

第3章 用 PHP 进行 Web 编程

PHP 是开发 Web 应用的首选语言之一，也是最佳选择。PHP 本身就是为 Web 而生的。它提供了一系列可以使 Web 开发更加方便、更加容易的功能和特性。

本章先介绍用 PHP 进行 Web 编程的一些基本用法，了解用 PHP 进行 Web 编程的一些特性，接着通过一个简单的完整实例实践这些用法和特性，加深对使用 PHP 进行 Web 编程的理解和掌握。

3.1 PHP 的 Web 编程基础

本节将讲述最基本的 PHP Web 编程知识，诸如获取表单数据、处理表单数据、PHP 中的 Session 和上传文件等。

3.1.1 访问和获取 HTML 表单数据

在 PHP 中，可以通过两个预定义变量，很方便地获取 HTML 表单数据。这两个预定义变量在前面提及过：\$_GET 和 \$_POST。它们都是 PHP 的自动全局变量，可以直接在 PHP 程序中使用。

- ❑ 变量 \$_GET 是表单数据组成的数组，它由 HTTP 的 GET 方法传递的表单数据组成。表单元素的名称就是数组的“索引”。这就是说，通过表单元素的名称（即 name 属性的值），就可以获得该表单元素的值。例如某表单中，有一个文本输入框，名称为“user_name”，那么在 PHP 程序中，就可以通过 \$_GET['user_name'] 获取文本框中用户输入的值。
- ❑ 变量 \$_POST 的用法和 \$_GET 类似。通过 HTTP 的 POST 方法获取的表单数据，都将存放在该变量中，该变量也是一个数组。

下面通过一个实例，来学习变量 \$_POST 的使用，变量 \$_GET 的用法完全类似。

（1）创建如代码 3-1 所示的 HTML 文档。

代码 3-1 一个含有表单的 HTML 文档 3-1.html

```
<html>
<head>
<title>3-1</title>
</head>

<body>
<form name="form1" method="POST" action="3-2.php">
输入姓名：<input name="user_name" type="text"><br/><br/>
选择性别：<input name="gender" type="radio" value="male"> 男    <input name="gender" type="radio"
value="female">女<br/><br/>
兴趣与爱好：<input name="hobby" type="checkbox" value="reading"> 阅读 <input name="hobby"
type="checkbox" value="travel"> 旅游 <input name="hobby" type="checkbox" value="sport"> 运动 <input
name="hobby" type="checkbox" value="internet">上网<br/><br/>
```

```

选择职业:
<select name="occup">
<option value="engineer">工程师</option>
<option value="teache">教师</option>
<option value="doctor">医生</option>
<option value="other">其他</option>
</select><br/><br/>
<input type="submit" value="提交数据">
</form>

</body>
</html>

```

该 HTML 文档定义了一个表单，其中的 method="POST" 表示用 POST 方法传送表单数据，action="3-2.php" 表示将表单提交给 3-2.php 处理。在表单中定义的元素有：名称为 user_name 的文本框，名称为 gender 的 radio 按钮，名称为 hobby 的 checkbox 多选框，名称为 occup 的下拉列表框。当表单提交时，表单元素的值由 POST 方式交由当前目录下的 3-2.php 处理。

(2) 接下来编写 3-2.php，该程序先获取表单提交的数据，然后将这些数据向浏览器输出。完整的代码如代码 3-2 所示。

代码 3-2 获取表单数据 3-2.php

```

<?php
//通过$_POST 全局变量，获取文本框 user_name 的值，并赋给变量$user_name
$user_name = $_POST['user_name'];
$gender = $_POST['gender'];
$hobby = $_POST['hobby'];
$prof = $_POST['occup'];

echo "用户名: ".$user_name."<br/>";
echo "性别: ".$gender."<br/>";
echo "爱好: ".$hobby."<br/>";
echo "职业: ".$prof."<br/>";
?>

```

(3) 这是个很简单的获取表单数据并且输出数据的 PHP 程序。如果在 3-1.html 页面填写的数据和所做的选择如图 3.1 所示，那么 3-2.php 将输出如图 3.2 所示的结果。

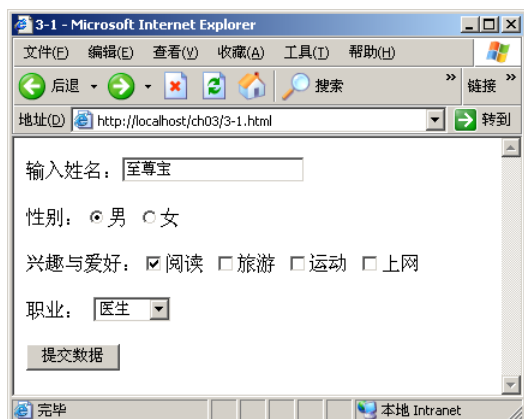


图 3.1 含有表单的 HTML 文档

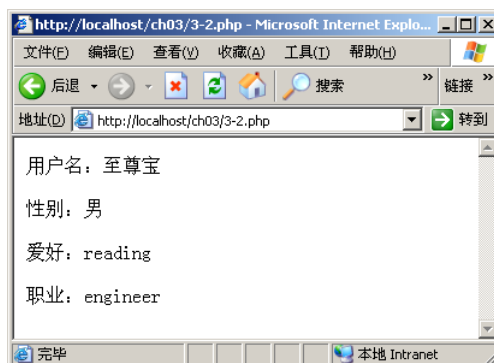


图 3.2 表单数据输出的结果

从上面的结果图示可以看出，PHP 输出的值，就是 HTML 表单元素的 value 属性所赋的值，这些值是当表单提交时，传给全局变量\$_POST 的。表单中每个元素的值，都将以元素的 name 属性的值作为索引，存入到数组变量\$_POST 中。在 PHP 程序中，通过访问\$_POST 数组，来获取 HTML 表单元素的值。

3.1.2 用 PHP 处理表单数据

在上小节的文档 3-1.html 中，对于表单中的“爱好”多选框，只选择了“阅读”一项。如果做了多个选择，再提交表单，3-2.php 输出的结果就有所不同。例如，对 3-1.html 的多选框，做如图 3.3 所示的选择。在 HTML 文档中，对于多选框 occup 选择了“阅读”、“旅游”、“上网”3 项。提交表单后，将看到如图 3.4 所示的结果。

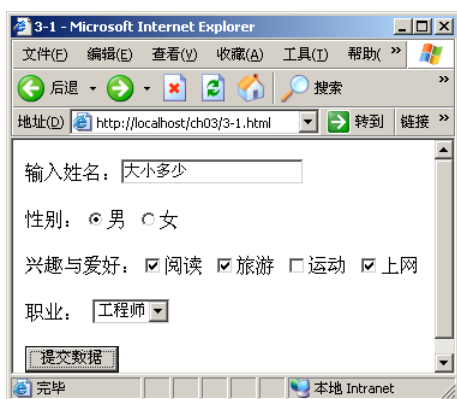


图 3.3 HTML 表单中的多选框

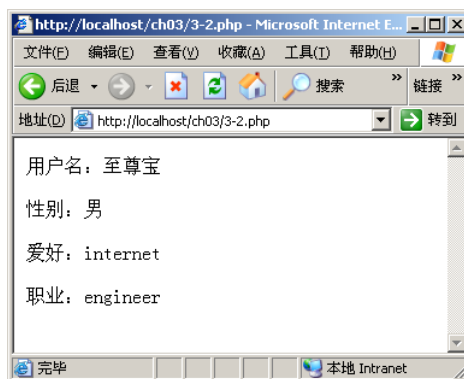


图 3.4 PHP 输出表单数据结果

从图 3.4 看出，所选择的 3 个 checkbox，只有最后 1 个的值被输出了，其他两个选项的值没有被输出，这并不是所期望的结果。之所以出现这种情况，是因为多选按钮元素 checkbox 的名称都为“hobby”，而 PHP 要求，如果表单元素同名，就必须以数组方式命名，并为其 value 属性赋值，这样 PHP 才能正确取值。

因此，首先修改 3-1.html 的中表单元素 checkbox 的名称，以数组方式命名 checkbox 元素，即在原来的名称“hobby”后加上“[]”，修改后的代码如下所示。

```
兴趣与爱好: <input name="hobby[]" type="checkbox" value="reading"> 阅读 <input name="hobby[]" type="checkbox" value="travel"> 旅游 <input name="hobby[]" type="checkbox" value="sport"> 运动 <input name="hobby[]" type="checkbox" value="internet"> 上网<br/><br/>
```

在 3-2.php 中通过\$_POST['hobby'][0]访问 3-1.html 中第 1 个 checkbox 的值，通过\$_POST['hobby'][1]访问 3-1.html 中第 2 个 checkbox 的值，以此类推。修改后的代码如下所示。

```
$hobby = $_POST['hobby'][0].",". $_POST['hobby'][1].",". $_POST['hobby'][2].",". $_POST['hobby'][3];
```

像这样修改 HTML 文档和 PHP 程序之后，再次多选“爱好”项，就会看到所选项的值都被输出，如图 3.5 所示。

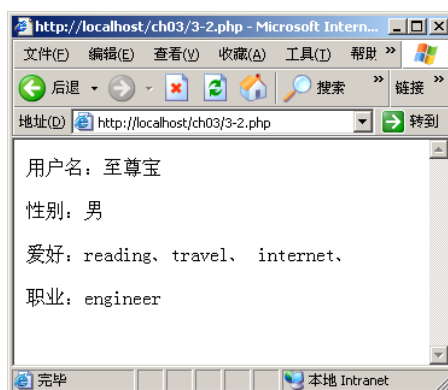


图 3.5 PHP 处理多选按钮元素的输出

提示：因为\$_POST 是一个数组变量，所以，除了使用类似\$_POST['hobby'][0]的方法获取同名 checkbox 元素的值之外，还可以使用另外一种专门用于操作数组的方法，这个方法会在第 4 章讲述数组处理时介绍。

3.1.3 用 PHP 验证表单数据有效性

在实际开发应用中，PHP 程序往往要对用户提交的数据做验证，以保证程序的执行安全和数据的完整、有效。

本小节将在前两小节程序的基础上，加入对提交数据的验证，只有在用户完全提交有效的数据后，程序才会向浏览器输出数据，否则将会向用户输出提示信息。对代码 3-2 做一些修改，使之成为代码 3-3，并按名称 3-3.php 保存在测试目录下。对代码 3-1 所示的 HTML 文档做修改，将表单提交到 3-3.php。

代码 3-3 用 PHP 验证数据 3-3.php

```
<?php
$user_name = $_POST['user_name'];
$gender = $_POST['gender'];
$hobby = $_POST['hobby'][0].". ".$_POST['hobby'][1].". ".$_POST['hobby'][2].". ".$_POST['hobby'][3];
$prof = $_POST['occup'];

//当用户名为空，即没有输入用户名时，则输出一个提示信息，并中断程序的执行
if($user_name == "")
{
    echo "请返回输入用户名！";
    exit;    //exit 语句将使程序立即中断，不再向下执行
}

if($gender == "")
{
    echo "请返回选择性别！";
    exit;
}

if($hobby == "")
{
```

```

    echo "请返回选择兴趣与爱好! ";
    exit;
}

echo "用户名: ".$user_name."<br/>";
echo "性别: ".$gender."<br/>";
echo "爱好: ".$hobby."<br/>";
echo "职业: ".$prof."<br/>";
?>

```

如果没有填写用户名便提交表单，程序就会输出一个提示信息“请返回输入用户名！”，实际效果如图 3.6 所示。

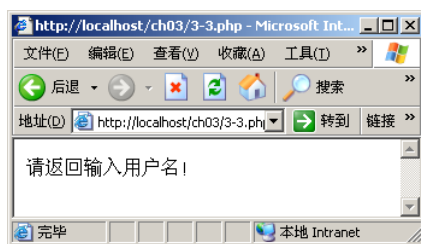


图 3.6 用 PHP 做数据验证

这个程序只对用户的输入值是否为空做了判断。事实上，数据的验证情况远远要比这多，比这复杂，如要求输入的数据只能是数字、限制输入内容的长度等等。随着学习的深入，这些内容将会所有讲述。

3.1.4 PHP 中的 session

session 是 Web 开发中最常见的概念，也是最常用的功能之一。简单地说，是 session 是指用户进入网站到浏览器关闭的这段时间（或过程）。

HTTP 是面向无连接（或无状态）的协议。这意味着，在 HTTP 中，一个完整的请求/响应过程结束之后，客户端（即浏览器）和服务端端的链接就已中断。此时，如果用户再从当前页面访问其他页面，即向服务器发出请求，服务器端并不知道此请求是哪个用户发起的，因此也就无法得知用户的浏览状态。这样就遇到一个问题：当前页面中的某个数据（或变量），无法在接下来访问的页面中使用。而在实际的 Web 开发中，经常要在页面之间传递数据，而且不同的访问用户，传递的数据是不同的。虽然解决这个问题的办法有很多，但通过 session 解决这个问题，会更加方便、快速、有效。通过 session 记录用户的有关信息，以供用户以此身份向服务器发起请求时，服务器能够根据 session 做出正确的判断，区分不同用户的请求。

在 PHP 中使用 session，就是通过注册一些 session 全局变量，在不同页面的程序中使用这些变量。这样就可以通过 session 完成用户身份验证、程序状态和页面之间的数据传递等功能。

一般使用类似 `$_SESSION['session_name']=session_value` 的代码注册一个 session 变量，其用法也和 `$_POST`、`$_GET` 类似。另外，在使用 session 的页面中，需要使用 `session_start()` 函数，它表示开始或返回一个已经存在的 session。这个函数要在浏览器有任何输出之前调用，也就是说，它往往是使用 session 的程序的第一行代码。

下面通过一个实例来理解 session 的使用。这个实例中有两个 PHP 程序，在第 1 个程序中，定义一些 session 变量，如代码 3-4 所示，然后通过第 1 个页面中的链接，请求第 2 个 PHP 程序。在第 2 个程序中，输出通过 session 前一个页面定义的 session 变量，如代码 3-5 所示。

代码 3-4 在 PHP 程序中注册 session 变量 3-4.php

```
<?php
session_start();           //使用 session 前必须调用该函数

$_SESSION['user'] = 'KingKong'; //注册一个 session 变量，变量的值为 “KingKong”
$_SESSION['explain'] = '这是 3-4.php 的 session 变量';
echo '这个页面已经通过 session 保存了一些变量';
echo '<br/><a href="3-5.php">进入 3-5.php</a>查看这些变量值';
?>
```

代码 3-5 在 PHP 程序中取得 session 变量的值 3-5.php

```
<?php
session_start();
echo $_SESSION['user']. "<br/>";
echo $_SESSION['explain']. "<br/>";
echo '<a href="3-4.php">返回 3-4.php</a>';
?>
```

访问 3-4.php，然后通过该页面的“进入 3-5.php”的链接，访问 3-5.php，可以看到如图 3.7 所示的结果。

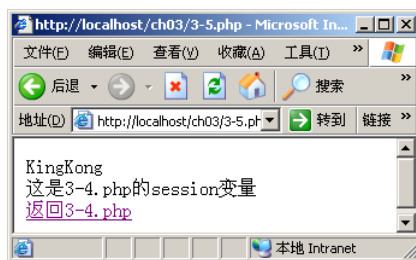


图 3.7 在 PHP 中使用 session 变量

从图 3.7 可以看到，程序 3-5.php 通过 session 取得了在 3-4.php 中注册的 session 变量，实现了数据的跨页面传递。

说明：上述传递 session 变量功能的实现，是基于客户端浏览器支持 cookie 的。cookie 是由服务器端产生的并且保存在客户端的一些文件，里面存放了一些用户信息和数据。有关 cookie 的知识会在第 8 章介绍。因为 PHP 的 session 机制的是通过 cookie 实现的，所以，如果浏览器不支持 cookie，那么上述的示例程序就无法看到预期的效果。

3.1.5 PHP 中的文件上传处理

在 Web 开发中，经常会遇到从客户端上传文件到服务器端的问题。通常，文件上传使用的是 HTTP 的 POST 方式，使用 POST 方式传递文件到服务器端。要完成文件上传处理，首先要定义 HTML 表单的 **enctype** 属性为“multipart/form-data”，如下代码所示。

```
<form enctype="multipart/form-data" action="somefile.php" method="POST">
```

只有这样的表单，才能确保文件可以提交并上传。其中，somefile.php 要替换为一个可以处理文件上传的真实 PHP 文件。代码 3-6 是一个支持文件上传的表单示例，该表单将一个文件提交至 3-7.php 进行处理，稍后完成 3-7.php 程序。

代码 3-6 支持文件上传的 HTML 表单 3-6.html

```
<html>
<head><title>3-6 支持文件上传的表单</title></head>
<body>

<!-- 表单的 enctype 属性必须指定为 multipart/form-data -->
<form enctype="multipart/form-data" action="3-7.php" method="POST">
    <!-- input 的 type 属性指定为 file, name 属性的值将会在 PHP 程序的$_FILES 数组中用到-->
    上传此文件: <input name="myfile" type="file" />
    <input type="submit" value="提交上传" />
</form>

</body>
</html>
```

浏览 3-6.html, 看到如图 3.8 所示的效果。该页面会生成一个“浏览”按钮, 通过它可以选择客户端文件。

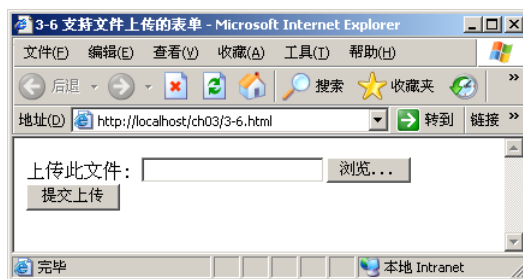


图 3.8 支持文件上传到表单

在 PHP 程序中, 使用全局变量 `$_FILES` 处理文件上传。`$_FILES` 是一个数组, 包含了要上传的文件的信息。下面, 以上述 HTML 表单为例, 介绍 `$_FILES` 数组的内容。

- ❑ `$_FILES['myfile']['name']` 表示客户端文件的原始名称, 即要上传的文件的文件名。其中 `myfile` 就是在代码 3-6 中定义的 `input` 元素的 `name` 属性的值: `<input name="myfile" type="file" />`。
- ❑ `$_FILES['myfile']['type']` 表示上传文件的类型, 例如 “image/gif”。
- ❑ `$_FILES['myfile']['size']` 表示已上传文件的大小, 单位为字节。
- ❑ `$_FILES['myfile']['tmp_name']` 表示文件上传后, 在服务器端存储的临时文件名。
- ❑ `$_FILES['myfile']['error']` 表示和文件上传的相关错误信息。

文件提交后, 一般会被存储到服务器的默认临时目录中, 可以通过修改 `php.ini` 中的 `upload_tmp_dir` 项, 修改为其他路径。使用函数 `move_uploaded_file` 将上传的文件移到指定的目录下。该函数的原型如下所示。第 1 个参数 `filename` 指合法的上传文件, 第 2 个参数 `destination` 是移动后的目标文件。如果上传的文件不合法, 或由于某种原因无法移动文件, 该函数会返回 `FALSE`。

```
move_uploaded_file (filename, destination)
```

代码 3-7 是处理文件上传的一个示例 PHP 程序, 代码 3-6 所示的 HTML 文档中的文件提交后, 将由 3-7.php 来处理。

代码 3-7 处理文件上传的 PHP 程序 3-7.php

```
<?php
//将文件移至服务器的根目录的 upload 目录下, upload 目录要事先建立好
```



```

$upload_path = $_SERVER['DOCUMENT_ROOT']."/upload/";
$dest_file = $upload_path.basename($_FILES['myfile']['name']);

//将临时文件移至目标文件
if(move_uploaded_file($_FILES['myfile']['tmp_name'],$dest_file))
{
    echo "文件已上传至服务器根目录的 upload 目录下";
}
else
{
    echo "文件上传时发生了一个错误".$_FILES['myfile']['error'];
}
?>

```

代码 3-7 首先定义一个存放上传文件的目录，然后通过函数 `move_uploaded_file` 将临时文件移至这个目录下。浏览 3-6.html 文档，选择一个要上传的文件，如图 3.9 所示。单击提交按钮后，如果上传成功，会显示上传成功的信息，如图 3.10 所示。



图 3.9 选择一个要上传的文件



图 3.10 文件上传成功

转到服务器根目录的 `upload` 目录下，将会看到刚刚上传的文件“`read.txt`”，如图 3.11 所示。

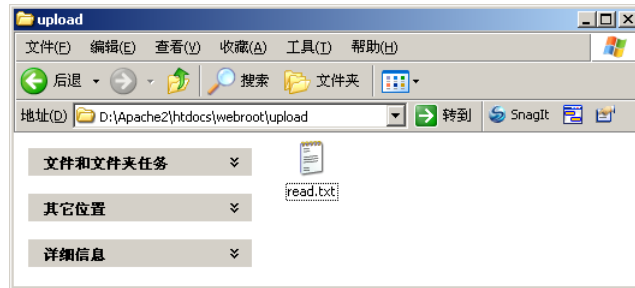


图 3.11 上传的文件在服务器中的位置

3.2 实例：用 PHP 开发一个简单的网站

通过上一小节对 PHP 开发 Web 应用基础知识的学习，读者掌握了使用 PHP 获取和处理表单数据、用 PHP 验证数据的有效完整、PHP 中 `session` 的使用和处理文件上传等基本技能。本节，将通过一个具体的实例来进一步加强对这些技能的应用。本节还将进一步学习界面设计和布局方面的应用。

3.2.1 网站功能设计

本节制作一个简单数据录入系统。首先用户提供登录名和密码登入系统，之后使用 session 维护用户状态。进入系统后，用户可以录入一些个人信息。该系统还应该实现对于不同用户录入的数据，有不同的显示结果。

该系统客户端使用 CSS 完成页面设计和布局，并且使用 JavaScript 验证数据是否有效。客户端使用 PHP 处理这些数据，并将数据显示至浏览器。

3.2.2 页面设计

初步考虑设计两个页面。一个页面是用户登录界面，另一个是用户信息的录入界面。此外还需要 PHP 程序完成用户验证，和负责获取表单提交的数据，并将表单数据显示出来。

- (1) 首先建立用户登录界面，如图 3.12 所示。
- (2) 接下来建立用户信息录入界面，如图 3.13 所示。



图 3.12 用户登录界面



图 3.13 用户信息录入界面

接下来的内容，将介绍如何实现这些页面和 PHP 程序。

3.2.3 用 JavaScript 实现客户端响应

用 JavaScript 可以在客户端验证数据的有效性。在上述登录页面中加入一些 JavaScript 脚本，可以验证用户是否输入了用户名。

代码 3-8 实现了如图 3.12 所示的用户登录界面，其中在<head>标签内加入了 JavaScript 脚本，用以检验用户输入的用户名是否为空，如果为空，则向用户弹出一个提示对话框。此外，该 HTML 文档中还加入了 CSS 代码，定义了页面字体的显示大小、表格单元格背景颜色等页面效果。

代码 3-8 用户登录界面 3-8.html

```
<html>
<head>
<title>3-8.html 用户登录</title>
<style>
.tbl{font-size:10pt;width:30%;text-align:right;background-color:#abcdef;}
</style>
```

```

<script language="JavaScript">
//用来检查用户输入是否为空的函数
function check_name()
{
    //判断表单 login 中，名为 user_name 的 input 元素的值是否为空
    if(login.user_name.value=="")
    {
        alert("请输入用户名！");
        return false;
    }
}
</script>
</head>

<body>
//注意，这里要添加 onsubmit 属性，通过它，当 submit 按钮被单击时，JavaScript 函数 check_name 被调用
<form name="login" action="3-9.php" method="POST" onsubmit="return check_name()">
<table border="0" width="200px" align="center">
    <tr>
        <td class="tbl">用户名: </td><td><input type="text" name="user_name"></td>
    </tr>
    <tr>
        <td><input type="submit" value="登入系统"></td>
    </tr>
</table>
</form>

</body>
</html>

```

在代码 3-8 中，使用 JavaScript 判断用户的输入是否为空，当用户没有任何输入而提交表单时，JavaScript 将会弹出一个警告对话框，如图 3.14 所示，通过这样的方式完成了客户端的响应。

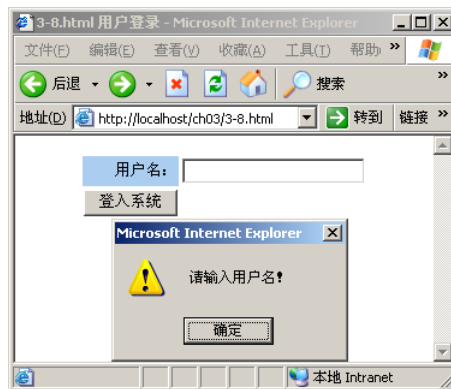


图 3.14 通过 JavaScript 完成客户端响应

3.2.4 服务器端用 PHP 处理请求

当用户输入了用户名后，该数据将提交至一个 PHP 程序做处理。服务器端的 PHP 处理程序需要显示用户刚刚输入的用户名，以及一个录入用户信息的界面。考虑到不同的用户录入的信息都是不同的，

因此对于不同的用户，只能看到自己的录入信息，所以要在程序中使用 session 维护不同用户的状态和数据。

(1) 如代码 3-9，用来处理用户输入的用户名，并且负责显示数据录入界面。代码 3-9 就是常见的在 HTML 文档中内嵌 PHP 代码的文件。

代码 3-9 处理用户名并且显示信息录入界面 3-9.php

```
<?php
session_start();           //开始 session

$user = $_POST['user_name']; //获取 3-8.html 传入的用户名
if(!empty($user))
{
    $_SESSION['user'] = $user; //将用户名注册到 session 变量中
    $welcome = "您好, ".$user."! 请录入以下信息后提交。<br/>";
}
?>
<html>
<head>
<title>3-9.php 用户信息录入</title>
</head>

<body>
<?php
    echo $welcome;           //显示一条欢迎信息
?>
<form name="info" action="" method="POST">
<table border="0">
    <tr><td> 性别: </td><td><input name="gender" type="radio" value="男"> 男 <input name="gender"
type="radio" value="女">女</td></tr>
    <tr><td>年龄: </td><td><input name="age" type="input" size="3"></td></tr>
    <tr>
        <td>血型: </td>
        <td>
            <select name="blood_type">
                <option value="A">A 型</option>
                <option value="B">B 型</option>
                <option value="O">O 型</option>
                <option value="AB">AB 型</option>
                <option value="其他">其他血型</option>
            </select>
        </td>
    </tr>
    <tr><td><input type="submit" value="提交"></td></tr>
</table>
</form>

</body>
</html>
```

(2) 当用户输入用户名并提交后, 上述程序将会输出一句问候语, 然后是用户信息的录入界面, 如图 3.15 所示。

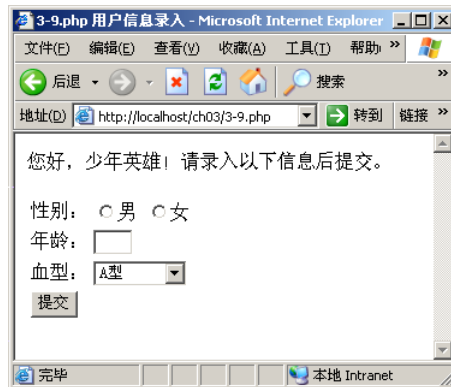


图 3.15 用户登录后看到的界面

(3) 接下来, 还需要一个处理录入信息的 PHP 程序。事实上, 这个程序可以在代码 3-9 中实现。在代码 3-9 的基础上, 加入一些对 3-9.php 中表单数据是否提交的判断, 就可以实现对录入信息的处理。如果用户没有提交数据, 那么, 仍然显示信息录入界面。代码 3-10 就是在代码 3-9 的基础上做一些修改, 用来完成对用户录入信息的处理, 并输出用户信息。

代码 3-10 处理用户录入信息 3-10.php

```
<?php
session_start();

$user = $_POST['user_name'];
if(!empty($user))
{
    $_SESSION['user'] = $user;
    $welcome = "您好, ".$user."! 请录入以下信息后提交。<br/>";
}

$gender = $_POST['gender'];
$age = $_POST['age'];
$blood = $_POST['blood_type'];

//如果当前用户提交了数据, 则输出这些数据
if(!empty($gender) && !empty($age) && !empty($blood))
{
    echo "性别: ".$gender."<br/>";
    echo "年龄: ".$age."<br/>";
    echo "血型: ".$blood."<br/>";
}
//如果用户没有提交数据, 则显示信息录入界面
else
{
    ?>

<html>
```

```

<head>
<title>3-9.php 用户信息录入</title>
</head>

<body>
<?php
    echo $welcome;
?>
<form name="info" action="" method="POST">
<table border="0">
    <tr><td> 性别 : </td><td><input name="gender" type="radio" value="男">男 <input name="gender"
type="radio" value="女">女</td></tr>
    <tr><td>年龄: </td><td><input name="age" type="input" size="3"></td></tr>
    <tr>
        <td>血型: </td>
        <td>
            <select name="blood_type">
                <option value="A">A 型</option>
                <option value="B">B 型</option>
                <option value="O">O 型</option>
                <option value="AB">AB 型</option>
                <option value="其他">其他血型</option>
            </select>
        </td>
    </tr>
    <tr><td><input type="submit" value="提交"></td></tr>
</table>
</form>

</body>
</html>
<?php
}
?>

```

(4) 将 3-8.html 中表单提交后的处理程序改为 3-10.php，即将表单的 action 属性改为“3-10.php”，输入用户名后提交至 3-10.php。在 3-10.php 生成的页面中，录入用户信息后提交，此时 3-10.php 中的表单数据提交到自身（3-10.php）。然后再通过程序判断，输出用户所录入的信息。

3.3 小结

本章讲述了使用 PHP 进行 Web 编程的基础知识和基本技能。包含：用 PHP 获取 HTML 表单数据、用 PHP 处理表单数据、用 PHP 验证数据、PHP 中 session 的使用及使用 PHP 处理文件的上传。最后通过一个简单实例，加深对这些基本技能的实践和掌握。