第 8 章　Zend Framework 框架

随着 Web 技术的发展，传统的编程模式已经越来越不能满足日益快速的技术发展。近些年来，一些致力于整合功能的框架技术蓬勃的发展起来。例如，Microsoft 的.Net 技术就是框架开发的一个成功典范。对于 PHP 而言，Zend 公司也为其构建了框架，称为 Zend Framework。目前该产品尚在测试阶段，目前的最新版本为 0.1.5，本章将以该版本为例简要介绍 Zend Framework 的使用方法。

8.1　Zend Framework 的安装

Zend Framework 可以从其官方网站 http://framework.zend.com/下载到，最新版本为 0.1.5，该版本需要 PHP 5.1.4 以上版本的支持。下载后的安装步骤如下所示。

（1）解压缩下载的压缩包到一个专门的文件夹，例如 C:\PHP\ZF\。

（2）修改 PHP 安装目录下的 php.ini 文件，将 Zend Framework 的安装目录添加到 include_path 参数中，如下所示。

```
; Windows: "\path1;\path2"
include_path = ".;c:\php\includes;c:\php\ZF\library"
```

这样，在调用 Zend Framework 时可以直接通过文件名来调用。

（3）修改 Apache 安装目录下 conf 文件夹中的 httpd.conf 文件，修改 mod_rewrite 行使其有效，修改后的代码如下所示。

```
LoadModule rewrite_module modules/mod_rewrite.so
```

这是因为 Zend Framework 需要 Apache 的 mod_rewrite 模块支持。在启动 Apache 的时候需要加载着个模块。

（4）修改 Apache 安装目录下 conf 文件夹中的 httpd.conf 文件，使 AllowOverride 的值为 All，如下所示。

```
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #     Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.    Please see
    # http://httpd.apache.org/docs/2.2/mod/core.html#options
    # for more information.
```

```
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #     Options FileInfo AuthConfig Limit
    #
    AllowOverride All

    #
    # Controls who can get stuff from this server.
    #
    Order allow,deny
    Allow from all

</Directory>
```

这样，允许使用.htaccess 文件来覆盖目录访问设置。

（5）在网站根目录下，创建.htaccess 文件，并填写如下内容。

```
RewriteEngine on
RewriteRule $ index.php
```

这里的作用是声明所有的对网站目录的访问均被 index.php 处理。需要注意的是在 Windows 下由于 Windows 的限制，.htaccess 文件可能无法直接创建。这时，可以直接从命令行通过如下的命令进行创建。

```
C:\Program Files\Apache Software Foundation\Apache2.2\htdocs>copy con .htaccess
RewriteEngine on
RewriteRule $ index.php
^Z
已复制            1 个文件。
```

这里使用 copy con 命令创建.htaccess 文件，在编辑后使用 Ctrl+Z 进行保存。

这时，Zend Framework 的安装就全部完成了，以后就可以直接在网站中调用 Zend Framework 的方法了。

8.2　dispatch 程序的编写

Zend Framework 的一个最大的特点就是完全隐藏了.php 文件的特征，所有对于 PHP 代码的访问均可以通过类似访问文件夹的方式来访问。

在网站根目录，需要创建一个 index.php 文件来进行用户访问的处理。具体代码如下所示。

```
<?php
include 'Zend.php';                                    //包含 Zend.php 文件

Zend::loadClass('Zend_Controller_Front');              //加载 Zend_Controller_Front 类

$controller = Zend_Controller_Front::getInstance();    //创建 Controller 对象
$controller->setControllerDirectory('./controllers');  //设置 controller 文件所在目录
$controller->dispatch();                               //调用 dispatch
?>
```

由于前面已经在.htaccess 文件中定义了文件访问规则，在有用户访问站点时，系统将会自动定向到 index.php 中并运行这段代码。

这段代码定义了 controller 所在文件夹。Zend Framework 的 controller 是用于进行具体操作的类，通过继承 Zend_Controller_Action 来实现。

8.3　controller 程序的编写

上一节介绍了 dispatch 程序的编写，dispatch 程序编写后，当有用户从浏览器访问站点时，Zend Framework 就会自动到 controllers 文件夹寻找相应的 controller 程序。

8.3.1　首页 controller 的编写

Zend Framework 规定，controller 程序的文件名的语法格式如下所示。

```
<controller_name>Controller.php
```

这时，对于文件的访问就会被自动调用到 controller 程序所在文件。例如，首页 index 的 controller 文件名为 indexController.php，该文件代码如下所示。

```
<?php
class IndexController extends Zend_Controller_Action                    //定义 IndexController 类
{
    function indexAction()                                               //定义首页
    {
        echo "Hello World";
    }

    function noRouteAction()                                             //当文件不可访问时输出错误信息
    {
        echo   "Sorry, this page cannot be access.";
    }
}
?>
```

这时，从浏览器上访问网站首页，可以看到如下所示的信息。

```
Hello World
```

可以看到，indexAction 被自动调用了。

8.3.2　其他页面 controller 的编写

除了首页以外，对于任何页面都可以使用这种 controller 形式来实现。下面在 controllers 文件夹创建一个 newsController.php 文件，并输入如下代码。

```
<?php
class NewsController extends Zend_Controller_Action                     //定义 NewsController 类
{
    function indexAction()                                               //页面内容
    {
        echo "Welcome to News!";
```

```
    }
}
?>
```

这时，从浏览器上访问页面 http://localhost/news/，可以看到如下所示的信息。

```
Welcome to News!
```

可以看到，浏览器上的地址被重新定向到了 newsController.php 文件，并调用了相应的方法。

对于传入 PHP 的参数，也可以通过文件夹的形式来传递，以下链接是一个传统的 PHP 站点的文件。

```
http://localhost/news.php?action=page&id=1&type=tech
```

这里将 page、1 和 tech 作为参数传入 news.php 文件进行处理。news.php 文件中的代码可能会根据 id 和 type 的值从数据库中获得相应的记录并根据 action 的值进行相应的处理。

对于 Zend Framework 来说，这种功能可以通过更容易的方法来实现。以下代码修改了上面的 NewsController 类，如下所示。

```
<?php
class NewsController extends Zend_Controller_Action                    //定义 NewsController 类
{
    function indexAction()                                              //首页页面内容
    {
        echo "Welcome to News!";
    }

    function pageAction()                                               //当 Action 为 page 时执行的方法
    {
        $id = $this->_getParam("id");                                   //获得 id 参数
        echo "ID: ".$id."<BR>";
        $type = $this->_getParam("type");                               //获得 type 参数
        echo "TYPE: ".$type."<BR>";
    }

}
?>
```

这时，从浏览器上访问网站首页 http://localhost/news/page/id/1/type/tech/，可以看到如下所示的信息。

```
ID: 1
TYPE: tech
```

可以看到，参数被 NewsController 类正确的获取了。其中使用_getParam()函数来获取参数，语法格式如下所示。

```
_getParam($parm_name)
```

对于浏览器的网址，其格式如下所示。

```
http://host_name/controller_name/action_name/param1/value1/param2/value2/...
```

其中，host_name 为访问网站根目录时用到的服务器域名或者地址。controller_name 为相应的 controller 名称，即 controller 程序文件名 Controller.php 前面的部分。action_name 为相应的方法名，即方法名中 Action 前的部分。param 和 value 是相应的参数名称和参数的值。

因此，前面的域名实际上是调用 newsController.php 文件中的 pageAction 方法。并将 id 和 type 参数传入方法体中。

8.4　视图文件的调用

如前面介绍的那样，Zend Framework 将所有的 PHP 行为均封装成类进行调用。那么对于常规的 PHP 代码和静态 HTML 文件，这种封装无疑带来了麻烦。因此，Zend Framework 提供了一个很好的调用常规文件的方法——使用 Zend_View 对象。

8.4.1　普通文件的直接调用

现在首先来建立本节例子要调用的普通 PHP 文件，该文件位于网站根目录下的 Views 文件夹，文件名为 example.php，代码如下所示。

```
<html>
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <p>Hello World!</p>
    </body>
</html>
```

由于.htaccess 文件的设置，通过在浏览器上直接访问这个文件 http://localhost/views/example.php 将不能获得该文件的内容。相反，输出了 index.php 的错误信息，如下所示。

```
Sorry, this page cannot be access.
```

下面在 controllers 文件夹下创建一个新的文件 exampleControll.php，并输入如下代码。

```
<?php
Zend::loadClass('Zend_View');                                    //加载 Zend_View 类

class ExampleController extends Zend_Controller_Action           //ExampleController 类
{
    function indexAction()                                       //首页调用方法
    {
        $view = new Zend_View();                                 //创建新的 Zend_View 对象
        $view->setScriptPath('./views');                         //设置要调用的文件所在目录
        echo $view->render('example.php');                       //输出要调用的文件
    }

    function noRouteAction()                                     //出错时显示的信息
    {
        echo   "Sorry, this page cannot be access.";
    }
}
?>
```

在浏览器中访问 http://localhost/example/将得到如下所示的结果。

```
Hello World!
```

上面的代码通过 Zend_View 对象的 render 方法，直接将要调用的文件中的内容输出了。

8.4.2　模版文件的调用

Zend Framework 还提供了一个很好的方法可以实现 controller 类与被调用的 PHP 文件中的变量交互。这种方法是通过在 controller 中对 Zend_View 对象的属性进行设置，然后在被调用的文件中使用 escape 方法来获得。

以下代码改写前面的被调用文件 example.php，使其成为了一个带有动态信息的模版文件。

```
<html>
    <head>
        <title><?php echo $this->escape($this->title); ?></title>
    </head>
    <body>
        <?php echo $this->escape($this->body); ?>
    </body>
</html>
```

这里，$this->escape($this->title)表示将 controller 中定义的 Zend_View 对象的 title 属性返回。相应的 controller 程序更改如下。

```
<?php
Zend::loadClass('Zend_View');                                    //加载 Zend_View 类

class ExampleController extends Zend_Controller_Action           //ExampleController 类
{
    function indexAction()                                       //首页调用方法
    {
        $view = new Zend_View();                                 //创建新的 Zend_View 对象
        $view->setScriptPath('./views');                         //设置要调用的文件所在目录
        $view->title = "This is the title";                      //定义 title 属性
        $view->body = "This is the body";                        //定义 body 属性
        echo $view->render('example.php');                       //输出要调用的文件
    }

    function noRouteAction()                                     //出错时显示的信息
    {
        echo    "Sorry, this page cannot be access.";
    }
}
?>
```

在浏览器上访问可以看到，example.php 中的相应字段已经获得了 Zend_View 对象的相应属性。

8.5　用户输入的验证与过滤

Zend Framework 提供了一个专门的类对用户输入信息进行验证和过滤。使用这个类，编写 PHP 程序时将不需要手工对用户的输入进行验证或过滤，只需要进调用相应的方法即可完成。

8.5.1　字符串的验证与过滤

Zend_View 类用于处理字符串的验证与过滤，该类提供了多种方法用于验证一个变量中的数据。以下代码用于验证一个字符串是否只由字母组成。

```
<?php
    Zend::loadClass('Zend_Filter');                    //加载 Zend_Filter 类
    $str = “ABCDE12345FGHIJ”;
    if(Zend_Filter::isAlpha($str))                     //判断$str 是否只有字母
    {
        echo “字符串只由字母组成”;
    }
    else
    {
        echo “字符串不仅只由字母组成”;
    }
?>
```

以下代码是一个对用户提交的表单进行数据验证和过滤的实例。首先在 views 目录下创建一个新的视图文件，如下所示。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>UserInfo</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<H1><p align="center">用户信息表单</p></H1>
<form name="form1" method="post" action="submit.php">
  <table width="500" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr>
      <td>姓名</td>
      <td><input name="name" type="text" id="name" size="50"></td>
    </tr>
    <tr>
      <td>年龄</td>
      <td><input name="age" type="text" id="age" size="20"></td>
    </tr>
    <tr>
      <td>电话号码</td>
      <td><input name="phone" type="text" id="phone" size="20"></td>
    </tr>
    <tr>
      <td>E-mail</td>
      <td><input name="email" type="text" id="email" size="20"></td>
    </tr>
    <tr>
      <td>个人简介</td>
      <td><textarea name="intro" cols="50" rows="10" id="intro"></textarea></td>
    </tr>
```

```
  </table>
  <p align="center">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</p>
</form>
</body>
</html>
```

相应的 PHP 代码如下所示。

```
<?php
Zend::loadClass('Zend_View');                                    //加载 Zend_View 类
Zend::loadClass('Zend_Filter');                                  //加载 Zend_Filter 类

class FilterController extends Zend_Controller_Action
{
    function indexAction()                                       //主页面，调用视图文件
    {
        $view = new Zend_View();                                 //创建 Zend_View 对象
        $view->setScriptPath('./views');                         //设置脚本路径
        echo $view->render('UserInfo.htm');                      //调用 Userinfo.htm 文件
    }

    function submitPHPAction()                                   //用于处理提交表单的方法
    {
        if(!Zend_Filter::isAlpha($_POST["name"]))                //判断姓名必须为英文字母
        {
            $name = $_POST["name"];
            echo "错误：姓名".$name."必须是英文字母<BR>";
        }
        $age = Zend_Filter::getInt($_POST["age"]);               //过滤年龄只保留数字
        echo "提示：过滤后的年龄为".$age."<BR>";
        if(!Zend_Filter::isPhone($_POST["phone"]))               //判断电话号码是否合法
        {
            $phone = $_POST["phone"];
            echo "错误：电话号码".$phone."的格式不正确<BR>";
        }
        if(!Zend_Filter::isEmail($_POST["email"]))               //判断 E-mail 地址是否合法
        {
            $email = $_POST["email"];
            echo "错误：E-mail 地址".$email."的格式不正确<BR>";
        }
        $intro = Zend_Filter::noTags($_POST["intro"]);           //过滤掉个人简介中的 HTML 标签
        echo "提示：过滤后的个人介绍为".$intro."<BR>";
    }
}
?>
```

在用户提交表单后，submitPHPAction 方法将被调用，并对表单中的每一项进行验证或过滤。

8.5.2　数组的验证与过滤

与上一小节类似，Zend Framework 还提供了另外一种用于验证和过滤数组的类 Zend_Filter_Input。该类与 Zend_Filter 的功能类似，不同的是 Zend_Filter_Input 可以直接读取数组，并通过数组中的键对数组中的各个元素进行验证或过滤。以下代码使用 Zend_Filter_Input 重新编写了前面的例子，读者可以看出两个类在使用方法上的区别。

```
<?php
Zend::loadClass('Zend_View');                                    //加载 Zend_View 类
Zend::loadClass('Zend_Filter_Input');                            //加载 Zend_Filter_Input 类

class InputFilterController extends Zend_Controller_Action
{
    function indexAction()                                       //主页面，调用视图文件
    {
        $view = new Zend_View();
        $view->setScriptPath('./views');
        echo $view->render('UserInfo.htm');
    }

    function submitPHPAction()                                   //用于处理提交表单的方法
    {
        $post = new Zend_Filter_Input($_POST);                   //创建新的 Zend_Filter_Input 对象
        if(!$post->testAlpha("name"))                            //判断 name 的值
        {
            $name = $_POST["name"];
            echo "错误：姓名".$name."必须是英文字母<BR>";
        }
        $age = $post->getInt("age");                             //过滤 age 的值
        echo "提示：过滤后的年龄为".$age."<BR>";
        if(!$post->testPhone("phone"))                           //判断 phone 的值
        {
            $phone = $_POST["phone"];
            echo "错误：电话号码".$phone."的格式不正确<BR>";
        }
        if(!$post->testEmail("email"))                           //判断 email 的值
        {
            $email = $_POST["email"];
            echo "错误：E-mail 地址".$email."的格式不正确<BR>";
        }
        $intro = $post->noTags("intro");                         //过滤 intro 中的 HTML 标签
        echo "提示：过滤后的个人介绍为".$intro."<BR>";
    }
}
?>
```

由于 PHP 中用于接收用户输入的方法往往是一个数组，例如$_POST，在实际应用中往往更多的使用 Zend_Filter_Input 类进行用户表单的验证。

8.6　Zend Framework 应用实例——留言本

本章将介绍一个 Zend Framework 开发的小型留言本，该留言本具有发表和浏览留言的功能。该留言本使用文本文件作为数据存储介质。

首先，在网站根目录创建一个名为 DB 的文件夹用于存储文本文件。然后在前面创建的 views 文件夹下创建 post.htm 文件用于存储用户表单，其代码如下所示。

```
<html>
<head>
<title>发表新的留言</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<H1><p align="center">发表新的留言</p></H1>
<form name="form1" method="post" action=submit.php>
  <table width="500" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr>
      <td>标题</td>
      <td><input name="title" type="text" id="title" size="50"></td>
    </tr>
    <tr>
      <td>作者</td>
      <td><input name="author" type="text" id="author" size="20"></td>
    </tr>
    <tr>
      <td>内容</td>
      <td><textarea name="content" cols="50" rows="10" id="content"></textarea></td>
    </tr>
  </table>
  <p align="center">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</p>
</form>
</body>
</html>
```

需要注意的是这里的<form>中的 action 是 submit.php，也就是用于接收表单的文件。但是，这个文件并不存在，这是因为 Zend Framework 将使用专门的方法来接收用户表单。其方法是在 controller 类中定义名为<filename>PHPAction 的方法。因此，这里需要在 controller 类中增加 submitPHPAction 方法。

以下代码为 controller 代码，该文件位于 controllers 文件夹下，名为 guestbookController.php。

```
<?php
Zend::loadClass('Zend_View');                                    //加载 Zend_View 类

class GuestbookController extends Zend_Controller_Action         //定义 controller
{

    var $path = "./DB/";                                         //定义文本文件储存路径
```

```
function indexAction()                                          //首页显示的代码
{
    $dr = opendir($this->path);                                 //打开文件夹
    while($filen = readdir($dr))                                //读取文件夹
    {
      if($filen != "." and $filen != "..")                      //如果文件不是当前目录或父目录
      {
            $fs = fopen($this->path.$filen, "r");               //打开文件并输出文件内容
            echo "<B>标题：</B>".fgets($fs)."<BR>";
            echo "<B>作者：</B>".fgets($fs)."<BR>";
            echo "<B>内容：</B><PRE>".fread($fs, filesize($this->path.$filen))."</PRE>";
            echo "<B>原文链接：</B><A HREF='./guestbook/thread/file/$filen'>点击进入</A>";
            echo "<HR>";
            fclose($fs);                                        //关闭文件
      }
    }
    closedir($dr);
}

function threadAction()                                         //调用各条留言的代码
{
    $filen = $this->_getParam("file");
    $fs = fopen($this->path.$filen, "r");
    echo "<B>标题：</B>".fgets($fs)."<BR>";
    echo "<B>作者：</B>".fgets($fs)."<BR>";
    echo "<B>内容：</B><PRE>".fread($fs, filesize($this->path.$filen))."</PRE>";
    echo "<A HREF='/guestbook'>Back</A>";
    echo "<HR>";
    fclose($fs);
}

function newpostAction()                                        //调用发表留言表单的代码
{
    $view = new Zend_View();
    $view->setScriptPath('./views');
    echo $view->render('Post.htm');
}

function submitPHPAction()                                      //接收表单输入的代码
{
    $filename = "S".date("YmdHis").".dat";
    $fp = fopen($this->path.$filename, "w");
    fwrite($fp, $_POST["title"]."\n");
    fwrite($fp, $_POST["author"]."\n");
    fwrite($fp, $_POST["content"]."\n");
    fclose($fp);
    echo "留言发表成功！";
    echo "<a href='/guestbook'>返回首页</a>";
}
```

```
    function noRouteAction()                                    //出错时的代码
    {
        echo   "Sorry, this page cannot be access.";
    }
}
?>
```

在浏览器上运行 http://localhost/guestbook/即可调用到 indexAction 方法的代码，运行效果如图 8-1 所示。

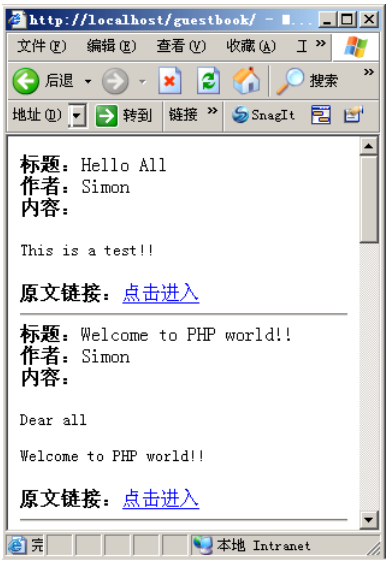

图 8-1　留言本首页

单击“点击进入”链接将运行 threadAction 方法。其链接如下所示。

```
http://localhost/guestbook/thread/file/S20060830125215.dat
```

这个链接是在 indexAction 方法中生成的，而 threadAction 也正是通过这个链接调用相应的文件的。

访问 http://localhost/guestbook/newpost/可以运行 newpostAction 方法，也就是 post.htm 文件，效果如图 8-2 所示。

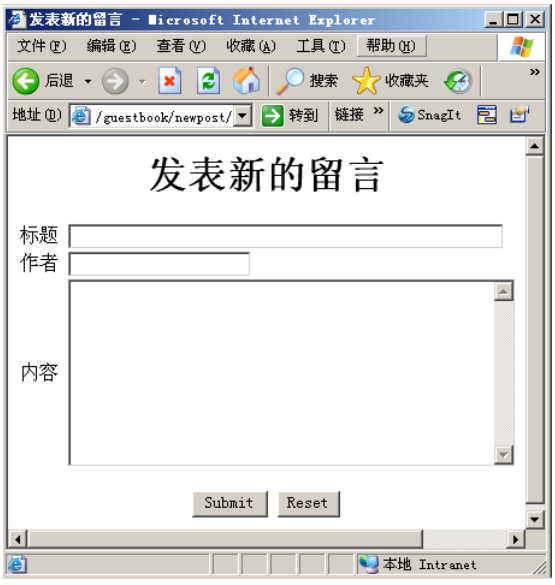

图 8-2　发表新留言页面

单击【Submit】按钮后，submitPHPAction 方法将被调用。这时，用户的留言将被储存在 DB 文件夹下。

可以看到，使用 Zend Framework 有利于功能的封装。通过一个结构清晰的类就很容易的实现了留言本的各项功能。如果需要对留言本的功能进行扩充，也可以很容易得通过对该类代码进行修改来完成。

8.7　小结

本章介绍了 Zend Framework 的一些基本使用方法。Zend Framework 封装了多种操作方法，甚至包含了 Yahoo!网站的搜索功能，这些例程可以在其官方网站的教程中找到。

虽然目前 Zend Framework 还处在预览版阶段，正式的版本一直没有推出。但是相信在不久的将来，或许 Zend Framework 还会有更大的发展的。