# 第23章　在PHP中使用会话控制

在本章中，我们将讨论PHP的会话控制功能。

在本章中，我们将主要介绍以下内容：

- 什么是会话控制
- cookie
- 创建一个会话的步骤
- 会话变量
- 会话和身份验证

## 23.1　什么是会话控制

我们可能曾经听说过“HTTP是无状态的协议”。这是说，HTTP协议没有一个内建机制来维护两个事务之间的状态。当一个用户在请求一个页面后再请求另外一个页面时，HTTP将无法告诉我们这两个请求是来自同一个用户。

会话控制的思想是指能够在网站中根据一个会话跟踪用户。如果我们可以做到这点，就可以很容易地做到对用户登录的支持，并根据其授权级别和个人喜好显示相应的内容。我们可以根据会话控制记录该用户的行为，还可以实现购物车。

在PHP 4及其以后版本中，PHP自身包含了会话控制函数。自从超级全局变量概念的引入，会话控制方法就发生了一些变化。如今可以使用$_SESSION超级全局变量。

## 23.2　理解基本的会话功能

PHP的会话是通过唯一的会话ID来驱动的，会话 ID是一个加密的随机数字。它由PHP生成，在会话的生命周期中都会保存在客户端。它可以保存在用户机器里的cookie中，或者通过URL在网络上传递。

会话ID就像一把钥匙，它允许我们注册一些特定的变量，也称为*会话变量*。这些变量的内容保存在服务器端。会话ID是客户端唯一可见的信息。如果在一次特定的网站连接中，客户端可以通过cookie或URL看到会话ID，那么我们就可以访问该会话保存在服务器上的会话变量。在默认情况下，会话变量保存在服务器上的普通文件中。（如果愿意编写自己的函数，我们可以改变这种情况，使用数据库来保存这些变量——在“配置会话控制”一节将详细介绍该内容。）

我们可能曾经使用过一些网站，它们将会话ID保存到URL中，如果在URL中有一串看起来像随机数字的字符串，可能它就是某种形式的会话控制。

cookie是与会话不同的解决方法，它也解决了在多个事务之间保持状态的问题，同时，它还可以保持一个整洁的URL。

### 23.2.1　什么是cookie

*cookie*其实就是一小段信息，它可以由脚本在客户端机器保存。可以通过发送一个包含特定数据并且具有如下格式的HTTP标题头，从而在用户端机器设置一个cookie：

```
Set-Cookie: NAME -VALUE; [expires=DATE;] [path=PATH;]
 [domain=DOMAIN_NAME;] [secure]
```

这会创建一个名为NAME，值为VALUE的cookie。除了该参数，其他参数都是可选的。expires域设置该cookie失效日期（请注意，如果失效日期不设置，cookie将永远有效，如果不手动将其删除的话）。path和domain域合起来指定URL或与cookie相关的URL。secure关键字的意思是在普通的HTTP连接中不发送cookie。

当浏览器连接一个URL时，首先要搜索当地保存的cookie。如果有任何与正在连接的URL相关的cookie，浏览器将它提交到服务器。

### 23.2.2　通过PHP设置cookie

可以使用setcookie()函数在PHP中手动设置cookie。该函数原型如下所示：

```
bool setcookie (string name [, string value [, int expire [, string path
 [, string domain [, int secure]]]])
```

参数和前述的Set-Cookie标题对应。

如果按如下方式设置一个cookie：

```
setcookie ( 'mycookie', 'value');
```

当用户访问站点中的下一页（或者重载当前页）的时候，可以通过名为$_COOKIE['mycookie']来实现。

可以调用setcookie()来删除一个cookie，setcookie()中的参数为要删除的cookie名称和已经过去的到期时间。也可以使用前面已经给出的cookie语法并通过header()函数来手动设置cookie。需要注意的一点是，cookie标题头必须在发送其他标题头之前发送，否则就无效（这是cookie的限制，而不是PHP的限制）。

### 23.2.3　在会话中使用cookie

使用cookie也存在一些问题：一些浏览器不接受cookie，一些用户可能将其浏览器的cookie功能关闭了。这也就是PHP会话控制使用cookie/URL双模式的原因之一（我们将在稍后的内容中详细介绍它）。

当使用PHP会话的时候，不必手动设置cookie。会话函数可以完成该操作。

可以使用函数session_get_cookie_params()来查看由会话控制设置的cookie内容。它将返回包含元素lifetime、path、domain和secure的相关数组。

也可以使用：

```
session_set_cookie_params($lifetime, $path, $domain [, $secure]);
```

来设置会话cookie的参数。

如果希望了解更多关于cookie的内容，可以参考Netscape网站中关于cookie的规范说明：http://wp.netscape.com/newsref/std/cookie_spec.html。

（我们很可能会忽略这个事实，那就是Netscape将其称为“基础规范”——该规范在1995年就已经出现了。它其实是类似于标准的文档，虽然还不是标准，但是大家都将其看作标准。）

### 23.2.4 存储会话ID

在默认情况下，PHP将在会话中使用cookie。如果可能，可以设置一个cookie用来存会话ID。

另一个使用会话的方法是将会话ID添加到URL中。如果在php.ini文件设置了session.use_trans_sid指令，则可以自动实现它。当然，启用该设置后，在使用时应该非常小心。因为它增加了站点的安全风险。如果将该指令设置为ON，用户就可以将包含该会话ID的URL通过电子邮件发送给其他人，而该URL就可以保存在一个公众可以访问的计算机中，或者在一个公众可以访问计算机的浏览器历史记录或书签。或者，可以手动将会话ID嵌入到链接中，这样它就可以一起传递到用户端。会话ID保存在常量SID中。要手动将其传递过去，需将它添加到一个与GET参数类似的链接末尾：

```
<A HREF='link.php?<?php echo strip_tags(SID); ?>'>
```

（在这里，使用strip_tags()函数可以避免一些跨站点的攻击。）

通常，如果可能的话，选中--enable-trans-sid选项对PHP进行编译会更容易一些。

## 23.3 实现简单的会话

使用会话的基本步骤如下：

1. 开始一个会话
2. 注册会话变量
3. 使用会话变量
4. 注销变量并销毁会话

请注意，这些步骤不一定都要发生在同一个脚本中，其中的一些步骤可以在多个脚本中发生。接下来，我们将依次讨论这些步骤。

### 23.3.1 开始一个会话

在使用会话功能前，必须开始一个会话。可以通过如下介绍的两种方法来开始一个会话。

第一种方法，也是最简单的方法，就是以调用session_start()函数开始一段脚本：

```
session_start();
```

该函数将检查是否有一个会话ID存在。如果不存在，就创建一个，并且使其能够通过超级全局数组$_SESSION进行访问。如果已经存在，将这个已经注册的会话变量载入以便使用。

你必须在使用会话的脚本开始部分调用session_start()函数。如果没有调用这个函数，所有保存在该会话的信息都无法在脚本中使用。

第二种方法是将PHP设置成当有用户访问网站的时候就自动启动一个会话。可以使用

php.ini文件中的session.auto_start选项完成该设置——我们在讨论配置的时候再详细介绍它。这种方法有一个很大的缺点：启用auto_start设置导致无法使用对象作为会话变量。这是因为该对象的类定义必须在创建该对象的会话开始之前载入。

### 23.3.2　注册一个会话变量

最近，在PHP中，会话变量的注册已经有所改变。自从PHP 4.1版本以后，会话变量保存在超级全局数组$_SESSION中。要创建一个会话变量，只需在这些数组中设置一个元素，如下所示：

```
$_SESSION[ 'myvar'] = 5;
```

以上代码创建会话变量只有在会话结束或手动重置它时才会失效。根据php.ini文件对会话gc_maxlifetime指令设置，该会话也可能会过期。该指令将确定会话的持续时间（秒为单位），超过时间，该会话将被垃圾回收。

### 23.3.3　使用会话变量

要使一个会话变量在某个范围内可以使用，必须首先使用session_start()函数启动一个会话。这样，就可以通过$_SESSION超级全局数组访问这个变量了。例如，$_SESSION['myvar']。

当使用对象作为会话变量时，在调用session_start()函数重新载入会话变量之前，必须包含该类对象的定义。这样，PHP就知道如何构建这个会话对象。

相反，在检查会话变量是否已经被设置时（例如，通过isset()函数或empty()函数），必须非常小心。请记住，变量可以被用户通过GET或POST设置。可以通过检查$_SESSION数组来确定一个变量是否是注册的会话变量。可以通过如下所示的代码来检查：

```
if (isset($_SESSION[ 'myvar'])) ...
```

### 23.3.4　注销变量与销毁会话

当使用完一个会话变量后，可以将其注销。通过注销$_SESSION数组的适当元素，可以直接注销该变量，如下所示：

```
unset($_SESSION[ 'myvar ]);
```

请注意，session_unregister()函数和session_unset()函数不再是必须的和推荐的方法。这些函数是在引入$_SESSION数组之前才使用的。

我们不能销毁整个$_SESSION数组，因为这样将禁用会话功能。要一次销毁所有的会话变量，可以使用如下所示语句：

```
$_SESSION = array();
```

当使用完一个会话后，首先应该注销所有的变量，然后再调用：

```
session_destroy();
```

来清除会话ID。

## 23.4　创建一个简单的会话例子

上面的内容看起来可能还有些抽象，下面让我们来看一个例子。我们将实现3个页面。

在第1页里，我们启动了一个会话并注册了$_SESSION['sess_var']变量。代码如程序清单23-1所示。

**程序清单23-1　page1.php——启动一个会话并注册一个变量**

```
<?php
  session_start();

  $_SESSION[ 'sess_var'] = 'Hello world!';

  echo 'The content of $_SESSION[\'sess_var\'] is '
       .$_SESSION[ 'sess_var'].'<br />';
?>
<a href='page2.php">Next page</a>
```

我们已经注册了该变量并设置了它的值。该脚本输出如图23-1所示。


图23-1　page1.php显示出的会话变量初始值

在该页面上，该变量的最终值可以被后续页面使用。在该脚本末尾，会话变量被序列化了，或者说被冻结了，直到再次调用session_start()函数后，该变量才会被载入。

因此，在接下来的脚本中，我们将调用session_start()函数。该脚本如程序清单23-2所示。

**程序清单23-2　page2.php——访问一个会话变量并注销它**

```
<?php
  session_start();

  echo 'The content of $_SESSION[\'sess_var\'] is
       .$_SESSION[ 'sess_var'].'<br />';

  unset($_SESSION[ 'sess_var']);
?>
<a href="page3.php">Next page</a>
```

在调用session_start()之后，可以获取$_SESSION['sess_var']变量和前面所存储的值，如图23-2所示。

在使用了该变量之后，必须注销该变量。这样，会话仍然存在，但是变量$_SESSION['sess_var']不再是一个注册变量。

最后，我们将进入到page3.php页面，它也是本例最后一页。该脚本的代码如程序清单23-3所示。

**程序清单23-3　page3.php——结束会话**

```
<?php

  session_start();

  echo 'The content of $_SESSION[\'sess_var\'] is '
       .$_SESSION['sess_var'].'<br />';

  session_destroy();
?>
```

可以从图23-3看出，我们将无法再访问$_SESSION['sess_var']变量值。

在PHP 4.3以前的版本中，当尝试注销$HTTP_SESSION_VARS数组或$_SESSION数组中的变量时，可能会遇到一个问题。如果发现无法注销元素（也就是说，它们还是处于注册状态），可以使用session_unregister()函数来清除这些变量。

通过调用session_destroy()函数清除会话ID。

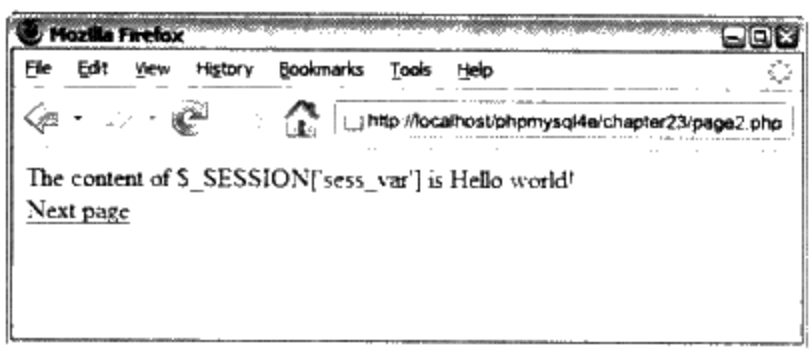


图23-2　会话变量的值已经通过会话ID传递到page2.php

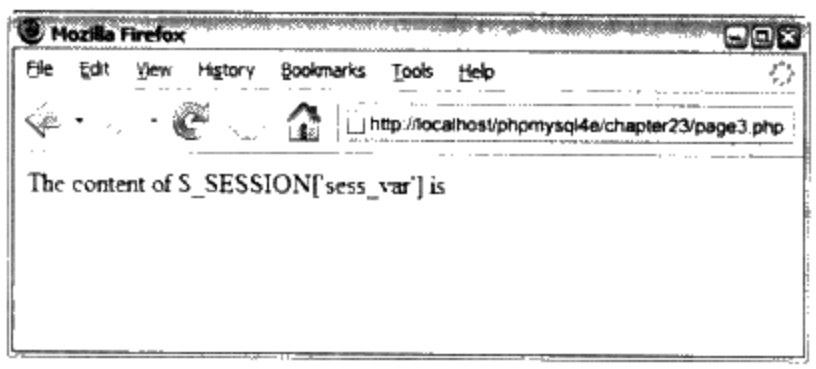


图23-3　注销的变量不再有效

## 23.5　配置会话控制

php.ini配置文件中有一组会话配置选项，可以对其进行设置。一些有意义的选项及其描述如表23-1所示。

**表23-1　会话配置选项**

| 选项名 | 默认 | 效果 |
|---|---|---|
| session.auto_start | 0（被禁用） | 自动启动会话 |
| session.cache_expire | 180 | 为缓存中的会话页设置当前时间，精确到分钟 |
| session.cookie_domain | none | 指定会话cookie中的域 |
| session.cookie_lifetime | 0 | cookie会话ID将在用户的机器上延续多久。默认值0表示延续到浏览器关闭 |
| session.cookie_path | / | 在会话cookie中要设置的路径 |

（续）

| 选项名 | 默认 | 效果 |
| --- | --- | --- |
| session.name | PHPSESSID | 会话的名称，在用户系统中用作会话名 |
| session.save_handler | files | 定义会话数据保存的地方。可以将其设置为指向一个数据库，但是需要编写自己的函数 |
| session.save_path | "" | 会话数据存储的路径。更通常的情况下，传递给存储的参量由session.save_handler函数来处理和定义 |
| session.use_cookies | 1（允许使用） | 配置在客户端使用cookie的会话 |
| session.cookie_secure | 0（被禁用） | 确定是否应该在安全连接中发送Cookie |
| session.hash_function | 0（MD5） | 允许指定用来生成会话ID的哈希算法。"0"表示使用MD5（128位）；"1"表示SHA-1（160位）。这种设置是在PHP 5.0版本中引入的。 |

## 23.6 通过会话控制实现身份验证

最后，我们将介绍使用会话控制的实际应用例子。

会话控制最常见的用法可能就是在用户通过一个登录机制验证后跟踪该用户的行为。在这个例子中，我们将结合MySQL身份验证功能和会话控制功能的使用来实现该功能。该功能将作为我们在第27章"建立用户身份验证机制和个性化设置"中实现的项目的基础，将在那个项目中再次使用。我们将再次使用在第17章"使用PHP和MySQL实现身份验证"中创建的身份验证数据库。可以查看程序清单17-3了解该数据库的详细信息。

本例包含3个简单的脚本。第一个是auth-main.php，它提供了一个登录表单，并且为站点成员提供了身份验证。第二个是members_only.php，只向成功登录的成员显示信息。第三个是logout.php，用户退出登录。

要了解这些是如何工作的，请查阅图23-4，它是由authmain.php显示的初始页面。


图23-4 因为用户尚未登录，所以显示的是一个登录页面

该页给出了用户登录的地方。如果用户没有登录而访问成员区，将得到如图23-5所示的信息。

然而，如果用户先登录了（用户名：testuser；密码password。在第16章中，我们已经创建了这个用户）然后尝试浏览成员页面，看到的页面如图23-6所示。

下面，我们看一下该应用程序的代码。该程序的大部分代码在authmain.php中，该脚本如程序清单23-4所示。我们将详细介绍它。

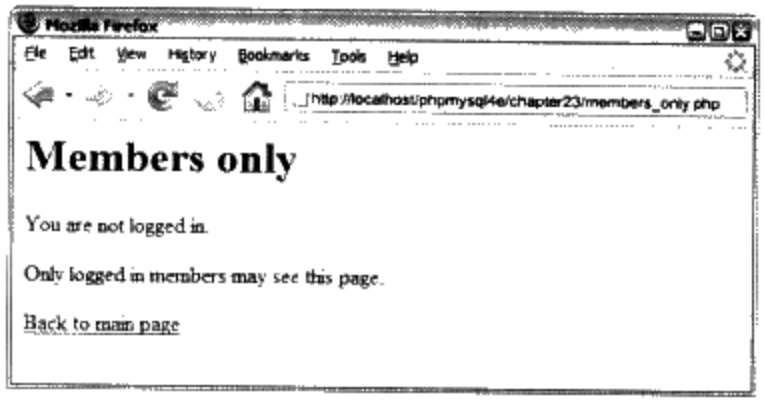


图23-5　用户没有登录就不能看到网站内容；网站将显示警告信息

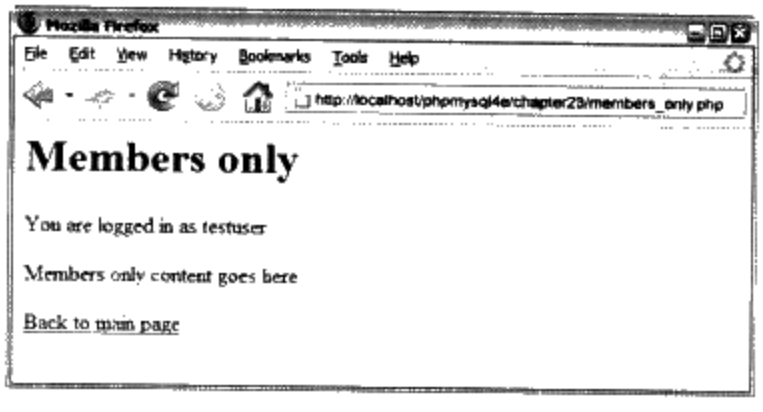


图23-6　用户已经登录，可以访问成员区的内容

**程序清单23-4　authmain.php——身份验证应用程序的主体部分**

```
<?php
session_start();

if (isset($_POST['userid']) && isset($_POST['password']))
{
  // if the user has just tried to log in
  $userid = $_POST['userid'];
  $password = $_POST['password'];

  $db_conn = new mysqli('localhost', 'webauth', 'webauth', 'auth');
  if (mysqli_connect_errno()) {
   echo 'Connection to database failed:'.mysqli_connect_error();
   exit();
  }

  $query = 'select * from authorized_users '
           .'where name='$userid'"
           .'and password=sha1('$password')";

  $result = $db_conn->query($query);
  if ($result->num_rows)
  {
    // if they are in the database register the user id
    $_SESSION['valid_user'] = $userid;
  }
  $db_conn->close();
}
?>
<html>
<body>
<h1>Home page</h1>
<?
```

```
if (isset($_SESSION[ 'valid_user ]))
{
  echo  You are logged in as:  .$_SESSION[ 'valid_user'].'<br />';
  echo  <a href="logout.php">Log out</a><br /> ;
}
else
{
  if (isset($userid))
  {
    // if they've tried and failed to log in
    echo  Could not log you in.<br />';
  }
  else
  {
    // they have not tried to log in yet or have logged out
    echo 'You are not logged in.<br /> ;
  }

  // provide form to log in
  echo  <form method="post'action="authmain.php">";
  echo  <table> ;
  echo '<tr><td>Userid:</td>';
  echo  <td><input type="text'name="userid'></td></tr>';
  echo '<tr><td>Password:</td>';
    echo '<td><input type-"password'name="password"></td></tr>';
    echo '<tr><td colspan-"2"align-"center">';
    echo '<input type="submit"value='Log in'></td></tr>';
    echo '</table></form>';
  }
?>
<br />
<a href="members_only.php">Members section</a>
</body>
</html>
```

因为它显示了登录表单，所以该脚本包含了一些比较复杂的逻辑；表单的行为也比较复杂，因为它包含了成功和失败登录操作的HTML代码。

脚本的执行是围绕会话变量valid_user展开的。其基本思想是如果某人成功登录，我们将注册一个$_SESSION['valid_user']的会话变量，该变量包含用户的ID。

在该脚本中，我们所做的第一件事就是调用session_start()函数。如果会话变量valid_user已经创建，这个函数调用将载入该变量。

在第一次执行脚本的时候，if条件均不成立，用户将直接进入脚本的末尾，脚本告诉用户尚未登录并提供一个表单以便登录。

```
echo '<form method="post'action="authmain.php">';
echo '<table>';
```

```
echo '<tr><td>Userid:</td>';
echo '<td><input type="text"name="userid"></td></tr>';
echo '<tr><td>Password:</td>';
echo '<td><input type="password"name="password"></td></tr>';
echo '<tr><td colspan="2"align='center'>';
echo '<input type='submit'value="Log in"></td></tr>';
echo '</table></form>';
```

当用户点击表单上的提交按钮后，这个脚本将从顶端开始重新执行。这次，我们有了用以验证的用户名和密码，它们分别存储在$_POST['userid']和$_POST['password']中。如果用户设置了这些变量，我们就可以进入身份验证模块：

```
if (isset($_POST['userid']) && isset($_POST['password']))
{
  // if the user has just tried to log in
  $userid = $_POST['userid'];
  $password = $_POST['password'];

  $db_conn = new mysqli('localhost', 'webauth', 'webauth', 'auth');

  if (mysqli_connect_errno()) {
  echo 'Connection to database failed:'.mysqli_connect_error();
  exit();
}
  $query = 'select * from authorized_users
           ."where name='$userid'"
           ."and password=sha1('$password')';

  $result = $db_conn->query($query);
```

我们连接到一个MySQL数据库并检查用户ID和密码。如果它们与数据库中的数据匹配，将创建变量$_SESSION['valid_user']，它包含该用户的ID，因此我们就知道谁登录进来了，并对其进行跟踪。

```
if ($result->num_rows >0 )
  {
    // if they are in the database register the user id
    $_SESSION['valid_user'] = $userid;
  }
  $db_conn->close();
}
```

由于我们知道该用户是谁，就不必再向他显示登录表单。取而代之，告诉他我们知道他是谁，并提供退出的选项：

```
if (isset($_SESSION['valid_user']))
{
    echo 'You are logged in as: '.$_SESSION['valid_user'].'<br />';
```

```
    echo '<a href="logout.php'>Log out</a><br />';
}
```

如果试图让用户登录，但是由于某种原因失败了，我们将拥有一个用户ID而不是$_SESSION['valid_user']变量，因此我们为该用户提供一个错误信息。

```
if (isset($userid))
{
  // if they've tried and failed to log in
  echo 'Could not log you in.<br />';
}
```

以上就是主要的脚本逻辑。现在，我们来了解成员页面。成员页面的代码如程序清单23-5所示。

**程序清单23-5 members_only.php——网站的有效用户检查部分，可以确定是否是有效成员**

```
<?php
  session_start();

  echo '<h1>Members only</h1>';

  // check session variable

  if (isset($_SESSION['valid_user']))
  {
    echo '<p>You are logged in as '.$_SESSION['valid_user'].'</p>';
    echo '<p>Members only content goes here</p>';
  }
  else
  {
    echo '<p>You are not logged in.</p> ;
    echo '<p>Only logged in members may see this page.</p>';
  }

  echo '<a href='authmain.php'>Back to main page</a>';
?>
```

以上代码非常简单。它所做的只是启动一个会话，并且通过检查$_SESSION['valid_user']变量是否被设置来检查当前的会话是否包含一个注册的用户。如果用户登录进入我们的网站，我们显示成员内容；否则告诉他尚未通过身份验证。

最后，我们将讨论logout.php脚本，该脚本可以让用户退出登录。脚本代码如程序清单23-6所示。

**程序清单23-6 logout.php——该脚本注销会话变量并销毁会话**

```
<?php
  session_start();
```

```
  // store to test if they *were* logged in
  $old_user = $_SESSION['valid_user'];
  unset($_SESSION['valid_user']);
  session_destroy();
?>
<html>
<body>
<h1>Log out</h1>
<?php
  if (!empty($old_user))
  {
    echo 'Logged out.<br />';
  }
  else
  {
    // if they weren't logged in but came to this page somehow
    echo 'You were not logged in, and so have not been logged out.<br />';
  }
?>
<a href="authmain.php">Back to main page</a>
</body>
</html>
```

源代码十分简单，当然，它也只完成了一些简单的工作。在这里，我们启动了一个会话，保存用户的旧用户名，注销valid_user变量，销毁了会话。然后，我们给用户发送一个消息，该消息在用户退出或者尚未登录的情况下的具体内容是不同的。

以上简单的脚本是在后续章节中要完成的许多工作的基础。

## 23.7　进一步学习

关于cookie的更多信息，请访问如下URL：

http: //home.netscape.com/newsref/std/cookie_spec.html。

## 23.8　下一章

本篇内容就此结束。在开始示例项目之前，我们还要先简要讨论一下本书其他部分没有介绍的PHP中的一些零碎但实用的功能。