创建 TPL 自定义模板

学习要点：

1.使用模板引擎
2.TPL 模板流程图
3.创建 TPL 模板

网站：http://www.fishsting.com

Web表现层最简单的形式是普通的HTML文档。PHP可以帮助我们轻松地将动态内容插入HTML文档。不过当程序代码变得复杂时，这种结合就会带来诸多不便。PHP代码和HTML就像一个连体婴儿一般，让程序员越发的厌恶。这个时候，我们就会想在他们之间来上一刀，将PHP代码和静态HTML代码进行分离，使代码的可读性和维护性得到显著提高。而且，这样的好处就是，让美工专心的设计HTML前台页面，程序专心去编写PHP业务逻辑。

一．使用模板引擎

我们所说的模板是Web模板，是主要由HTML标记组成的语言来编写的页面，但也有如何表示包含动态生成内容的方式（解析标签）。模板引擎是一种软件库，允许我们从模板生成HTML代码，并指定要包含的动态内容。

模板引擎的特点：

1.鼓励分离：让更个系统的可读性和维护性得到提高。
2.促进分工：使得程序员和美工去专心处理自己的设计。
3.比PHP更容易解析：编译文件和缓存文件加载更快、占资源更少。
4.增加安全性：可限制模板设计师进行不安全的操作的能力避免误删误访问等。

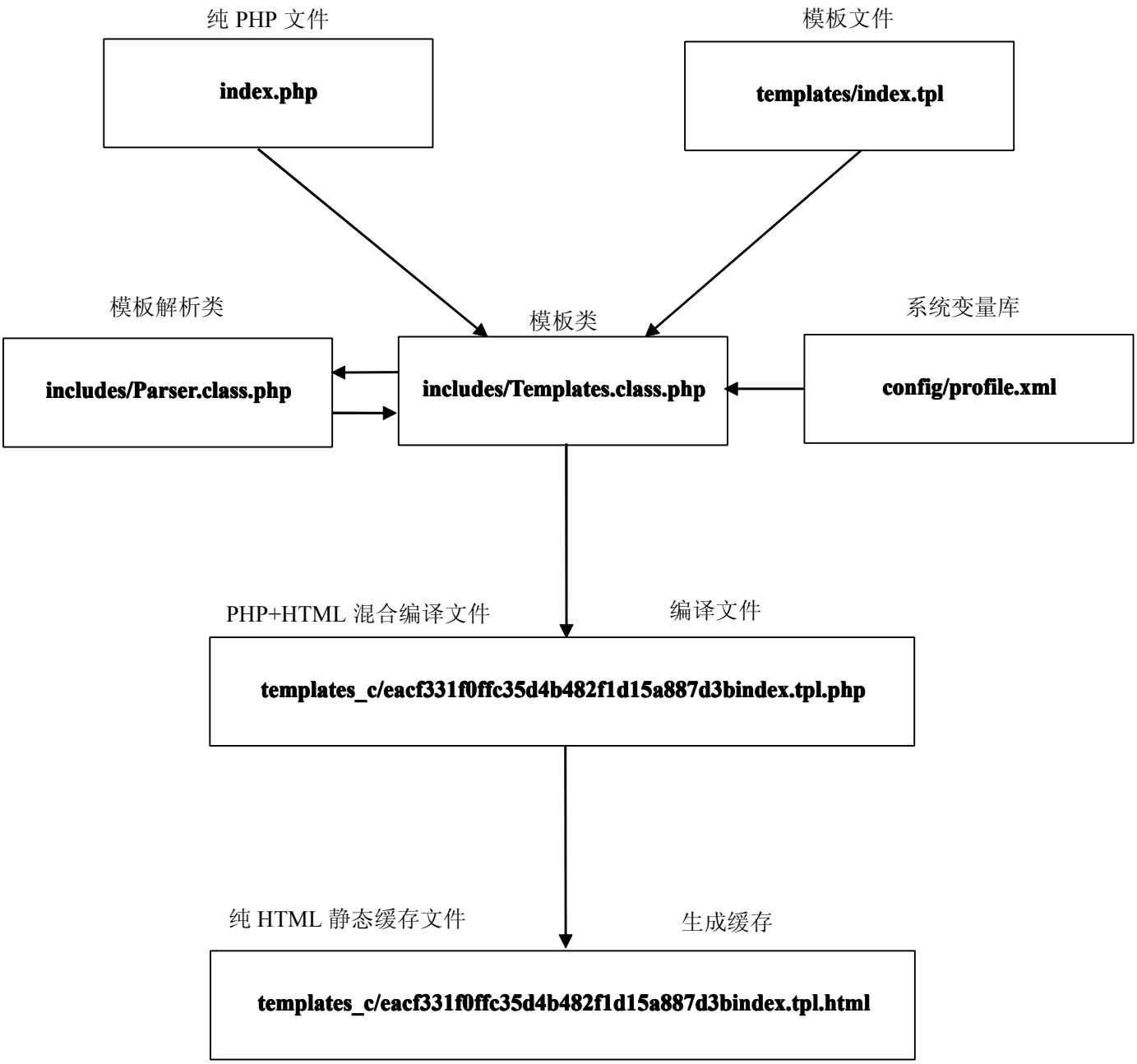
模板引擎的产品：

PHP 有很多团队专门开发的模板引擎，比如 Smarty、Heyes Templates Class、FastTemplate 等等。这些模板引擎我们直接拿过来使用，就可以完全实现以上的诸多特点。可是对于初学者来说，了解模板引擎的原理可以更加深刻的理解为什么要使用模板。所以，本章我们需要自己来创建一个模板引擎。

二．TPL模板流程图

当我们自己创建模板引擎的时候，最大的好处就是从简。因为很多团队编写好的模板引擎，它的功能虽然很多很强大，安全性也很高。但缺点就是很多我们用不到，并且体积非常的臃肿。

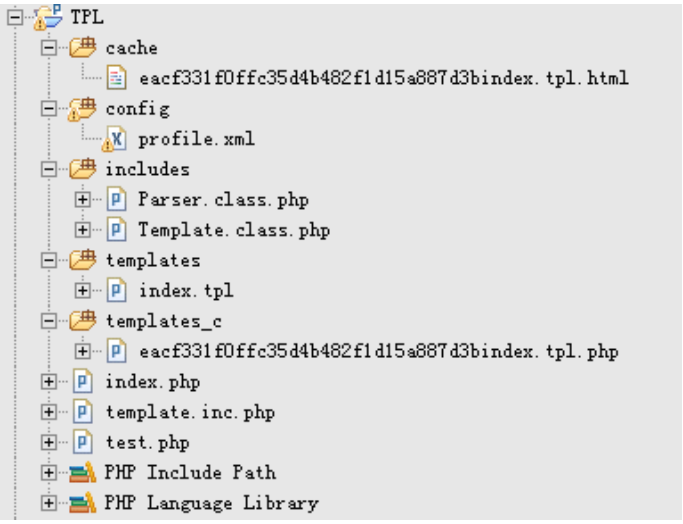
TPL模板引擎设计流程图


纯 PHP 文件
index.php
模板文件
templates/index.tpl
模板解析类
includes/Parser.class.php
模板类
includes/Templates.class.php
系统变量库
config/profile.xml
PHP+HTML 混合编译文件
编译文件
templates_c/eacf331f0ffc35d4b482f1d15a887d3bindex.tpl.php
纯 HTML 静态缓存文件
生成缓存
templates_c/eacf331f0ffc35d4b482f1d15a887d3bindex.tpl.html

三．创建TPL模板引擎

1.创建初始模板所需要的文件夹和文件。

a) index.php主文件，用于编写业务逻辑。
b) template.inc.php模板初始化文件，用于初始模版信息。
c) templates目录存放所有的模板文件。
d) templates_c目录存放所有编译文件。
e) cache目录存放所有缓存文件。
f) includes目录存放所有的类文件。
g) config目录存放模板系统变量配置文件。
h) test.php用于教学中各种生疏知识点测试用的文件。



2.首先，在根目录下创建index.php文件

```
//设置编码为utf-8
header('Content-Type:text/html;charset=utf-8');
//网站根目录
define('ROOT_PATH',dirname(__FILE__));
//存放模板文件夹
define('TPL_DIR',ROOT_PATH.'/templates/');
//编译文件夹
define('TPL_C_DIR',ROOT_PATH.'/templates_c/');
//缓存文件夹
define('CACHE_DIR',ROOT_PATH.'/cache/');
```

3.创建Templates.class.php模板类和Parser.class.php模板解析类

```
//模板类
```

```
class Templates {
    //
}

//模板解析类
class Parser {
    //
}
```

4.在Templates.class.php创建构造方法来验证各个目录是否存在

```
//创建一个构造方法
public function __construct() {
    //验证一下目录是否存在
    if (!is_dir(TPL_DIR) || !is_dir(TPL_C_DIR) || !is_dir(CACHE_DIR)) {
        exit('ERROR：模板文件夹或者编译文件夹或者缓存文件夹没有创建！');
    }
}
```

5.在templates下创建一个index.tpl模板文件

```
.....
<body>

我将被index.php导入

{$name}这个标签必须经过Parser.class.php这个解析类来解析它

</body>

.....
```

6.在Templates.class.php创建display()方法来载入.tpl模板文件，在index.php中创建Templates方法，然后载入。

```
//将模板导入到php文件中
public function display($_file) {
    //设置模板文件的路径
    $_tplFile = TPL_DIR.$_file;
    //判断模板文件是否存在
    if (!file_exists($_tplFile)) {
        exit('ERROR：模板文件不存在！');
```

```
        }
        //设置编译文件的文件名
        $_parFile = TPL_C_DIR.md5($_file).$_file.'.php';
        //判断编译文件是否存在，模板文件是否修改过
        if (!file_exists($_parFile) || filemtime($_parFile) < filemtime($_tplFile)) {
            //生成编译文件
            file_put_contents($_parFile,file_get_contents($_tplFile));
        }
        //载入编译文件
        include $_parFile;
    }

    //引入模板类
    require ROOT_PATH.'/includes/Template.class.php';
    //实例化模板类
    $_tpl = new Template();
    //载入index.tpl
    $_tpl->display('index.tpl');
```

7.将 Templates.class.php 里 display() 方法里生成编译文件的代码转移到Parser.class.php中，目的是为了能够解析后在生成编译文件。

```
    //引入Parser.class.php模板解析类
    require ROOT_PATH.'/includes/Parser.class.php';
    //实例化模板解析类
    $_parser = new Parser($_tplFile);   //模板文件
    $_parser->compile($_parFile);   //编译文件
```

8.创建Parser.class.php里的构造方法，用来获取模板文件，读取模板内容并保存到成员字段里。

```
    //模板的内容
    private $_tpl;
    //构造方法，初始化模板内容
    public function __construct($_tplFile) {
        //读取模板里的内容
        if (!$this->_tpl = file_get_contents($_tplFile)) {
            exit('ERROR：读取模板出错！');
        }
    }
```

9.创建Parser.class.php里的compile()方法，用来解析模板里的各种动态标签内容，并且生成出编译文件。

```
//对外入口，通过Templates.class.php来引入
public function compile($_parFile) {
    //生成编译文件
    if (!file_put_contents($_parFile,$this->_tpl)) {
        exit('ERROR：编译文件生成失败！');
    }
}
```

10.在Templates.class.php创建变量注入方法assign()，并且在index.php往.tpl文件里注入一个变量。

```
//创建一个存放数组的字段
private $_vars = array();
//创建变量注入方法
public function assign($_var, $_value) {
    //$_var表示模板里的变量名，$_value是php文件里声明的变量值
    if (isset($_var) && !empty($_var)) {
        $this->_vars[$_var] = $_value;
    } else {
        exit('ERROR：请设置变量名！');
    }
}

//创建一个普通变量
$_name = '李炎恢';
//注入变量
$_tpl->assign('name',$_name);
```

11.在Parser.class.php里创建parVar()方法，用来解析普通变量。然后在compile()方法内执行。

```
//解析普通变量
private function parVar() {
    //普通变量的匹配模式
    $_patten = '/\{\$([\w]+)\}/';
    //查找是否有普通变量存在
    if (preg_match($_patten,$this->_tpl)) {
        //替换，将{$name}替换成$this->_vars['name']
        $this->_tpl = preg_replace($_patten,
                    "<?php echo \$this->_vars['$1'];?>",$this->_tpl);
    }
}
```

12.在Parser.class.php里创建parIf()方法，用于解析if语句。在index.php声明一个布尔变量，在.tpl文件里创建一个if模板，在compile()执行parIf()方法。

```
//解析if语句
private function parIf() {
    //开头if模式
    $_pattenIf = '/\{if\s+\$([\w]+)\}/';
    //结尾if模式
    $_pattenEndIf = '/\{\/if\}/';
    //else模式
    $_pattenElse = '/\{else\}/';
    //查找是否有if语句
    if (preg_match($_pattenIf,$this->_tpl)) {
        //判断一下是否有结尾的if
        if (preg_match($_pattenEndIf,$this->_tpl)) {
            //替换开头if
            $this->_tpl = preg_replace($_pattenIf,"<?php if
                                        (\$this->_vars['$1']){ ?>",$this->_tpl);
            //替换结尾if
            $this->_tpl = preg_replace($_pattenEndIf,"<?php } ?>",$this->_tpl);
            //判断是否有else
            if (preg_match($_pattenElse,$this->_tpl)) {
                //替换else
                $this->_tpl = preg_replace($_pattenElse,"<?php } else { ?>"
                                                        ,$this->_tpl);
            }
        } else {
            exit('ERROR：IF语句没有关闭！');
        }
    }
}


//布尔变量
$_tpl->assign('a',5<4);

//if模块
{if $a}
    123
{else}
    456
{/if}
```

14.在Parser.class.php里创建parForeach()方法，用于解析foreach语句。在index.php声明一个布尔变量，在.tpl文件里创建一个foreach模板，在compile()执行parForeach()方法。

```
//解析foreach语句
private function parForeach() {
    //搜索foreach语句里的变量
    $_pattenVar = '/\{\@([\w]+)\}/';
    //搜索foreach语句的结尾
    $_pattenEndForeach = '/\{\/foreach\}/';
    //搜索foreach语句的
    $_pattenFirstForeach = '/\{foreach\s+\$([\w]+)\(([\w]+)\,([\w]+)\)\}/';
    //搜索foreach开始语句
    if (preg_match($_pattenFirstForeach,$this->_tpl)) {
        //搜索foreach结束语句
        if (preg_match($_pattenEndForeach,$this->_tpl)) {
            $this->_tpl = preg_replace($_pattenVar,
                                    "<?php echo \$$1; ?>",$this->_tpl);
            $this->_tpl = preg_replace($_pattenEndForeach,
                                                "<?php } ?>",$this->_tpl);
            $this->_tpl = preg_replace($_pattenFirstForeach,
                "<?php foreach(\$this->_vars['$1'] as \$$2=>\$$3) { ?>",
                                                        $this->_tpl);
        } else {
            exit('ERROR：没有发现foreach闭合语句！');
        }
    }
}


{foreach $array(key,value)}
    {@key}....{@value} <br />
{/foreach}

//测试数组
$_array = array(1,2,3,4,5,6,7);
$_tpl->assign('array',$_array);
```

15.在Parser.class.php里创建parInclude()方法，用于解析parInclude语句。在.tpl文件里创建一个parInclude模板，在compile()执行parInclude()方法。

```
//解析include语句
private function parInclude() {
    //判断是否有include包含语句
    if (preg_match('/\{include \"(.*)\"\}/',$this->_tpl,$_file)) {
```

```
            //判断包含文件不得为空，或是否存在
            if (empty($_file[1]) || !file_exists($_file[1])) {
                exit('ERROR：包含文件有误！');
            }
            //include模式
            $_patten = '/\{include \"(.*)\"\}/';
            //替换
            $this->_tpl = preg_replace($_patten,
                                "<?php include '{$_file[1]}';?>",$this->_tpl);
        }
    }

    {include "test.php"}
```

16.在Parser.class.php里创建parCommon()方法，用于解析注释语句。在.tpl文件里创建一个注释模板，在compile()执行parCommon()方法。

```
    //解析注释语句
    private function parCommon() {
        if (preg_match('/\{#}/',$this->_tpl)) {
            $_patten = '/\{#\}(.*)\{#\}/';
            $this->_tpl = preg_replace($_patten,"/* $1 */",$this->_tpl);
        }
    }

    {#}这里是存放的{#}
```

17.创建config目录，创建profile.xml文件，用来存放系统变量。

```
    <?xml version="1.0" encoding="utf-8"?>
    <root>
        <taglib>
            <name>webname</name>
            <value>CMS系统</value>
        </taglib>
        <taglib>
            <name>pagesize</name>
            <value>10</value>
        </taglib>
    </root>
```

18.在Templates.class.php的构造方法里获取xml系统变量。

```
//获取系统变量
$_sxe = simplexml_load_file(ROOT_PATH.'/config/profile.xml');
$_tagLib = $_sxe->xpath('/root/taglib');
//将每个标签和值赋给数组
foreach ($_tagLib as $_tag) {
    $this->_config["$_tag->name"] = $_tag->value;
}
```

19.在Parser.class.php中解析系统变量，在.tpl中创建系统变量的模块。

```
//解析系统变量
private function configVar() {
    //判断是否有系统变量
    if (preg_match('/<!--{([\w]+)}-->/',$this->_tpl)) {
        //系统变量模式
        $_patten = '/<!--{([\w]+)}-->/';
        //替换
        $this->_tpl = preg_replace($_patten,
                "<?php echo \$this->_config['$1']?>",$this->_tpl);
    }
}
```

20.创建缓存目录cache，设定是否开启缓存，在display()方法里创建生成缓存的代码。

```
    //是否开启缓存
    define('IS_CACHE',true);
    //是否开启缓存处理
    IS_CACHE ? ob_start() : null;

    if (IS_CACHE) {
        //写入缓存文件
        file_put_contents($_cacheFile,ob_get_contents());
        //清理缓存
        ob_end_clean();
        //载入静态缓存
        include $_cacheFile;
    }


    //判断是否启用的缓存机制
    if (IS_CACHE) {
        //判断是否存在缓存文件
        if (file_exists($_cacheFile) && file_exists($_par_file)) {
```

```
            //是否修改过
            if (filemtime($_par_file) >= filemtime($_tpl_file) &&
                    filemtime($_cacheFile) >= filemtime($_par_file)) {
                echo '正在浏览的缓存文件';
                include $_cacheFile;
                return;
            }
        }
    }
```

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(yc60.com)联合提供：

本次主讲老师：李炎恢

我的博客：hi.baidu.com/李炎恢/

我的邮件：yc60.com@gmail.com