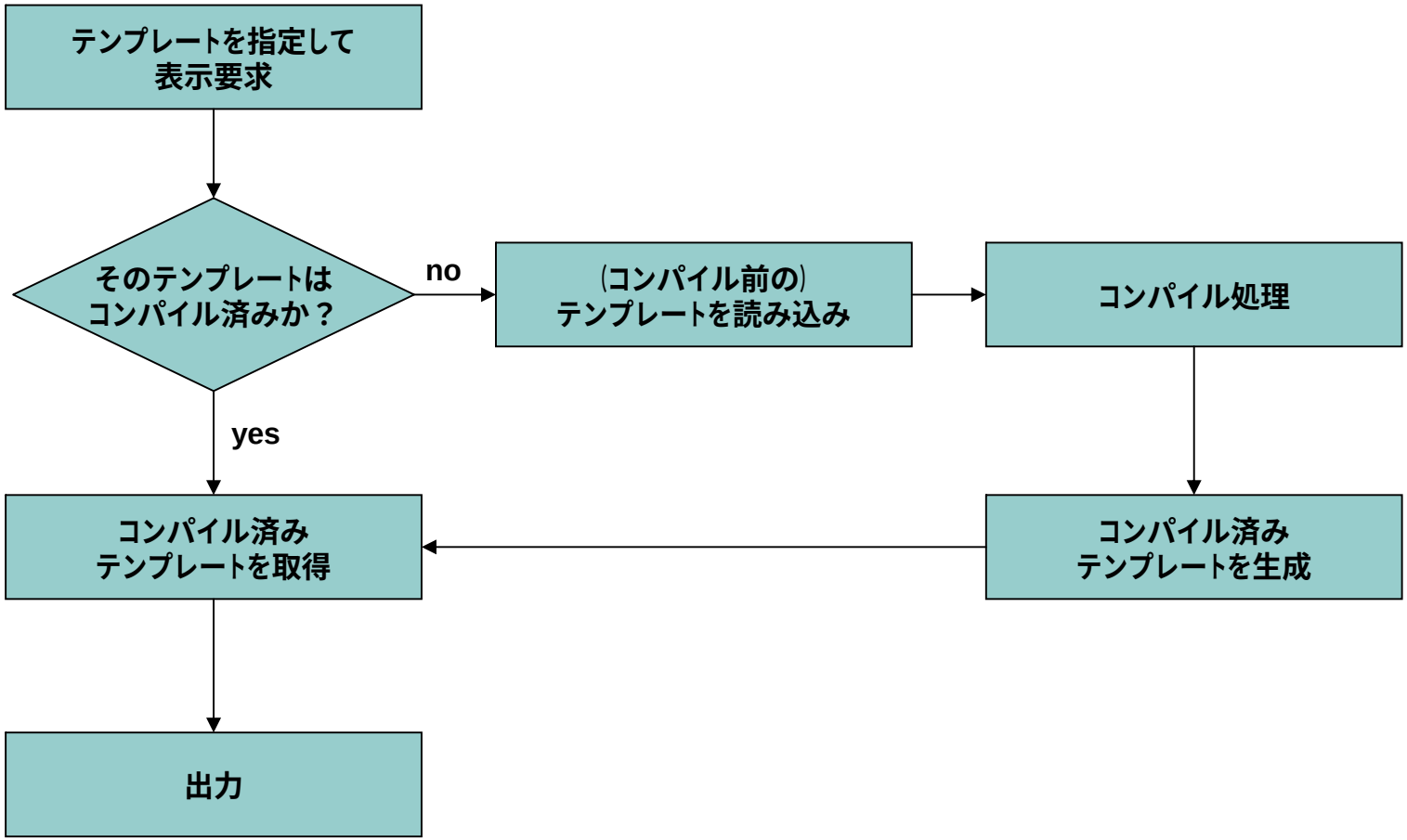
```
<body>
<h1>{$title|default:"タイトルを設定してください"}</h1>
<p>{$contents|default:"コンテンツを設定してください"}</p>
</body>
```



コンパイル済みのテンプレートファイル(%%45^45E^45E480CD%%index.tpl.php)：

```
<body>
<h1><?php echo ((is_array($_tmp=@$this->_tpl_vars['title'])) ? $this->_run_mod_handler('default', true, $_tmp, "タイトルを設定してください") : smarty_modifier_default($_tmp, "タイトルを設定してください")); ?>
</h1>
<p><?php echo ((is_array($_tmp=@$this->_tpl_vars['contents'])) ? $this->_run_mod_handler('default', true, $_tmp, "コンテンツを設定してください") : smarty_modifier_default($_tmp, "コンテンツを設定してください")); ?>
</p>
</body>
```

コンパイルの内部処理はこんな感じ（あくまでもイメージです）



参考元：極める！PHP (p73) / 翔泳社発行

前説
ここまで

まずは
下準備

日本語マニュアルとSmarty本体の入手

日本語マニュアルの入手

http://sunset.freespace.jp/smarty/

有志の方(Shinsuke Matsuda氏)のサイト。掲示板によればhttp://nidieu.flnet.org/smartylabo/が移転先となっているが、長い間アクセス不可能になっている。移転先での最新版マニュアルは2.6.7だったが、旧サイトでは今のところ2.6.6のマニュアルしか入手できない模様。HTML版とchm版(Windowsヘルプ形式)で配布されているけど、個人的には見やすくて検索ができるchm版がお勧め

Smarty本体の入手

http://smarty.php.net/

オフィシャルサイト。2005/08/18時点での最新版は2.6.10となっている。オフィシャルサイトには日本語版マニュアルはないけど、英語版とかなら最新のものがあるのでチェック

マニュアルの読み方

「はじめに」「デザイナーのためのSmarty」「プログラマのためのSmarty」「付録」に大きく分かれているが、開発を進めていく上でよく使うのは「デザイナーのための～」と「プログラマのための～」の2つ。

デザイナーのためのSmarty

テンプレートに割り当てられた変数(テンプレート変数)の整形(修飾)や、テンプレート内で呼び出す関数などに関するマニュアル。

↓

テンプレートファイルの見栄え・整形に関するマニュアル

プログラマのためのSmarty

Smartyクラスのメンバ変数・メンバ関数の解説や、Smartyで使えるプラグイン・フィルタ・キャッシュなど拡張機能に関するマニュアル。

↓

Smartyクラスのインタフェイスや拡張機能などに関するマニュアル

Smartyのインストール

解凍したフォルダ内の./libs/以下を適当なディレクトリに格納したら終了

./libs/以下の内容はこんな感じ。

Smarty.class.php ※コレがSmartyの本体部分
Smarty_Compiler.class.php
Config_File.class.php
debug.tpl
/internals/*.php
/plugins/*.php

セキュリティ上の理由から、/libs/以下をウェブサーバのドキュメントルート以外、かつ全アプリケーションがアクセスするところに配置することが推奨されている。これはバージョンアップにも簡単に対応できることにも考慮してのこと。
もしも各アプリケーションにSmartyを個別にインストールしたければ、各アプリケーションが管理するディレクトリ下、かつドキュメントルート以外の場所にインストールしておけばいいだけの話。

各アプリケーション毎のセットアップ

インストール後、Smartyを利用するためには
各アプリケーション毎のセットアップが必要

- Smarty.class.phpにパスを通す
- アプリケーション用ディレクトリの作成

各アプリケーション毎のセットアップ1
【Smarty.class.phpにパスを通す】

Smarty.class.phpをrequire()して呼び出す必要があるので、このファイルをどうにかして読めるようにする

↓

その方法は？

- include_pathにSmartyインストールディレクトリを追加する
- require()で呼び出す時にSmarty.class.phpを絶対パスで指定する

上記以外にも定数SMARTY_DIRにインストールディレクトリを定義する手法があるが、通常は上記のどちらかをやれば問題ないのであまり気にしないでいいような気がする。詳しく知りたい人はマニュアルの「はじめに」あたりを参照。

各アプリケーション毎のセットアップ2 【アプリケーション用ディレクトリの作成】

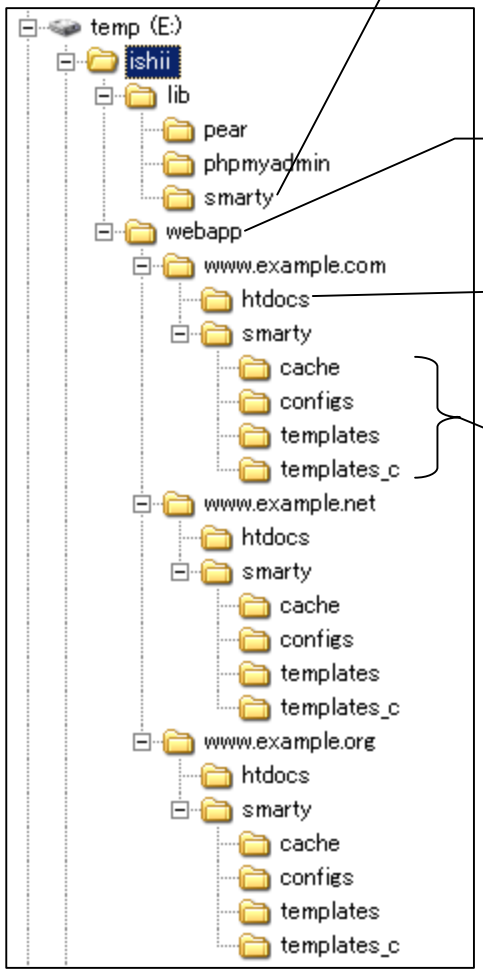
下記4つのディレクトリを、アプリケーションごとに用意する。

- templates　　テンプレートファイル格納ディレクトリ
- templates_c　　コンパイル済みテンプレートファイル格納ディレクトリ
- configs　　設定ファイル格納ディレクトリ
- cache　　キャッシュファイル格納ディレクトリ

「templates_cディレクトリ」および「cacheディレクトリ」の2つは、ウェブサーバの権限で書き込み可能に設定しておく。なお、これらディレクトリもSmarty本体と同じく、ウェブサーバのドキュメントルート外に設置するのを推奨

これにて
セットアップ終了

セットアップ終了後のディレクトリ構成例



Smarty(Smarty.class.phpなど)の
インストール先はここ

ドメイン別の
アプリケーション格納先

www.example.comの
ドキュメントルート

www.example.comで
Smartyが利用するディレクトリ群
(アプリケーション毎に用意する)

Hello, World!

Index.php：

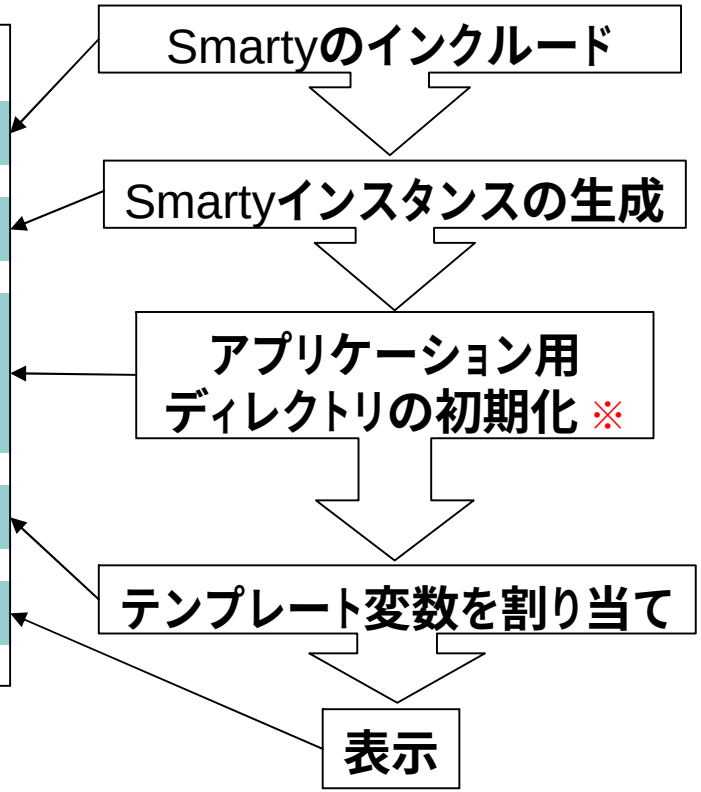
```
<?php

// Smartyライブラリを読み込む
require('Smarty.class.php');

// Smartyオブジェクトのインスタンスを作成
$smarty =& new Smarty;

// アプリケーション用ディレクトリを初期化
$smarty->template_dir = '/web/www.example.com/smarty/templates/';
$smarty->compile_dir = '/web/www.example.com/smarty/templates_c/';
$smarty->config_dir = '/web/www.example.com/smarty/configs/';
$smarty->cache_dir = '/web/www.example.com/smarty/cache/';

// テンプレート変数$messageに値を割り当て
$smarty->assign('message','Hello, World!');

// テンプレートファイル「index.tpl」を使って画面表示
$smarty->display('index.tpl');
?>
```

Index.tpl：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN“
   "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">

<head>
   <meta http-equiv="Content-Type" content="text/html; charset=euc-jp">
   <title>{$message}</title>
</head>

<body>

<h1>{$message}</h1>

</body>

</html>
```

ソースレベルでの処理の流れ (プログラマ側)

```php
<?php

// Smartyライブラリを読み込む
include "Smarty.class.php";

// Smartyオブジェクトのインスタンスを作成
$tmpl =& new smarty;

// アプリケーション用ディレクトリを初期化
$tmpl->template_dir = '/web/www.example.com/smarty/guestbook/templates/';
$tmpl->compile_dir = '/web/www.example.com/smarty/guestbook/templates_c/';
$tmpl->config_dir = '/web/www.example.com/smarty/guestbook/configs/';
$tmpl->cache_dir = '/web/www.example.com/smarty/guestbook/cache/';

// テンプレート変数$messageに値を割り当て
$smarty->assign('message','Hello, World!');

// テンプレートファイル「index.tpl」を使って画面表示
$smarty->display('index.tpl');
?>
```



※smarty継承クラスを作成し、コンストラクタで設定を済ませるのがセオリー

ソースレベルでの処理の流れ (デザイナ側)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN“
   "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">

<head>
   <meta http-equiv="Content-Type" content="text/html; charset=euc-jp">
   <title>{$message}</title>
</head>

<body>

<h1>{$message}</h1>

</body>

</html>
```

普通にHTMLをコーディング

動的変更が必要な部分で、
PHPから渡されたテンプレート変数を呼び出し、
好きなように整形して表示

拡張セットアップ

このまま進めても問題はないけれど、アプリケーション用ディレクトリの初期化を毎回するのは面倒。なのでSmarty継承クラスをつくり、アプリケーション用ディレクトリ初期化などの共通処理をコンストラクタで済ませるようにする。

mysmarty.php：

```
require_once 'Smarty.class.php';

/**
 * Smarty初期化クラス
 */
class MySmarty extends Smarty {

    /**
     * コンストラクタ
     */
    function MySmarty ()
   {
        /**
         * Smartyクラスのコンストラクタを呼び出す
         */
        $this->Smarty();

        $basedir = '/web/www.example.com/smarty/';
        $this->template_dir  = $basedir . 'templates/';
        $this->compile_dir   = $basedir . 'templates_c/';
        $this->config_dir    = $basedir . 'configs/';
        $this->cache_dir     = $basedir . 'cache/';
   }
}
```

最低限
知っておけ！
プログラマ編

プログラマは知っとけ その1
【Smartyクラス変数】

- $template_dir　　// テンプレートファイル格納先
- $compile_dir　　// コンパイル済みファイル格納先
- $config_dir　　// 設定ファイル格納先
- $cache_dir　　// キャッシュファイル格納先
- $left_delimiter　　// デリミタ開始文字列
- $right_delimiter　　// デリミタ終了文字列
- $debugging　　// デバッグフラグ(後述)

プログラマは知っとけ その2
【Smartyクラス関数】

- assign()　　// テンプレート変数割り当て
- config_load()　　// 設定ファイル読み込み
- display()　　// 出力結果表示
- fetch()　　// 出力結果取得

最低限
知っておけ！
デザイナ編

デザイナは知っとけ その1
【デリミタ】

{ と } で囲むと、
Smarty特有の処理開始と終了を表す。

Ex.

{* コメントだったり *}

{$var}

{section name=I loop=$var}
・
・
・
{/section}

デザイナは知っとけ その2
【コメント】

デリミタ({ })と * で囲む。

Ex.

{* ここはコメント *}

{*
こんなのもコメント
*}

{**********
これだってコメントだ
**********}

デザイナは知っとけ その3
【テンプレート変数】

PHPから表示用に渡される変数のこと。bool、int、string、配列、オブジェクトなど、PHPで変数として扱えるものは全てテンプレート変数として渡すことができる。テンプレート変数$varへのアクセス方法はこんな感じ。

{$var}

Ex.

phpファイル側：
$smarty->assign(‘var’, ‘ふー’); // これでテンプレート変数$varに「ふー」を割り当てたことになる

テンプレートファイル側：
{$var} ←これが「ふー」に置き換わって表示される

デザイナは知っとけ その4
【必修のテンプレート変数】

連想配列：

{$var.id} ←「.」の後にキーを書く

配列のインデックス：

{$var[6]} ←PHPと同じ

オブジェクト：

{$var->id} ←PHPと同じ

設定ファイルからの変数：

{#var#} もしくは {$smarty.config.var}

例外な変数：

{$SCRIPT_NAME} は
{$smarty.server.SCRIPT_NAME}と同じ

予約変数$Smarty：

get, post, cookies, server, environment, sessionの
ようなリクエスト変数にアクセスする。その他、
{$smarty.now}で現在時刻へアクセスしたりできる。

$_GET['name']へアクセス
{$smarty.get.name}

$_POST['id']へアクセス
{$smarty.post.id}

$_SESSION['cart']へアクセス
{$smarty.session.cart}

$_ENV['PATH']へアクセス
{$smarty.env.PATH}

$_SERVER['REQUEST_URI']へアクセス
{$smarty.server.REQUEST_URI}

デザイナは知っとけ その5
【修飾子】

テンプレート変数を整形(修飾)するもの。複数の修飾子を「|」で連結することもできる。テンプレート変数$var**に修飾子**modifier**を適用する場合はこんな感じ。**

{$var|modifier}

{$var|modifier:param1:param2･･･}

Ex.
{* **タグを取り除く**strip_tags**修飾子を**$value**に適用** *}
{$value|strip_tags}

{* **修飾子を連結** *}
{$name|default:'**ゲスト**'|escape:'htmlall'}

{* **現在日付を**yyyy/mm/dd**形式で表示** *}
{$smarty.now|date_format:'%y/%m/%d'}

デザイナは知っとけ その6
【必修の修飾子】

- default // デフォルト値を設定
- escape // 各種エスケープ
- date_format // 日付フォーマット整形
- string_format // 文字列フォーマット整形
- number_format // 数値をカンマ区切りに整形

なお、全てのPHP関数は「暗黙で」修飾子として使うことが可能。ただし引数が多い関数の場合は、自作プラグインを作らないとわかりにくくなる傾向がある。

Ex. 3つのパラメータをもつPHP関数fooを「暗黙の」修飾子として使うと・・・
function foo(param1, param2, param3)
↓
{param1|foo:param2:param3・・・}

デザイナは知っとけ その7
【組み込み関数】

テンプレート内で条件分岐(if)を行ったり、繰り返し処理(section)を行ったり。覚えればかなり強力。if文はこんな感じ。

{if $var==true}〜{/if}

Ex.

```
{* $nameが設定されていたらウェルカムメッセージを表示 *}

<div class="message">
{if $name!=''}
<p>ようこそ {$name|escape:'html'} さん</p>
{else}
<p>ログインしてください</p>
{/if}
</div>
```

デザイナは知っとけ その8
【必修の組み込み関数】

- {if}〜{elseif}〜{else}〜{/if}
- {section}〜{sectionelse}〜{/section}
- {include}
- {config_load}
- {literal}〜{/literal}

デザイナは知っとけ その9
【カスタム関数】

テンプレート内で新たにテンプレート変数を作ったり、計算をしたり。組み込み関数よりは細々とした機能のものが多い。カスタム関数customの属性varにparam1を渡すときはこんな感じ。

{custom var=param1}

Ex.

{* テンプレート変数$nameを作成 *}
{assign var="name" value="Bob"}

{* 5000×(100+5)/100を計算し、その結果をテンプレート変数$allに格納 *}
{math equation='price*(100+tax)/100' price=$price tax=$tax assign="all"}

デザイナは知っとけ その１０
【必修のカスタム関数】

- {assign}
- {math}
- {html_checkboxes}**系**

デザイナは知っとけ その１１
【テンプレート内で {と} を文字列として使う】

例えば、Javascript **で**

function foo () { alert(‘foo’); }

とか、CSS**で**

h1 { font-size: 120%; }

とかをテンプレートファイルに直接書きたいときはどうするか？

- **外部ファイル化する**(.js**ファイルや**.css**ファイルへ**)
- **デリミタを**{ } **以外のものに変更する**(PHP**側で対処**)
- {literal}**〜**{/literal}**で囲む**(**テンプレート側で対処**)

デザイナは知っとけ その１２
【その他の決まりごと】

- デリミタ内で文字列【テンプレート変数、真偽値(true/false、yes/no、on/off)、数値以外】を扱いたい場合は“(ダブルクォート)もしくは‘(シングルクォート)で囲む必要がある。
- “(ダブルクォート)内にテンプレート変数を含みたい場合は、そのテンプレート変数が数字、文字、アンダーバー、[]のみで構成されていればそのまま書けるが、これら以外の文字列(.とか->とか)がある場合は`(バッククォート)で囲む必要がある。詳しくはマニュアル「基本構文」を参照。

真骨頂の
キャッシュ機能
(さわりだけ)

キャッシュって何よ

- 前回リクエストされた際のHTMLを一定時間保存して(キャッシュして)おき、次回以降のリクエストではassign()など動的処理の多くをすっ飛ばして高速化を図る
- 当然サーバの負荷が軽くなる
- 毎回同じ商品データをDBから取得して一覧表示しているようなページや、アクセスが多いわりに更新頻度が低いページ(トップページやRSSなど)に使うと効果的
- キャッシュが有効でも、一部を動的に変えるという芸当もできる(「ようこそｘｘｘさん」部分だけは毎回動的に変えたりとか)
- コンパイルとは違うのだよ、コンパイルとは

ＰＨＰ側でソースを
修正するだけでも使えたりして便利

```
<?php

$tmpl =& new MySmarty();
$tmpl->caching=true;
if (!$tmpl->is_cached()) {

    $tmpl->assign('abc', $abc);
    $tmpl->assign('def', $def);
    ・
    ・ // フェッチしたりアサインしたり
    ・
    $tmpl->assign('xyz', $xyz);

}

$tmpl->display('index.tpl', md5($_SERVER['REQUEST_URI']));

?>
```

キャッシュされていれば、
ここの処理は全てスキップできる

詳しくは下記参照

PHP関西セミナーでSmartyキャッシュに関する講演をされた方(ushiro様)が、その際の資料をアップされています。感謝。とてもわかりやすいです。

『Syrup Factory』
http://syrup-factory.com/

『Linuxを使ってみる』
http://www.syrup-factory.com/b/linux/

『2005年06月02日「Smarty Cacheは難しくない」の資料』
http://www.syrup-factory.com/b/archives/2005/06/smarty_cache_1.php

最後に
いろいろと

2バイト文字を考慮して
作られていないことを認識する

truncate**修飾子など、指定したキャラクタ数で**
ああするこうする、といったものは、2バイト文
字の扱いが考慮されていない。

そのため、日本語ベースでこれらの機能を使
いたければ、現状ではプラグインを作るしか
ない。

セキュリティを十分に考慮する

ユーザによって入力された値を表示する際には、必ずescape**修飾子を用いること。**XSS**の脆弱性を引き起こす可能性がある。**

$default_modifiers**に**
escape:'htmlall' (htmlentities相当)
もしくは
escape:'html' (htmlspecialchars相当)
を入れておくのも一つの手。(html**のがいいかも？**)

デバッグ機能を使う

$smarty->debugging=true

と設定したSmartyインスタンスでdisplay()を呼ぶと、全てのテンプレート変数に割り当てられた値をポップアップウィンドウで一覧することができる。ただしfetch()の時は使えないので注意。

なお、ブラウザ側でポップアップを禁止している場合には解除しないと出てこないので、ポップアップしなければ一度ブラウザ設定を見直すこと。

今回触れてないけど結構重要なこと

- 独自の修飾子やカスタム関数の作成方法
- プリフィルタ・ポストフィルタ・アウトプットフィルタ
- キャッシュの詳細な使い方　とか

身につけておくと何かと心強いかと思います。

関連書籍のご紹介

『極める！PHP』￥1,764

坂井 恵 (著), 坂井 恵, 上鍵 忠志, 田中 正裕, 月宮 紀柳, 森川 穣
翔泳社 ; ISBN: 4798108766 ; (2005/06/02)

『Smarty入門~PHP5＋テンプレート・エンジンでつくるMVCアプリケーション~ 』￥2,940

山田 祥寛 (著)
翔泳社 ; ISBN: 4798108839 ; (2005/03/15)

『まるごとPHP! ⟨Vol.1⟩』￥1,995

山田 祥寛 (著), 桝形 誠二 (著), 広川 類 (著), 山本 勇 (著), 岩切 洋一 (著)
インプレス ; ISBN: 4844320254 ; Vol.1 巻 (2004/09)

ご静聴
ありがとう
ございました