第 7 章　Smarty 类库

在前面的第一篇和第二篇中，本书介绍了很多在 HTML 页面中内嵌 PHP 代码的例子。随着 Web 技术的不断发展，很多 Web 程序员已经不能满足于这种在 HTML 页面中内嵌 PHP 代码的方法了。这样做的一个很大弊端是为后期维护带来很多麻烦，程序员不得不修改 PHP 代码来迎合页面设计的改动。

本章要介绍的 Smarty 类库有效的解决了这一问题。Smarty 类库通过建立模版库来存放静态 HTML 页面中的 HTML 代码，并使用 PHP 代码对其中的元素进行赋值。这样，当页面的设计发生变化时，只需要修改模版页面就可以完成了。

7.1　Smarty 简介

Smarty 是一款流行的模版引擎类库。所谓模版引擎，就是一个用于执行 PHP 和模版页面的中间件。在开发网站或者其他基于 Web 的应用时，美工设计人员根据实际需要使用模版语言设计出相应的模版页面，而负责编写程序的程序员不需要对模版进行任何修改，只需要在另外的一个 PHP 代码中对模版中的变量进行赋值即可。Smarty 的工作原理如图 7-1 所示。

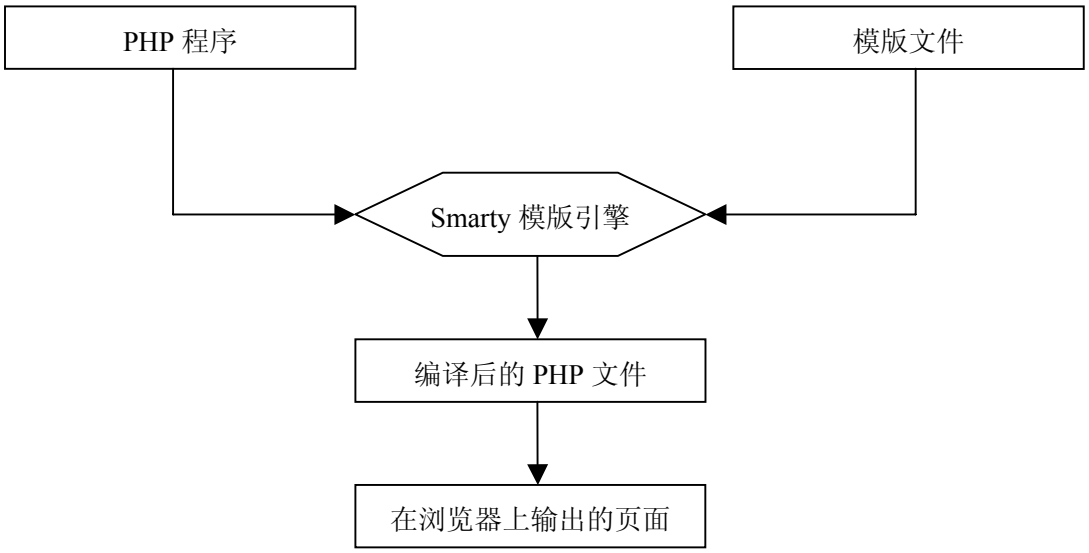

图 7-1　Smarty 工作原理

从图 28-1 可以看出，Smarty 模版引擎一方面读取模版文件中的页面样式，一方面读取 PHP 程序并进行编译，然后结合模版文件和 PHP 程序并进行编译，得到编译后的 PHP 文件，最后在浏览器上将页面输出。

由于这一过程是一个比较复杂的过程，在实际运行时会消耗很多时间，不利于用户在浏览器上的访问。因此，Smarty 类库使用了先编译后执行的机制。也就是说，如果在页面被访问了一次以后，模版文

件和 PHP 代码都没有发生任何变化，则 Smarty 将直接调用编译过的 PHP 文件并将其输出。这一机制节省了编译得时间，保证了用户访问的快捷。

使用模版引擎开发网站主要有以下两点好处。

- 高效：在开发的过程中使美工人员和程序员分工明确，有效的提高效率。
- 易维护：如果在开发结束时，页面的样式或者程序的需求发生了变化，都可以很容易的实现改动。

7.2　Smarty 的安装与配置

下载 Smarty 类库可以通过访问 Smarty 的官方网站 http://smarty.php.net 来获取。目前最新的版本是 2.6.14，因此，本章中所讲的例子与操作方法均以这个版本为例。

7.2.1　Smarty 的安装

文件下载后，可通过如下步骤对 Smarty 类库进行安装。

（1）将 Smarty 压缩包下的全部文件解压缩到一个文件夹中。例如 D:\smarty。

（2）打开 PHP 的安装目录，编辑 php.ini 文件并修改其中的 include_path 参数，在其后增加前面的文件夹名。例如 include_path = ".;d:\smarty"。

（3）重新启动 Apache 服务器使改动生效。

如果希望 Smarty 类库仅对当前站点有效，还可以按照以下的简便方法来进行安装。

（1）将 Smarty 压缩包下 libs 文件夹中的全部文件解压缩到网站所在目录的文件夹中。例如 D:\WWW\htdocs\libs。

（2）修改文件夹名称为自己希望的名字，例如 D:\WWW\htdocs\smarty。

如果在服务器中装有 pear，还可以直接将 libs 文件夹放到 pear 所在文件夹下并将其重命名成一个有意义的名字以方便插件的管理。

7.2.2　Smarty 的配置

Smarty 要求至少存在以下三个文件夹以供 Smarty 运行时使用。

- configs：用于存储配置信息。
- templates：用于存放模版。
- templates_c：用于存放编译后的 PHP 文件。

最简单的方法是将这三个文件夹放置在默认的位置，也就是 PHP 文件所在的目录下。例如，当前要运行的 PHP 文件在 D:\www\htdocs\目录下，那么这三个文件夹分别为 D:\www\htdocs\configs、D:\www\htdocs\templates 和 D:\www\htdocs\templates_c。

Smarty 还允许将这三个文件夹放在其他的目录下，但是需要在使用时声明。具体声明方法将在下一节具体介绍。

7.3　Smarty 程序设计

本节将介绍如何定义一个简单的 Smarty 模版，以及如何通过 PHP 代码将模版中的内容替换成需要的内容。

7.3.1　简单的 Smarty 程序设计

本节将从一个简单的 Smarty 程序入手，介绍一下如何应用 Smarty 来开发 Web 程序。首先创建一个模版文件 test.htm 并保存在前面创建好的 templates 文件夹中，如下所示。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

    <body>
        <H1>Hello, {$Name}</H1>
    </body>

</html>
```

可以看到其中有一个特殊的标记{$Name}，这个标记就是提供给 Smarty 来进行替换的。PHP 程序将查找模版文件中的这种标记并用 PHP 的指定值来替换。默认情况下，这种标记是用一对大括号来声明的，在两个大括号之间为 PHP 的变量名。

在浏览器中浏览这个模版文件，效果如图 7-2 所示。

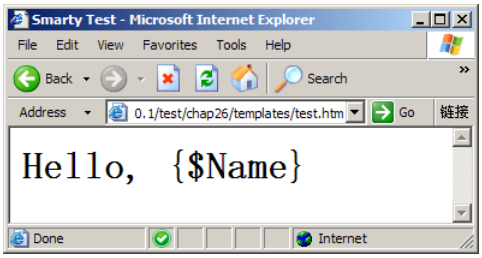

图 7-2　模版文件运行效果

从图 7-2 中可以看到，页面与一个静态的 HTML 完全相同，用来作为标记的{$Name}也被原封不动的输出了。

相应的 PHP 程序如下所示。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$smarty->assign("Name","Simon");                    //对模版中的变量赋值
$smarty->display('test.htm');                       //显示页面
?>
```

上面的程序首先创建了一个新的 Smarty 对象，然后对模版中的变量进行赋值，最后将页面输出。其中对模版页面进行赋值的方法为 assign，其语法格式如下所示。

```
$smarty->assign($variable, $value)
```

其中$smarty 为创建的$Smarty 对象，$variable 为模版文件上的变量的名字，$value 是要被替换成的值。

输出页面的方法为 display，其语法格式如下所示。

```
$smarty->display($filename)
```

其中$smarty 为创建的$Smarty 对象，$filename 为模版文件的文件名。在浏览器中运行该 PHP 代码可以看到如图 7-3 所示的效果。

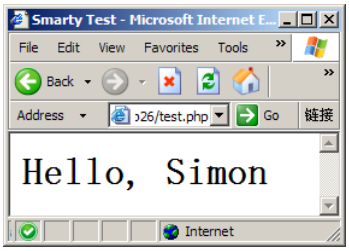

图 7-3　PHP 文件运行效果

图 7-3 所示效果与图 7-2 相同，不同之处只在于模版中的变量被 PHP 中指定的值替换了。这个时候访问 templates_c 文件夹，可以看到一个刚刚生成的文件，代码如下所示。

```
<?php
/* Smarty version 2.6.14, created on 2006-10-22 03:33:06
         compiled from test.htm */
?>
<html>
    <head>
        <title>Smarty Test</title>
    </head>

    <body>
        <H1>Hello, <?php echo $this->_tpl_vars['Name']; ?></H1>
    </body>

</html>
```

这就是编译过的 PHP 文件，实际上在浏览器中看到的效果就是这个文件产生的。

7.3.2　模版对象属性的定义

在 7.2 节中说过，模版所在的文件夹是可以放在其他位置的。也就是说，模版所在文件夹不一定要放在和 PHP 文件所在的同一目录下。但是，必须对模版对象的属性进行重新赋值，常用的模版对象的属性包含以下几种。

- $smarty->template_dir：模版所在文件夹的路径，也就是前面的 templates 文件夹。
- $smarty->compile_dir：编译好的 PHP 文件存放路径，也就是前面的 templates_c 文件夹。
- $smarty->config_dir：配置信息所在文件夹存放路径，也就是前面的 configs 文件夹。
- $smarty->left_delimiter：模版文件的左标记。
- $smarty->right_delimiter：模版文件的右标记。

以下代码对上述属性进行了重定义。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
```

```
define('SITEROOT', 'C:/Program Files/xampp/htdocs/TEST/Chap26');    //指定路径
$smarty = new Smarty;                                               //创建一个新的 Smarty 对象

$smarty->template_dir = SITEROOT . "/templates/";                   //定义 templates 所在路径
$smarty->compile_dir = SITEROOT . "/templates_c/";                  //定义 templates_c 所在路径
$smarty->config_dir = SITEROOT . "/configs/";                       //定义 configs 所在路径
$smarty->left_delimiter = '@';                                      //定义变量的左标记
$smarty->right_delimiter = '_';                                     //定义变量的右标记

$smarty->assign("Name","Simon");                                    //对模版中的变量赋值
$smarty->display('test.htm');                                       //显示页面
?>
```

上面重定义了 Smarty 所需要的文件夹所在路径，并重定义了模版的标记符，相应的模版修改如下所示。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

    <body>
        <H1>Hello, @$Name_</H1>
    </body>

</html>
```

可以看出，用于标记变量的标示符由一对大括号变成了“@”和“_”。运行效果不变。

7.3.3　Smarty 程序编写的一般步骤

通过前面的介绍，可以看出在模版文件编写好以后，Smarty 的 PHP 程序编写主要有以下几步。

（1）通过包含文件载入 Smarty 类库。

（2）定义 Smarty 模版对象。

（3）设定 Smarty 模版对象的参数。

（4）将程序中处理的变量通过 assign 方法替换模版中的变量。

（5）将替换好的页面通过 display 方法输出。

下一节将具体介绍 Smarty 的模版设计，读者会发现，很多逻辑不仅可以在 PHP 代码中实现，也可以将其放到模版文件中。这样做的好处是更大的增加了灵活性。

7.4　Smarty 模版设计

本节将介绍如何设计 Smarty 模版来实现一些较复杂的功能。

7.4.1　模版中的变量

从上一节的简单模版中可以看到模版中的变量与 PHP 中的变量定义方法相同，即使用一个美元符号“$”开头的变量名。但是，与 PHP 不同的是，模版中的变量需要用一对定界符来标示。默认情况下，这一对定界符是一对大括号“{}”。以下代码就是使用了一对大括号来标记模版中的变量$name。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

    <body>
        <H1>Hello, {$name}</H1>
    </body>

</html>
```

7.4.2　变量的修饰

在模版中，Smarty 提供了对变量进行修饰的方法，也就是可以通过格式化方法对 Smarty 中的变量进行格式化输出。按照以往的经验，这种格式化往往是由 PHP 来完成的，如以下代码所示。

PHP 代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                          //包含 Smarty 类库文件
$smarty = new Smarty;                                     //创建一个新的 Smarty 对象
$total = 12345;                                           //对$total 赋值
$smarty->assign("total",$total);                          //对模版中的变量赋值
$formatted_total = number_format($total);                 //格式化$total
$smarty->assign("formatted_total",$formatted_total);      //对模版中的变量赋值
$smarty->display('test1.htm');                            //显示页面
?>
```

模版代码如下所示。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

    <body>
        <H1>Total is {$total}</H1>
        <H1>Formatted Total is {$formatted_total}</H1>
    </body>

</html>
```

可以看到，在模版上面有两个变量，一个是$total，一个是$formatted_total。在 PHP 代码中，使用 number_format 函数对$total 函数进行格式化并将$formatted_total 放置到模版中。

Smarty 提供了在模版中进行格式化的方法，其语法格式如下所示。

```
{$variable|format_function}
```

其中，$variable 为变量名，format_function 为用于格式化的方法。以下代码将上面的模版改写成这

种形式。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

    <body>
        <H1>Total is {$total}</H1>
        <H1>Formatted Total is {$total|number_format}</H1>
    </body>

</html>
```

相应的 PHP 代码改写如下。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$total = 12345;
$smarty->assign("total",$total);                    //对模版中的变量赋值
$smarty->display('test2.htm');                      //显示页面
?>
```

可以看出，使用这种方法的好处是在模版中只需要定义一个变量即可，大大精简了 PHP 代码的复杂性。其运行结果与前面相同，如图 7-4 所示。

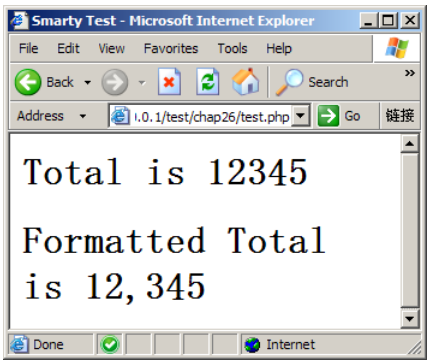

图 7-4　格式化函数应用

7.4.3　区域循环方法

Smarty 模版还提供了循环的功能，用于对数组中的每一个元素实现相同的操作。Smarty 模版主要提供 foreach 和 section 两种方法来实现循环。

foreach 的语法格式如下所示。

```
{foreach key=key1 item=item1 from=$array1}
{$item1}
{/foreach}
```

其中，一对 foreach 标记一个循环区域，key1 表示数组中的每一个键值，item1 表示数组中的每一个元素，$array1 表示传入的数组变量名称。以下代码实现了一个将数组中的所有元素按照一个表格的形式输出的功能。

模版文件如下所示。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

    <body>
        <table border=1>
            {foreach key=key1 item=item1 from=$array1}
            <tr>
                <td>{$key1}</td>
                <td>{$item1}</td>
            </tr>
            {/foreach}
        </table>
    </body>

</html>
```

PHP 代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$array1 = array(1 => "Simon", 2 => "Elaine", 3 => "Susan");   //定义数组
$smarty->assign("array1",$array1);                  //对模版中的变量赋值
$smarty->display('test3.htm');                      //显示页面
?>
```

运行结果如图 7-5 所示。

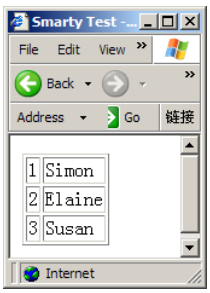

图 7-5　foreach 例子

可以看出，上面的例子循环输出了数组中的全部内容。

模版中的另一种循环的方法是用 section 来实现，其语法格式如下所示。

```
{section name=section1 loop=$array1}
{$array1[section1]}
{/section}
```

其中，一对 section 标记一个循环区域，section1 表示这个循环区域的名字，$array1 表示传入的数组变量名称。以下代码实现了前面的将数组中的所有元素按照一个表格的形式输出的功能。

模版文件如下所示。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>
```

```
    <body>
        <table border=1>
            {section name=section1 loop=$array1}
            <tr>
                <td</td>
                <td>{$array1[section1]}</td>
            </tr>
            {/section}
        </table>
    </body>

</html>
```

PHP 代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                                    //包含 Smarty 类库文件
$smarty = new Smarty;                                               //创建一个新的 Smarty 对象
$array1 = array(0 => "Simon", 1 => "Elaine", 2 => "Susan");         //定义数组
$smarty->assign("array1",$array1);                                  //对模版中的变量赋值
$smarty->display('test.htm');                                       //显示页面
?>
```

其运行结果与前面相同。需要注意的是传递给 section 的数组的键值必须是以 0 开始正整数，否则数组中的内容不会正确的被 section 获得。

Smarty 中循环的一个重要应用是按照数组中的内容输出树状结构，如以下代码所示。

模版文件如下所示。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

  <body>
  <table width="200" border="0">
      {section name=sec1 loop=$array1}
      <tr>
          <td colspan="2">{$array1[sec1].category_name}</td>
      </tr>
      {section name=sec2 loop=$array1[sec1].item}
      <tr>
          <td width="25">·</td>
          <td width="164">{$array1[sec1].item[sec2].item_name}</td>
      </tr>
      {/section}
      {/section}
  </table>
  </body>

</html>
```

PHP 代码如下所示。

```
<?php
```

```
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$array1 = array(                                    //定义数组
    array("category_id" => 1, "category_name" => "水果",
        "item" => array(
            array("item_id" => 1, "item_name" => "苹果"),
            array("item_id" => 2, "item_name" => "梨"),
            array("item_id" => 3, "item_name" => "香蕉")
        )
    ),
    array("category_id" => 2, "category_name" => "蔬菜",
        "item" => array(
            array("item_id" => 4, "item_name" => "白菜"),
            array("item_id" => 5, "item_name" => "油菜")
        )
    ),
    array("category_id" => 3, "category_name" => "主食",
        "item" => array(
            array("item_id" => 6, "item_name" => "米饭"),
            array("item_id" => 7, "item_name" => "馒头")
        )
    )
);
$smarty->assign("array1",$array1);                  //对模版中的变量赋值
$smarty->display('test5.htm');                      //显示页面
?>
```

运行结果如图 7-6 所示。

图 7-6　section 应用实例

7.4.4　模版中的条件判断

在 Smarty 模版中，也可以进行根据变量的值进行条件判断。判断的方法是通过 if 语句来实现的，其语法格式如下所示。

```
{if condition}
```

```
当 if 条件成立时输出的部分
{else}
当 if 条件不成立时输出的部分
{/if}
```

其中，condition 是用于条件判断的逻辑语句。以下代码是一个 if 语句的应用实例，模版文件根据变量的值的不同，输出不同的信息。

模版文件如下所示。

```
<html>
      <head>
            <title>Smarty Test</title>
      </head>

  <body>
  <table width="200" border="0">
       {if $cond == 1}
       <tr>
            <td>条件成立</td>
       </tr>
       {else}
       <tr>
            <td>条件不成立</td>
       </tr>
       {/if}
  </table>
  </body>

</html>
```

PHP 代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$cond = 1;
$smarty->assign("cond",$cond);                      //对模版中的变量赋值
$smarty->display('test6.htm');                      //显示页面
?>
```

运行结果如下所示。

```
条件成立
```

可以看到，模版中的逻辑根据$cond 变量的值进行判断，并只输出了需要的内容。

7.4.5　外部文件的载入

Smarty 模版还可以包含其他文件，这里的其他文件包括其他静态 HTML 文件以及其他模版文件。由于 Smarty 模版可以接收来自 PHP 代码的变量设定，一个模版文件还可以根据需要动态包含不同的文件。Smarty 模版包含文件的语法格式如下所示。

```
{include file=$filename variable=$value …}
```

其中，$filename 是要包含的文件的文件名，variable 和 value 是用于替换被包含文件中关键字的变量设定。以下代码是一个简单的包含文件的例子，其中 test.htm 是模版文件，inc.htm 是被包含的一个静

态 HTML 文件。

模版文件的 HTML 代码如下所示。

```
<html>
      <head>
            <title>Smarty Test</title>
      </head>

  <body>
  {include file="inc.htm"}
  </body>

</html>
```

被包含的文件 inc.htm 代码如下所示。

```
This is a test!
```

PHP 代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$smarty->display('test7.htm');                      //显示页面
?>
```

运行结果如下所示。

```
This is a test!
```

可以看到，被包含的文件内容被显示出来了。被包含的文件名还可以通过 PHP 代码动态指定，下面将 test7.htm 改写如下。

```
<html>
      <head>
            <title>Smarty Test</title>
      </head>

  <body>
  {include file=$inc_name}
  </body>

</html>
```

PHP 代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$inc_name = "inc.htm";
$smarty->assign("inc_name",$inc_name);              //对模版中的变量赋值
$smarty->display('test8.htm');                      //显示页面
?>
```

运行结果与前面相同。

如果被包含的文件是模版，则需要使用调用该模版文件的模版来指定其中的变量。将前面的 inc.htm 改写如下。

```
{$sentence}
```

模版文件 test9.htm 如下所示。

```
<html>
    <head>
        <title>Smarty Test</title>
    </head>

  <body>
  {include file=$inc_name sentence=$sentence}
  </body>

</html>
```

PHP 代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$inc_name = "inc.htm";
$sentence = "This is a test!";
$smarty->assign("inc_name",$inc_name);              //对模版中的变量赋值
$smarty->assign("sentence",$sentence);              //对模版中的变量赋值
$smarty->display('test.htm');                       //显示页面
?>
```

可以看到上面的例子将一个变量$sentence 从 PHP 代码传入模版文件 test.htm 中，又通过 test.htm 文件传入 inc.htm 中，实现了与前面例子相同的功能。

7.5　Smarty 的实际应用——多模板网站

在很多网站中，特别是一些论坛系统，往往支持用户根据自己的喜好选择不同的模板来浏览网站。使用 Smarty 可以很方便的实现这个功能。本节将以一个多模板页面的实现方法为例，简要说明如何使用 Smarty 实现一个多模板网站。

7.5.1　模板的设计

为了实现这个多模板页面，首先需要创建多个模板。这里，创建三个类似的模板，如下所示。

1．模板一

```
<html>
    <head>
        <title>模板 1</title>
    </head>
  <body>
  <a href="?model=1">模板 1</a> |
  <a href="?model=2">模板 2</a> |
  <a href="?model=3">模板 3</a>
  <p align=CENTER><font color=RED>{$title}</font></p>
  <hr>
  {$content}
  </body>
```

```
</html>
```

2．模板二

```
<html>
      <head>
            <title>模板 2</title>
      </head>
  <body>
  <a href="?model=1">模板 1</a> |
  <a href="?model=2">模板 2</a> |
  <a href="?model=3">模板 3</a>
  <p align=CENTER><font color=GREEN>{$title}</font></p>
  <hr>
  {$content}
  </body>
</html>
```

3．模板三

```
<html>
      <head>
            <title>模板 3</title>
      </head>
  <body>
  <a href="?model=1">模板 1</a> |
  <a href="?model=2">模板 2</a> |
  <a href="?model=3">模板 3</a>
  <p align=CENTER><font color=BLUE>{$title}</font></p>
  <hr>
  {$content}
  </body>
</html>
```

这三个模板分别以 model1.htm、model2.htm 和 model3.htm 的文件名存放在 templates 文件夹中，后面要编写的 PHP 代码将根据用户的选择调用不同的模板。需要注意的是三个模板文件中的模板切换链接，这里将使用在地址栏传入参数的形式传入当前的用户对于模板的选择。

7.5.2　页面实现

对于 PHP 页面来说，一方面需要对模板中的变量进行合理的赋值，另一方面也要通过获取地址栏传入的参数来调用不同的模板。具体实现代码如下所示。

```
<?php
require 'libs/Smarty.class.php';                    //包含 Smarty 类库文件
$smarty = new Smarty;                               //创建一个新的 Smarty 对象
$title = "Test";
$content = "This is a test!";
$smarty->assign("title",$title);                    //对模版中的变量赋值
$smarty->assign("content",$content);                //对模版中的变量赋值
if(!isset($_GET['model']))                          //根据参数选择不同的模板
{
      $smarty->display('model1.htm');
```

```
}
else
{
    if(file_exists('templates/'.'model'.$_GET['model'].'.htm'))          //判断模板文件是否存在
    {
        $smarty->display('model'.$_GET['model'].'.htm');
    }
    else
    {
        echo "模板参数不正确！";
    }
}
?>
```

上面代码中对于模板文件是否存在进行了判断，当用户选择的模板不存在时，页面将显示相应的错误提示。

7.6　小结

本章介绍了 Smarty 模版的使用，在实际应用中，Smarty 模版通常应用在一些较大规模的网站或应用系统中。由于 Smarty 能够将 PHP 代码和 HTML 的页面有效的分开，对项目的开发和维护时都有很大好处。除此之外，由于 Smarty 模板更换的灵活性，也为网站的改版和升级提供了很大的便利。特别是对于需要提供多个模板的网站，使用 Smarty 可以使程序设计变得更加简单。