第24章　其他有用的特性

PHP中有一些非常实用的功能和特征，它们不属于任何特定的类型。本章将专门介绍这些内容。

在本章中，我们将主要介绍以下内容：

- 使用eval()函数对字符串求值
- 中断执行：die和exit
- 序列化变量和对象
- 获取PHP环境信息
- 暂时改变运行时环境
- 加亮源代码
- 在命令行中使用PHP

24.1　使用eval()函数对字符串求值

函数eval()可以计算出PHP代码字符串的值。例如：

```
eval ( "echo 'Hello World';");
```

将接收字符串的内容并执行它。这行代码将产生与如下语句运行结果相同的输出：

```
echo 'Hello World';
```

eval()函数可以在许多不同的情况下使用。例如，我们可以在数据库里存储一段代码，以便此后可以检索它们并求值。也可能在一个循环里生成代码，然后使用eval()来执行它。

eval()函数的最常见用法是系统模板化。我们可以从数据库中载入HTML、PHP和纯文本的混合。模板系统可以对这些内容格式化并且通过eval()函数来执行任何PHP代码。

可以使用eval()函数有效地更新和校正已有代码。如果我们知道一堆脚本需要在什么地方进行修改，那么可以写一段新的脚本代码，将老的脚本代码加载到一个字符串中（效率较低），运行regexp进行修改，然后用eval()执行修改过的脚本。

我们甚至可以想像，某人对系统非常信任时，可能会希望在浏览器端输入PHP代码，然后在服务器端执行。

24.2　终止执行：die()和exit()

在本书的前面内容中，我们已经介绍了使用语言结构exit来终止一段脚本的执行。回顾一下，该语句的使用只有一行代码，如下所示：

```
exit;
```

它不会返回任何值。也可以使用该函数的别名函数die()。

要实现一个功能性更强的终止操作，可以向exit()函数传递一个参数。这可以在终止代码执行之前输出一个错误信息或者执行特定的函数。这对于Perl程序员来说是非常熟悉的。例如：

```
exit('Script ending now');
```

更常见的是，这两个语句OR（与）一个可能失败的语句一起使用，主要用来实现推出错误。例如，打开一个文件或者连接数据库：

```
mysql_query($query) or die( 'Could not execute query');
```

如果不希望以上代码只是打印出一条简单的错误信息，可以在脚本中断之前调用一个函数：

```
function err_msg()
{
    return  MySQL error was: '.mysql_error();
}

mysql_query($query) or die(err_msg());
```

这样做可以使用户了解程序失败的原因或关闭HTML元素的方法，或在输出缓冲中清除一个已经完成部分的页面，这对用户来说是非常有帮助的。

或者，可以将错误信息以电子邮件的方式发送给自己，这样可以了解程序是否出现重要错误，还可以将错误信息添加到日志文件中。

24.3　序列化变量和对象

序列化是将可以保存在PHP变量或对象中的任何数据转换成字节流的处理过程，这个字节流可以存储在数据库中，或者通过URL在网页之间传递。如果不使用这个处理过程，很难存储和传递整个数组或对象的内容。

随着session（会话）控制的引入，序列化的作用有了一定的降低，数据的序列化现在可以使用会话控制来实现。事实上，为了将会话变量在HTTP请求之间存储，可以使用会话控制函数来序列化会话变量。

但是，我们可能还希望将PHP数组或对象保存到一个文件或数据库中。如果要这样做，就必须了解如何使用这两个函数：serialize()和unserialize()。

可以按如下方式调用serialize()函数：

```
$serial_object = serialize($my_object);
```

如果希望知道序列化都完成了哪些操作，可以通过查看序列化后返回的结果来获知。显而易见，它可以将对象或数组的内容转换为字符串。

例如，我们可以查看对一个简单的employee对象上执行序列化操作的结果。该对象的定义和初始化如下：

```
class employee
{
  var $name;
  var $employee_id;
}
```

```
$this_emp = new employee;
$this_emp->name = 'Fred';
$this_emp->employee_id = 5324;
```

如果对其进行序列化操作并显示到浏览器，其输出结果为：

```
O:8:"employee":2:{s:4:"name";s:4:"Fred";s:11:"employee_id";i:5324;}
```

可以很容易看出原始对象数据和序列化后的数据之间的关系。

由于序列化后的数据只有文本，你可以将它写到数据库或其他位置。请注意，与往常的操作一样，应该使用mysql_real_escape_string()函数对将要写入数据库的数据进行处理，这样可以转义任何特殊字符。注意上述序列化字符串中的引号，我们就知道这是必要的。

要恢复原来的对象，可以调用unserialize()函数，如下所示：

```
$new_object = unserialize($serial_object);
```

显然，如果在将对象保存到数据库之前调用addslashes()函数，那么也需要在反序列化字符串之前调用stripslashes()函数。

当序列化类或使用它们作为会话变量时需要注意一点：在PHP能够重新实例化一个类之前，它必须知道类的结构。因此，必须在调用session_start()或unserialize()函数之前包含该类的定义文件。

24.4 获取PHP环境信息

在PHP中，有许多函数可以用来获取PHP的环境信息。

24.4.1 找到所加载的PHP扩展部件

使用get_loaded_extensions()函数和get_extension_funcs()函数，可以方便地了解哪些函数集合是可用的，以及在每个函数集合中又有哪些函数是可用的。

get_loaded_extensions()函数将返回PHP当前版本中可供使用的函数集合数组。如果给定特定函数集合名称或扩展部件名称，get_extension_funcs()返回该集合所包含的函数数组。

程序清单24-1中的脚本使用了这两个函数来获知当前安装的PHP所包含的函数。

程序清单24-1 list_functions.php——该脚本列出了PHP所有可供使用的扩展部件，每一个扩展所包含的函数

```
<?php
  echo 'Function sets supported in this install are:<br /> ;
  $extensions = get_loaded_extensions();
  foreach ($extensions as $each_ext)
  {
    echo '$each_ext <br />';
    echo '<ul> ;
    $ext_funcs = get_extension_funcs($each_ext);
    foreach($ext_funcs as $func)
```

```
        {
          echo "<li> $func </li>";
        }
        echo "</ul>";
    }
?>
```

请注意，get_loaded_extensions()函数不带任何参数，而get_extension_funcs()函数只带一个参数——扩展部件的名称。

如果要查询是否成功地安装了某个扩展部件，或者正在编写能够在安装过程中生成有用的诊断信息的可移植代码，这些信息是非常有用的。

24.4.2　识别脚本所有者

通过调用get_current_user()函数，可以识别当前处于运行状态的脚本的所有者。如下所示：

```
echo get_current_user();
```

以上语句可在解决权限问题的时候用到。

24.4.3　确定脚本最近修改时间

将最近修改时间写到站点的每一页面上是非常普遍的做法。

可以使用getlastmod()函数（注意函数名中间没有下划线）可以确定脚本的最近修改时间，如下所示：

```
echo date("g:i a, j M Y",getlastmod());
```

该函数将返回UNIX时间戳，它作为date()函数的参数（如上），从而产生可读的日期。

24.5　暂时改变运行时环境

可以在php.ini文件中查看指令集，或者也可以在一个脚本的运行周期中修改它们。这个特性是非常有用的。例如，如果我们知道脚本需要运行一段时间，可以通过设置max_execution_time指令来限制脚本的最长运行时间。

使用ini_get()函数和ini_set()函数，可以访问和修改指令。程序清单24-2显示了使用这些函数的简单脚本。

程序清单24-2　iniset.php——重置php.ini文件中的变量的脚本

```
<?php
  $old_max_execution_time = ini_set( 'max_execution_time', 120);
  echo "old timeout is $old_max_execution_time <br />";

  $max_execution_time = ini_get( 'max_execution_time');
  echo "new timeout is $max_execution_time <br />";
?>
```

ini_set()函数需要两个参数。第一个参数是文件php.ini中的配置指令的名称，它是要修改的对象，第二个参数是修改后的值。该函数将返回配置指令的前一个值。

在这个例子中，我们将脚本最长运行时间的默认值30s（或任何在php.ini文件的设置值）修改为最大值120s。

ini_get()函数只检查特定配置指令的值。指令名称需要通过字符串形式传给该函数。在这个例子中，我们只是用它来检查指令值是否已经改变。

并不是所有的INI选项都是可以通过这种方式进行设置的。每一个选项都有允许设置的级别。可能的级别包括：

- PHP_INI_USER——可以通过ini_set()函数在脚本中改变这些值。
- PHP_INI_PERDIR——如果使用Apache，可以改变php.ini、.htaccess或httpd.conf文件的设置值。能够在.htaccess中修改这些设置值意味着可以在目录级别修改这些值。
- PHP_INI_SYSTEM——可以修改php.ini或httpd.conf文件中的设置值。
- PHP_INI_ALL——可以使用以上任意一种方法修改这些值——也就是，在一个脚本中，在.htaccess文件，或者httpd.conf、php.ini文件中。

ini选项的完整集合和可以设置的级别可以在PHP手册中找到：http: //www.php.net/ini_set。

24.6 源代码加亮

与许多IDE一样，PHP内置一个默认的语法加亮器。这项功能在与别人共享代码或在Web页面上讨论并展示这些代码时非常有用。

函数show_source()和highlight_file()是一样的（确切地说，show_source()函数是highlight_file()的别名函数）。这两个函数的输入参数都是一个字符串，表示文件名称（这个文件必须是PHP文件，否则加亮的结果将会是没有意义的）。例如：

```
show_source( 'list_functions.php');
```

执行以上代码后，该文件的代码将显示在浏览器窗口中，该文件内容将会以不同的颜色显示，而这些颜色是根据文本类型决定的，例如是否是字符串、注释、关键词或HTML。浏览器具有背景颜色。不属于这些类型的文本内容将以默认颜色显示。

highlight_string()函数的工作原理类似，但是它的参数是字符串，并且以语法加亮格式显示在浏览器中。

可以在php.ini文件中设置加亮语法的颜色。该文件中，设置语法加亮颜色的部分如下所示：

```
; Colors for Syntax Highlighting mode
highlight.string  =  #DD0000
highlight.comment  =  #FF9900
highlight.keyword  =  #007700
highlight.bg  =  #FFFFFF
highlight.default  =  #0000BB
highlight.html  =  #000000
```

加亮颜色是用标准的HTML RGB格式来表示的。

24.7　在命令行中使用PHP

通常，我们可以编写或下载许多小程序，并且在命令行下运行它们。在UNIX系统中，这些程序通常都是用shell脚本语言或Perl编写的。在Windows平台上，它们通常都是一个批处理文件。

对于一个Web应用程序，我们可能会首选PHP，但是使其成为一个优秀的Web语言的相同文本处理工具是它的命令行工具程序。

在命令行下，有3种方法来执行PHP脚本：通过一个文件、管道或直接在命令行下。

要在一个文件中执行PHP，请确认PHP可执行文件（php或php.exe，取决于操作系统）位于路径下，并且以脚本名称作为参数调用它。如下所示示例：

```
php myscript.php
```

myscript.php文件只是一个常规的PHP文件，因此它包含了任何常规的PHP语法和PHP标记。

要通过管道传递代码，可以运行任何能够生成有效的PHP脚本的程序，并且以该程序作为输出，导出到php可执行文件。如下示例使用了echo程序给出了一个单行的程序：

```
echo '<?php for($i=1; $i<10; $i++) echo $i; ?>' | php
```

需要再次指出的是，以上的PHP代码封闭在PHP标记中（<?php和?>）。还需要注意的是，这是命令行程序，不是PHP的语言结构。

单行程序的特性使得该程序很容易在命令行下直接传递，如下例所示：

```
php -r 'for($i=1; $i<10; $i++) echo $i;'
```

这里，情况有所不同。在这个字符串中传递的PHP代码并没有封闭在PHP标记中。如果将该字符串封闭在PHP标记中，将遇到一个语法错误。

为命令行使用而编写的PHP程序是不受限制的。我们可以为PHP应用程序编写安装程序。可以在将文本文件导入到数据库之前使用一个脚本来重新格式化这个文本文件。甚至还可以编写一个能够执行任何需要重复的任务，而这些任务需要在命令行下执行；一个很好的例子就是，可以编写一个PHP脚本，在开发用途的Web服务器中将所有PHP文件、图像和MySQL表格结构复制到产品服务器中。

24.8　下一章

本书的第五篇“创建实用的PHP和MySQL项目”，将介绍一些使用PHP和MySQL构建相对复杂的实用项目。这些项目为我们可能面对的任务提供了很好的例子，此外，它们还说明了如何在大型项目中应用PHP和MySQL。

第25章“在大型项目中使用PHP和MySQL”讨论了大型项目中使用PHP编程可能遇到的问题。这些问题包括软件工程的一些基本原则，例如软件设计、文档管理和变更管理。