How to Correctly Load Templates - Simple PHP Template Engine

September 30, 2009. Tags: PHP, Templating, HTML, Coding

PHP for templating?

PHP is a bit of a rare language as it can already template into text in markup with zero modifications or libraries. It is probably one of the big contributing factors why PHP is one of the most popular languages on the web today. (Can't be the only factor, it didn't work for ColdFusion) Most other web languages have a one or more templating languages with a different syntax that need to be learned on top of the implementing language. PHP lowers the bar to entry by allowing you to put your PHP code right into your html. But as we all know, sometime in your PHP tour, you will realize the need to separate presentation logic and the application logic. Some developers go running to some other solution that provides a different syntax. I am a bit puzzled on why this seems to be common practice, PHP can provide the same features without throwing another template syntax on top of what PHP already does. You can still achieve the separation needed with a simple class (shown at the end of this article).

I mainly see two solutions that people tend to use over PHP. One solution is to use a premade template engine like Smarty on top of PHP. The other is to use string replacement techniques on a template file.

String replacement... sucks

If you wrote your own template engine most likely using a *str_replace* or perhaps *preg_replace* to implement the embedding of dynamic parts of the code. There are two problems with this: One, it's slow; Secondly it's difficult to implement all the features needed to provide a robust templating language. Things like formatting functions, control structures etc are a bit clumsy to add to a solution like this. The other option is to implement very simple variable replacement, and then doing your formatting functions, control structures, etc. in your controller and just assign the result to variable replacement, however, that is completely against the point of having a template engine. The separation of presentation logic and app logic gets pretty blurry when you do some of the presentation logic outside of the template.

Did I mention it's a pretty slow solution?

Smarty and other template engines

Smarty and similar template engines are pretty darn redundant. Here is an example of the workflow for Smarty:

- Smarty language is parsed
- Compiled to PHP
- PHP code is cached
- PHP code is parsed
- PHP code is compiled to opcodes
- If you have a opcode cache, opcodes are cached
- opcodes are ran

If you don't see the redundancy there, I'm not doing my job very well. The whole idea of Smarty is like having a car on top of a car and believing it improves your gas mileage. Most people complain that the Smarty syntax is better than PHP's for templating. Bull. There is nothing really gained in Smarty's syntax, it only looks more concise, but in reality there is not enough gains to support having the bloat on top of PHP. You save a couple of keystrokes, big deal. `{$var}` vs. `<?=$var?>`. That is micro-optimization if I ever saw it. PHP control structres and formatting are much more concise and cleaner looking than Smarty's. Smarty doesn't work with most IDE's, so with PHP you gain everything you get with your IDE (or editor), code completion, highlighting, syntax linting, and more!

common (lame) excuses

My designers don't know PHP

They also don't know the templating language you pick for them. If they are going to learn something, have them just learn enough PHP to do their templating. The syntax of something like smarty isn't really easier to learn at all.

I can't use PHP! that's not separating the presentation logic!

Actually you can achieve clear separation. See the class code below.

PHP syntax sucks for Templating

No, it's just fine. Really.

I don't trust my designers with PHP

You are using a templating engine to solve the wrong problem. Template engines are meant to achieve higher maintainability through separation of logic. What you are trying to solve is a flaw in your job culture. These days, designers don't need to write bad server side code to cause a lot of hurt, they can write some horrid client side code that can be just as bad. Also your templating engine should be flexible in case you run into a wall in implementation. You don't want to paint yourself in a architectural corner because you don't do code review. (if you really don't trust your designers, only let them make static html mockups, have a jr. developer make them into templates.)

The Code:

```php
<?php
class Template
{
    private $vars = array();

    public function __get($name)
    {
        return $this->vars[$name];
    }

    public function __set($name, $value)
    {
        if($name == 'viewTemplateFile')
        {
            throw new Exception("Cannot bind variable named 'viewTemplateFile'");
        }
        $this->vars[$name] = $value;
    }

    public function render($viewTemplateFile)
    {
        extract($this->vars);
        ob_start();
        include($viewTemplateFile);
        return ob_get_clean();
    }
}
```

Usage:

main.php template:

```php
<html>
<head>
    <title><?php echo $title; ?></title>
</head>
<body>
    <h1><?php echo $title; ?></h1>
    <div>
        <?php echo $content; ?>
    </div>
</body>
</html>
```

content.php:

```
<ul>
    <?php foreach($links as $link): ?>
        <li><?php echo $link; ?></li>
    <?php endforeach; ?>
</ul>
<div>
    <?php echo $body; ?>
</div>
```

controller.php:

```
$view = new Template();

$view->title = "hello, world";
$view->links = array("one", "two", "three");
$view->body = "Hi, sup";
$view->content = $view->render('content.php');

echo $view->render('main.php');
```

Other PHP templating solutions you may want to check out

PHP Savant
Zend View, Part of the Zend Framework

Resource:

http://chadminick.com/articles/simple-php-template-engine.html