

第 1 章 PHP 起步

欢迎来到 PHP 的世界！

PHP 语言是最受欢迎的 Web 开发语言之一。它以学习简单、开发快速、性能稳定而倍受 Web 开发人员的青睐。PHP 不仅使用人员众多、数以万计的 Web 站点用它构建，而且有强大的社区支持，使得无论是用 PHP 开发 Web 应用，还是学习 PHP 语言，都会快速有效、事半功倍。在各种 Web 开发语言、框架、概念纷扰的今天，PHP 仍以其独特魅力吸引更多的 Web 开发人员学习和使用。

选择 PHP，没错的！

1.1 初识 PHP

PHP 是一种服务器端的、嵌入 HTML 的脚本语言。它的语法结构和 C 语言极为相似。为了能够快速编写动态 Web 页面，PHP 还加入了自己的某些语言特征，这些特征都非常容易理解和使用。先来看一个嵌入有 PHP 代码的 HTML 文档。

```
<html>
<head><title>Welcome</title></head>
<body>
<?php
    echo "Welcome to PHP's world!";
?>
</body>
</html>
```

上面代码中由“<?php”和“?>”所包含部分，即“echo "Welcome to PHP's world!";”，就是 PHP 代码，这个 PHP 最终生成的 HTML 文档如下所示。

```
<html>
<head><title>Welcome</title></head>
<body>
Welcome to PHP's world!
</body>
</html>
```

事实上，最基本的 PHP 编程，其实就是在 HTML 文档中嵌入一些 PHP 代码，这些代码经 Web 服务器执行后，产生的内容和其他的 HTML 组合在一起，从而生成用户所看到的 HTML 文档。一般情况下，用 PHP 控制 Web 页面的动态内容，用 HTML 构建静态内容。

凡是有 C 语言基础的读者，都可以轻松学习和理解 PHP。因为 PHP 语法结构简单，并且提供了大量预定义变量和函数，即便没有任何编程语言基础的读者，通过阅读本书，也可以轻松学习和掌握 PHP。

1.2 LAMP——锋利四剑客：Linux、Apache、MySQL 和 PHP

要想学习 PHP，就有必要了解和 PHP 关系密切的其他 3 种技术：Linux 操作系统、Apache 网络服务器和 MySQL 数据库。

LAMP 这个名词最早由 Michael Kunze 创造，用来代表 Linux 操作系统、Apache 网络服务器、MySQL 数据库和 PHP（如果可以，Perl 和 Python 也是不错的选择），LAMP 正是这 4 种技术的首字母。

PHP 作为强有力的 Web 开发语言，和 Linux、Apache、MySQL 的支持是密不可分的。它们都是开源软件，并且有强有力的社区支持，它们的完美组合构成了当今 Web 开发世界中不可忽视的一极重要力量。而且，这支力量近年一直在不断地变得更加强大。虽然，这 4 种技术并不是专门被设计成在一起工作的，但多年来，这些软件之间的兼容性不断完善，不仅完改善了个组件之间的协作，扩展出更多的功能，而且在目前几乎所有的 Linux 版本中都默认包含了这些产品，使得这些产品共同组成了一个强大的 Web 应用平台。

注意：LAMP 并不单纯代表了这 4 种技术，更深层面的是，LAMP 是 1 种 Web 应用开发的解决方案，可以用这个解决方案构建、运行各种商业应用和其他各种网络应用。因为是开源解决方案，因而也更加有竞争力和吸引力，LAMP 无论在质量、性能还是价格方面都成为各行业在信息化时不得不考虑的平台。

图 1.1 说明了 LAMP 的架构体系。

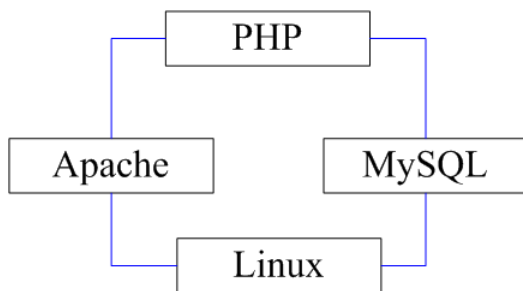


图 1.1 LAMP 体系架构

Linux 处于 LAMP 体系的最下层，提供 Apache 和 MySQL 的运行环境。PHP 位于 LAMP 体系的最上层，由 Apache 支持对 PHP 代码的解析，同时 PHP 和 MySQL 交互，完成对数据库的操作。

要完全掌握 LAMP 就必须对 Linux、Apache、MySQL 和 PHP 都有全方位的认识，如 Linux 的维护、Apache 的配置、MySQL 的维护等。在这里之所以介绍 LAMP，主要是让初学者对 PHP 开发的底层架构的某些方面有个了解，这样可以帮助初学者学习 PHP 及其相关知识。本书主要介绍 PHP 语言及其在 Web 开发方面的应用，因此，对 Linux 系统的操作、维护方面的内容基本没有涉及，对 Apache 的配置也限制在和 PHP 开发有关的几个点上，而对 MySQL 的介绍相对多些，因为 Web 开发几乎都要建立在数据库系统之上。

1.3 PHP Web 编程的体系结构和基本内容

概括地说，任何 Web 编程体系，无外乎服务器端和客户端的程序开发。基于 PHP 的 Web 应用开发也不例外，其中客户端的开发包括使用 HTML 设计 Web 页面，使用样式表控制 Web 页面的显示效果，

还需要客户端的脚本语言来控制浏览器的特效、验证 HTML 表单数据等，这些脚本语言包括 JavaScript、VBScript 等。服务器端的开发就需要掌握 PHP 语言和 MySQL 数据库的有关内容。

本节的各小节将对这些内容做简要说明，之后的各节将较为详细地介绍 HTML、样式表和 JavaScript，它们都是 Web 开发的基础内容，对这些内容不了解的 PHP 初学者有必要掌握。

1.3.1 了解 PHP、HTML、层叠样式表（CSS）和 JavaScript 及其关系

作为服务器端的脚本语言，PHP 多数情况下都是和 HTML 相互搭配来使用的。PHP 用来完成和逻辑有关的动态内容。PHP 程序执行的输出结果，通过 HTML 文档表现给用户。一般情况下，在 HTML 页面中需要输出数据的地方内嵌入 PHP 代码，这个页面也就成为了 PHP 程序。

在创建 HTML 页面时，都要考虑页面设计。如页面布局、页面颜色、字体、边距空白等。这些设置可以单独在页面中进行，但那样会使 HTML 页面变得臃肿，也不便于将来的维护。层叠样式表解决了这些问题，它允许页面设计人员在层叠样式表（即 CSS）文件里设定页面表现效果，如字体大小、边距控制等。CSS 使得 HTML 页面的表现更加丰富、美观，也更加容易维护。

有时，在 HTML 页面中需要对浏览器进行控制，如新建窗口、模拟浏览器的返回按钮等。这就需要在客户端使用 JavaScript 完成这种效果。在与 HTML 表单进行交互时，比如验证用户提交的数据，也可以通过 JavaScript 实现。JavaScript 还可以用来处理用户 cookie。

简单地说，PHP 在服务器端执行，执行结果会通过 HTML 页面展示给客户端。HTML 页面的一些样式需要通过 CSS 来设置，以便更丰富地设计 Web 页面，而对 HTML 表单数据的验证可以通过 JavaScript 来实现。

1.3.2 HTML 文档

HTML 文档就是使用 HTML 标记语言创建的文档。通过浏览器浏览的网页，几乎都是 HTML 文档，或者是由服务器端程序生成的 HTML 文档。下面就是一个 HTML 文档的代码。

```
<html>
<head><title> HTML 文档示例</title></head>
<body>
<h1>HTML 语言</h1>
<p>First Web Page</p>
</body>
</html>
```

每个 HTML 文档由 HTML 标签和文档内容构成。也可以这么说，通过 HTML 标签为一个普通文档加上标记，就构成一个 HTML 文档。例如上面的文档，实际内容（这里将文档标题除外）如下。

```
HTML 语言
First Web Page
```

对“HTML 语言”，加注标记“<h1>”和“</h1>”，即使之成为一级标题。

对“First Web Page”，加注标记“<p>”和“</p>”，即使之成为一个段落。

提示：浏览器通过分析 HTML 标签后，显示出带有一定格式的文档内容，而 HTML 标签本身并不会被显示出来。

1.3.3 使用样式表实现页面效果

按照 HTML 的设计初衷，文档的内容、结构与格式是分离开来的。如果过多地在 HTML 文档中控制页面的显示效果，会使 HTML 文档的维护越来越难以维护，也违背了 HTML 的设计初衷。为了解决这个问题，引入了样式表的概念。一个简单的样式表如下所示。

说明：样式表负责控制页面的格式和显示效果，HTML 文档负责控制结构和显示内容。

```
p{
font-size:10pt;
}
```

这个样式表定义的含义是，HTML 文档中所有用<p>标签定义的内容，都将使用 10pt 大小的字体显示。关于样式表语法和如何使用样式表，将在后面做讲述。

1.3.4 客户端的响应

Web 页面作为客户端，有时需要直接对用户的请求作出响应。这主要是通过客户端脚本来实现，这些脚本语言包括 JavaScript 等。

可以在 HTML 页面中加入 JavaScript 脚本，这些脚本可以直接在浏览器里执行。这样，就可以达到在客户端响应用户请求的需求。一般这样的响应主要是验证表单数据、用户操作提示等。

本书将在 1.6 节介绍 JavaScript。

1.3.5 嵌入式脚本页面

PHP 代码一般都是嵌入在 HTML 文档当中，通过服务器解释这些 PHP 代码，并用代码执行产生的结果替换 PHP 代码内容，最后返回给用户的是内嵌代码执行后的 HTML 文档。例如内嵌生成“Hello World”的 HTML 页面代码如下：

```
<html>
<head>
<title>HTML-PHP</title>
</head>
<body>
<h2><?php echo "Hello World"; ?></h2>
</body>
</html>
```

其中<?php echo "Hello World"; ?>就是内嵌在 HTML 文档中的 PHP 代码。当用户浏览该页面时，实际返回的结果如下：

```
<html>
<head>
<title>HTML-PHP</title>
</head>
<body>
<h2>Hello World</h2>
</body>
</html>
```

1.4 Web 编程的基础知识之一：HTML

HTML 是创建 Web 应用的最基本内容，无论是动态还是静态页面，最终都要产生 HTML 文档。所有的 Web 开发都要涉及到用 HTML 设计 Web 页面。本节将介绍 HTML 语言及如何使用 HTML 创建 Web 页面。

1.4.1 HTTP 协议简介

网络上的计算机之间要进行通信，就必须遵守一定的规则，这种通信规则就是网络协议。协议保证网络上各种不同的计算机之间能够理解彼此传递的消息，好比操不同语言的人们之间，通过翻译来理解对方所说话的含义一样。现在应用最广的 Internet 使用的是 TCP/IP 协议，而浏览 WWW 使用的是 HTTP 协议，即超文本传输协议（HyperText Transfer Protocol），此协议建立在 TCP/IP 协议之上。

浏览网页的过程，其实就是一系列请求/响应的过程。HTTP 协议定义了这个请求/响应过程中请求和响应的格式，及维护 HTTP 链接的内容。

用户使用浏览器浏览一个 Web 站点，首先通过一个网址，向网络中的某台计算机（即服务器）发出请求，请求浏览某个页面。服务器在找到这个页面后，在响应中返回相应页面的内容。图 1.2 表现了这个请求、响应过程。

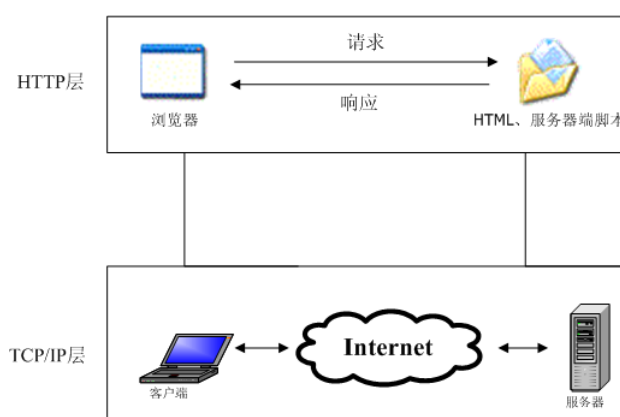


图 1.2 HTTP 请求和响应

1.4.2 HTML 基本知识：标签

HTML 的全称是 HyperText Markup Language，即超文本标记语言。它是一种简单、通用的标记语言。之所以叫标记语言，是因为，HTML 通过不同的标签，来标记文档的不同部分。读者看到的每个 Web 页面，都是由 HTML 通过一系列定义好的标签生成的。

从简单的文本编辑器，如 Windows 的记事本，到专业化的编辑工具，如 Dreamweaver，都可以用来编辑 HTML 文档，编辑好的 HTML 文档必须按后缀.html 或.htm 来保存，最后，通过浏览器打开 HTML 文档，来查看页面效果。

在 HTML 文档中，标签是包含在“<”和“>”之间的部分，如<p>就是一个标签。标签一般是成对使用的，如和同时使用，其中是开始标签，是结束标签。HTML 的标签不区分大小写，因此和表示的含义相同。表 1-1 列举了常见的 HTML 标签及其含义。

表 1-1 常见的HTML标签及其含义

标签	含义
<html>	定义一个HTML文档，结束标签为</html>。
<head>	定义文档头部信息，结束标签为</head>。
<body>	定义HTML的文档体（即实际内容），结束标签为</body>。
<p>	定义一个段落，结束标签为</p>。
 	在文档中插入一个换行，或使用 。HTML会忽略空白字符（如换行符），因此使用此标签来生成新行。
<!--	定义一个注释，注释结束使用-->，HTML文档的注释内容不会在Web页面中显示出来。

图 1.3 展示了一个 HTML 文档的基本结构，并对其做了说明。

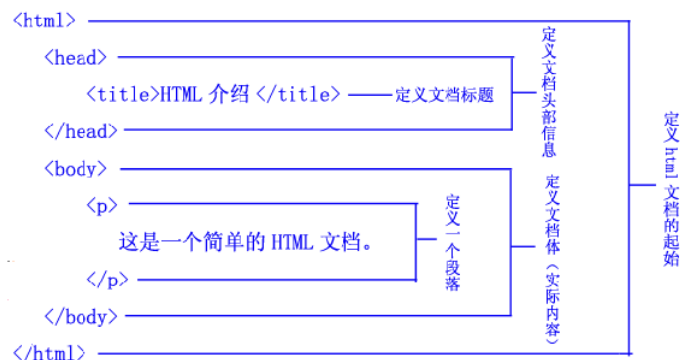


图 1.3 HTML 文档的基本结构

1.4.3 HTML 基本知识：元素

HTML 元素由标签定义，标签所定义的内容就叫“元素”，元素包含在开始标签和结束标签之间。分析代码 1-1 所示的 HTML 文档。

代码 1-1 HTML 文档示例 1-1.html

```
<html>
<head><title> HTML 文档示例</title></head>
<body>
<h1>HTML 文档</h1>
一个简单的 Web 页面
</body>
</html>
```

上述 HTML 文档的如下部分：

```
<h1>HTML 文档</h1>
```

就是一个 HTML 元素，“HTML 文档”是这个元素的内容。

```
<body>
<h1>HTML 文档</h1>
一个简单的 Web 页面
</body>
```

上面这段 HTML 代码同样也是 HTML 元素，这个元素以<body>为开始标签，以</body>为结束标签。用浏览器打开 1-1.html 后，将看到如图 1.4 所示的效果。



图 1.4 HTML 基本元素

每一种 HTML 元素，一般都会有一个或数个属性，属性用来设置或表示元素的一些特性、名称或显示效果等。属性放在元素标签中，紧跟标签名称之后，它和标签名称之间有一个或数个空格。元素的每个属性都有一个值，属性的值的设定使用“属性=”值”的格式，可以为属性的‘值’加上引号或不加引号。下面的 HTML 代码为标签<form>设置了 name 属性，其值为 login，表示这个表单的名称为 login。

```
<form name="login">
```

1.4.4 HTML 基本知识：HTML 的基本元素

元素是组成 HTML 文档的关键，本节介绍 HTML 常用的几种基本元素。

1. 标头元素

HTML 使用标签<head>定义一个标头，结束标签是</head>。一般在<head>标签中设置文档的全局信息，如 HTML 文档的标题（title）、搜索引擎关键字（keyword）等。HTML 文档的标题放在在头元素里，使用<title>标签定义。下面的代码 1-2.html 是一个只有文档标题的 HTML 文档。

代码 1-2 只有文档标题的 HTML 文档 1-2.html

```
<html>
<head><title>HTML 文档示例</title></head>
<body>
</body>
</html>
```

用浏览器访问 1-2.html，将看到一个空白的 HTML 页面，但在浏览器窗口上，可以看到 HTML 的标题。显示效果如图 1.5 所示。



图 1.5 HTML 文档的标题

2. 标题元素

这里的标题，是指 HTML 文档中，内容的标题。标题元素由标签<h1>到<h6>定义。<h1>定义最大的标题，<h6>定义最小的标题。下面的示例代码 1-3.html 显示了所有的 HTML 标题，显示效果如图 1.6

所示。

代码 1-3 HTML 标题元素 1-3.html

```
<h1>标题 1</h1>
<h2>标题 2</h2>
<h3>标题 3</h3>
<h4>标题 4</h4>
<h5>标题 5</h5>
<h6>标题 6</h6>
```

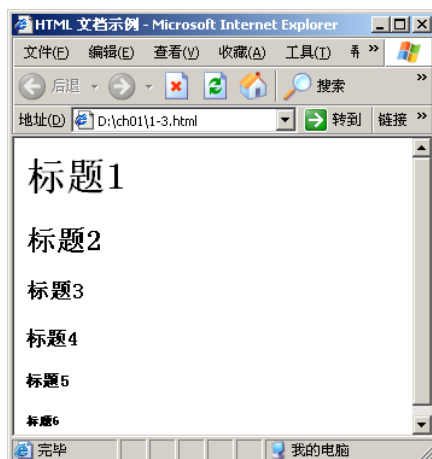


图 1.6 标题元素效果图

3. 段落元素

HTML 中使用标签<p>和</p>定义一个段落。代码 1-4.html 显示了两个段落，显示效果如图 1.7 所示。

代码 1-4 HTML 的段落元素 1-4.html

```
<p>这段文字构成了一个段落元素</p>
<p>这段文字构成了另一个段落元素</p>
```

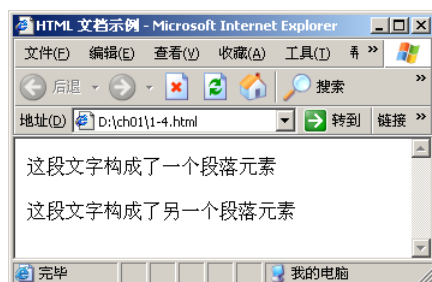


图 1.7 段落元素效果图

4. 字形元素

使用标签和定义一个粗体字形元素，如代码 1-5.html 中使用Some Text显示一个粗体字体。

代码 1-5 粗体字标签和 1-5.html

```
<html>
```



```
<head><title>HTML 文档示例</title></head>
<body>
<b>Some Text</b>
</body>
</html>
```

显示效果如图 1.8 所示。



图 1.8 粗体字形效果图

使用标签<u>和</u>定义一个下划线字形元素，如代码 1-6.html 中使用<u>Some Text</u>显示一个带下划线的字体。

代码 1-6 下划线字体标签<u>和</u> 1-6.html

```
<html>
<head><title>HTML 文档示例</title></head>
<body>
<u>Some Text</u>
</body>
</html>
```

显示效果如图 1.9 所示。

使用标签<i>和</i>定义一个斜体字形元素，如代码 1-7.html 中使用<i>Some Text</i>显示一个斜体字体，

代码 1-7 斜体字体标签<i>和</i> 1-7.html

```
<html>
<head><title>HTML 文档示例</title></head>
<body>
<i>Some Text</i>
</body>
</html>
```

显示效果如图 1.10 所示。

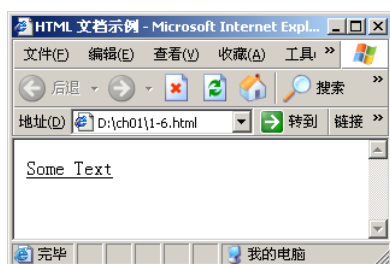


图 1.9 下划线字形效果图

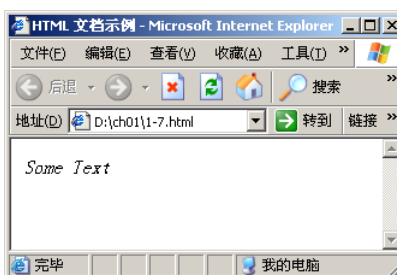


图 1.10 斜体字形效果图

5. 链接

HTML 文档中指向其他 Web 资源，如另一个 HTML 页面、图片等的链接被称为“锚”。在 HTML 中使用标签<a>和定义一个锚元素，即链接元素，也就是说在<a>和之间的内容，会成为一个超链接。

6. 图像元素

使用标签定义一个图片元素，在标签中使用属性 src 来指向一个图片资源，象这样，其中 url 是指向资源所在位置。这个位置可以是一个 URL，也可以是一个相对地址，如，这时，图片 one.gif 和 HTML 文档在同一目录下。

7. 表格元素

使用标签<table>和</table>定义一个表格元素。一个表格由“行”构成，每一行由数据单元构成。表格的“行”用标签<tr>和</tr>定义，数据单元用标签<td>和</td>定义。代码 1-8 所演示的 HTML 文档显示了一个完整的表格，效果如图 1.11 所示。

代码 1-8 HTML 的表格元素 1-8.html

```
<table border="1">
<tr>
<td>第一行数据单元 1</td>
<td>第一行数据单元 2</td>
</tr>
<tr>
<td>第二行数据单元 1</td>
<td>第二行数据单元 2</td>
</tr>
</table>
```



图 1.11 HTML 表格效果图

8. 列表元素

说明：HTML 的列表分为无序列表和有序列表。

无序列表用标签和定义，每一个列表项用标签和定义。代码 1-9 所演示的 HTML 文档显示了一个无序列表，它有 3 个列表项。显示效果如图 1.12 所示。

代码 1-9 HTML 的无序列表元素 1-9.html

```
<ul>
<li>PHP</li>
<li>Python</li>
<li>Perl</li>
```

```
</ul>
```

有序列表用标签``和``定义，每一个列表项用标签``和``定义。代码 1-10 所演示的 HTML 文档显示了一个有序列表，它有 3 个列表项，显示效果如图 1.13 所示。

代码 1-10 HTML 的有序列表元素 1-10.html

```
<ol>
<li>C++</li>
<li>Java</li>
<li>Pascal</li>
</ol>
```



图 1.12 HTML 无序列表效果图

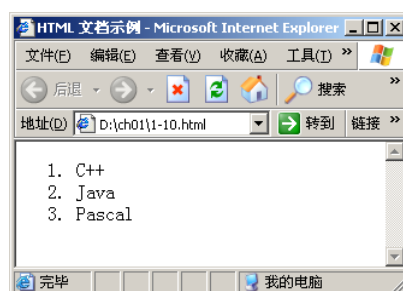


图 1.13 HTML 有序列表效果图

9. 表单元素

HTML 表单是一个包含表单元素的区域，表单元素一般会作为数据，提交给后台服务器做处理。表单域用标签`<form>`和`</form>`定义。表单元素是那些定义在表单域里，可以输入信息的元素，如文本框、单选按钮、下拉列表等。

最常用的表单标签是`<input>`，标签的类型由标签属性 `type` 决定，不同的标签类型，显示出不同的表单元素。用`<input>`的属性“`type="text"`”来定义普通文本输入框，属性“`type="password"`”来定义密码输入框。代码 1-11 所演示的 HTML 文档显示一个文本输入框和一个密码输入框，显示效果如图 1.14 所示。

代码 1-11 HTML 表单元素之文本输入框 1-11.html

```
<form>
用户名: <input type="text" name="user_name">
<br/>
密码: <input type="password" name="pwd">
</form>
```

用`<input>`的属性“`type="radio"`”来定义单选按钮。代码 1-12 所演示的 HTML 文档显示一组单选按钮，显示效果如图 1.15 所示。

代码 1-12 HTML 元素之单选按钮 1-12.html

```
<form>
男<input type="radio" name="sex" value="male">
女<input type="radio" name="sex" value="female">
</form>
```



图 1.14 文本输入框效果图



图 1.15 单选按钮效果图

注意：要在多个单选按钮中实现单选效果，那么这几个单选按钮 name 属性的值必须相同，即定义为同一组单选按钮。

用<input>的属性“type=checkbox”定义复选框。代码 1-13 所演示的 HTML 文档显示一组复选框，显示效果如图 1.16 所示。

代码 1-13 HTML 表单元素之复选框 1-13.html

```
<form>
我喜欢 PHP<input type="checkbox" name="script" value="PHP">
我喜欢 JSP<input type="checkbox" name="script" value="JSP">
我喜欢 ASP<input type="checkbox" name="script" value="ASP">
</form>
```

表单元素的值，一般都会被提交到后台服务器的某个程序做处理。这个提交的动作，由表单域的提交按钮完成。用<input>的属性“type=submit”来定义一个提交按钮。代码 1-14 所演示的 HTML 文档显示一个表单域，其中包含有一个提交按钮，显示效果如图 1.17 所示。

代码 1-14 HTML 表单元素之提交按钮 1-14.html

```
<form name="form_1" action="afile.php" method="post">
用户名: <input type="text" name="user_name"><br/>
<input type="submit" value="提交">
</form>
```



图 1.16 复选框效果图



图 1.17 提交按钮效果图

1.4.5 创建 HTML 文档

使用任何一款文本编辑器，都可以编辑 HTML 文档。编辑好的 HTML 文档，按后缀名.html 或 .htm 保存，最后通过浏览器访问 HTML 文档。打开一个文本编辑器，键入如下 HTML 代码（代码 1-15），并按名称 1-15.html 保存到硬盘某个位置。

代码 1-15 用 HTML 创建 Web 页面 1-15.html

```
<html>
<head>
  <title>TEST HTML</title>
</head>

<body>
  <p>HTML 的全称是 HyperText Markup Language，即超文本标记语言，它是一种简单、通用的标记语言。之所以叫标记语言，是因为，HTML 通过不同的标签标记文档的不同部分。我们看到的每个 WEB 页面，都是由 HTML 通过一系列定义好的标签生成的。</p>
  <h1>HTML 的基本元素</h1>
  <ul>
    <li>标题元素</li>
    <li>段落元素</li>
    <li>链接元素</li>
    <li>列表元素</li>
    <li>图像元素</li>
    <li>表格元素</li>
    <li>表单元素</li>
  </ul>

  <h1>表单元素举例</h1>
  <p>
    <form>
      您的 Email: <input type="text" name="email"><br/><br/>
      您是如何知道本书的: <br/>
      书店 <input type="radio" name="from" value="store"><br/>
      网络 <input type="radio" name="from" value="internet"><br/>
      朋友介绍 <input type="radio" name="from" value="other"><br/>
      <input type="submit" value="提交">
    </form>
  </p>
</body>

</html>
```

用浏览器打开 1-15.html，将看到如图 1.18 所示的效果。

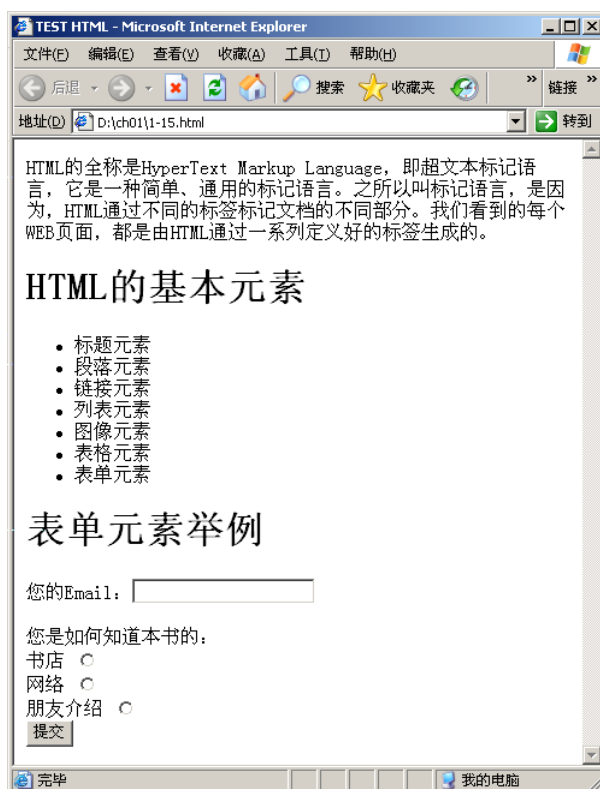


图 1.18 一个 Web 页面

为了增加 HTML 文档的可读性，在编辑 HTML 文档时，不同级的元素之间最好有层次缩进。如下代码所示。

```
<head>
  <title>TEST HTML</title>
</head>
```

上述 HTML 代码的表单域部分，其表单元素的值其实是无法提交的，因为表单域没有设置提交动作属性，有关提交动作的属性问题将在后面讲述。

1.4.6 创建 HTML 列表

HTML 最基本的列表有两种：无序列表和有序列表，列表可以嵌套使用，亦可同时使用。

- ❑ 创建无序列表，通过设置标签的 type 属性，可以更改列表项的标志。type 属性的取值可以是 disc, square 或 circle 中的任意一个。
- ❑ 创建有序列表，通过设置标签的 type 属性，可以为有序列表的列表项设置不同的顺序标志。对于无序列表来说，type 取值可以是表 1-2 中列举的任意一个。

表 1-2 标签的type属性取值

type属性值	顺序标志
1	按阿拉伯数字排序，如：1，2，3……
A	按大写字母排序，如A，B，C……
a	按小写字母排序，如a，b，c……
I	按大写罗马数字排序，如I，II，III……
i	按小写罗马数字排序，如i，ii，iii……

下面举例说明无序列表和有序列表的应用。代码 1-16 所演示的 HTML 文档显示了使用 type 属性创建的无序列表，显示效果如图 1.19 所示。

代码 1-16 HTML 不同形式的无序列表 1-16.html

```
<html>
<head>
<title>LIST TEST</title>
</head>
<body>
  <p>以下列表 type 属性值取 disc</p>
  <ul type="disc">
    <li>北京</li>
    <li>上海</li>
  </ul>

  <p>以下列表 type 属性值取 square</p>
  <ul type="square">
    <li>北京</li>
    <li>上海</li>
  </ul>

  <p>以下列表 type 属性值取 circle</p>
  <ul type="circle">
    <li>北京</li>
    <li>上海</li>
  </ul>

</body>
</html>
```

代码 1-17 所演示的 HTML 文档创建一个按大写字母排序的有序列表，显示效果如图 1.20 所示。

代码 1-17 HTML 按字母排序的有序列表 1-17.html

```
<html>
<head>
  <title>LIST TEST</title>
</head>
<body>
  <ol type="A">
    <li>Linux</li>
    <li>Apache</li>
    <li>MySQL</li>
    <li>PHP</li>
  </ol>
</body>
</html>
```

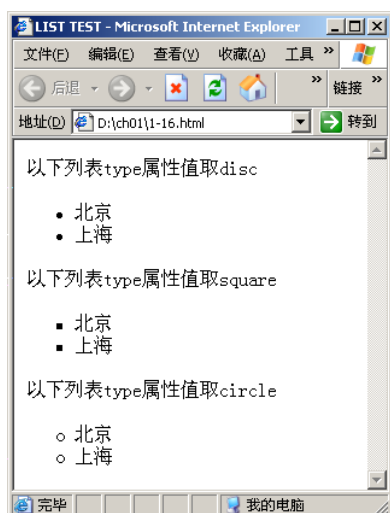


图 1.19 不同形式的无序列表



图 1.20 按字母排序的有序列表

注意：如果不设置 type 属性，有序列表将默认按阿拉伯数字排序。

1.4.7 创建页面表格

标签<table>和</table>用来创建一个表格。标签<table>主要的常用属性有 width、align 和 border。

❑ width 属性：设置表格的宽度，如下所示。这里定义一个宽度为 600px（600 点像素）的表格。

```
<table width="600px">
```

❑ align 属性，设置表格相对于浏览器区域的对其方式，可以选取的值有：left（居左）、center（居中）或 right（居右）。

❑ border，设置表格的边框宽度，单位为像素。

按下面的示例代码 1-18 创建 HTML 文档，按文件名 1-18.html 保存。该 HTML 文档定义了 3 个表格，分别居左，居中，居右显示，宽度分别为 200px，300px，400px，边框的宽度分别为 1px，0px，2px。显示效果如图 1.21 所示。

代码 1-18 具有不同属性的表格 1-18.html

```
<html>
<head>
  <title>table 测试</title>
</head>

<body>
  <table width="200px" align="left" border="1px">
    <tr><td>单元 1</td><td>单元 2</td></tr>
    <tr><td>单元 3</td><td>单元 4</td></tr>
  </table>
  <br/>
  <br/>
  <br/>
  <br/>
  <table width="300px" align="center" border="0px">
```



```

<tr><td>单元 1</td><td>单元 2</td></tr>
<tr><td>单元 3</td><td>单元 4</td></tr>
</table>
<br/>

<table width="400px" align="right" border="2px">
<tr><td>单元 1</td><td>单元 2</td></tr>
<tr><td>单元 3</td><td>单元 4</td></tr>
</table>
</body>
</html>

```

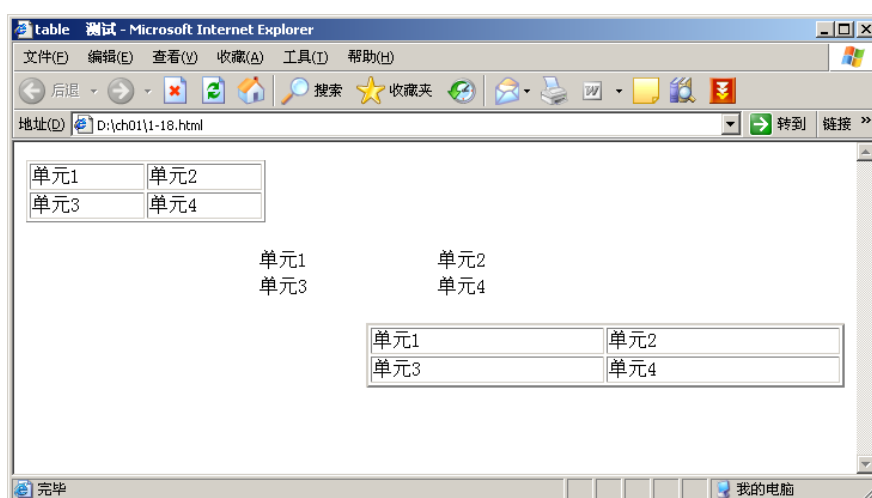


图 1.21 具有不同属性的表格

1.4.8 建立页面表单

使用标签<form>和</form>创建一个表单。<form>的主要属性是 action 和 method。

- ❑ action 属性：用来指定表单数据被提交后，处理这些数据的程序的地址。如下 HTML 代码，表示当表单提交后，表单的数据将被传到文件 login.php，由 login.php 来处理传入的数据。

```
<form action="login.php">
```

- ❑ method 属性：指定用何种 HTTP 方式传递数据。

有两种传递数据的方式：POST 方式和 GET 方式。POST 方式将表单数据放在 HTTP 数据的正文部分传递。GET 方式将表单数据加到 action 所指的地址之后传递。

说明：当表单被提交时，表单元素的 value 属性所对应的值将会被传送。对于文本框的 value 属性，其值就是用户输入的数据。

1.5 Web 编程的基础知识之二：层叠样式表 (CSS) 简介

层叠样式表的英文全称是 Cascading Style Sheet (简称 CSS)。HTML 最初设计时，只是用来定义文档的内容。比如标签<table>就是用来定义一个 HTML 文档的表格。页面的布局由浏览器显示，并不由 HTML 标签控制。

随着一个站点 HTML 文件的越来越多，如果用仍 HTML 标签排版和控制页面显示效果，它的局限性和困难性的问题会日益突出。甚至，在标签无法满足页面显示效果时，有的设计人员又加入 JavaScript 来控制页面效果。可以想象，这样 HTML 文档会变得越发臃肿，并且越来越难以维护。

层叠样式表（CSS）的出现解决了这个问题，即使用 CSS 决定网页内容如何显示，用 CSS 控制页面显示效果。

1.5.1 样式表的基本语法

一个样式(Style)的语法由 3 部分构成：Selector（选择器），属性(Property)，属性值(Value)。格式如下：

```
selector {property: value}
```

例如下面的例子，p 就是 selector，color 就是属性，blue 就是属性值。

```
p {color:blue}
```

其中 p 就是指 p 标签<p>，这个样式（Style）的含义表示凡是用<p>标签标记的文档内容，其文本颜色显示为蓝色。HTML 中所有的标签都可以作为 selector。

如果想为 Style 添加多个属性，可以在两个属性之间用分号分隔。下面的 Style 就包含两个属性，一个是对齐方式，其值为居中，一个字体颜色，其值为红色，它们之间用分号分隔开。

```
p {text-align:center;color:red}
```

1.5.2 设置页面字体格式

使用 CSS 可以设置字体的名称、大小、显示风格等样式。

1. 字体名称属性：font-family

用这个属性设定字体的名称，如 Arial, Tahoma, Courier 等。示例代码 1-19 演示了如何使用该属性。

代码 1-19 CSS 字体名称属性：font-family 1-19.html

```
<html>
<head>
<title>字体名称属性 font-family</title>
<STYLE>
.s1 {font-family:Arial}
.s2 {font-family:Courier}
.s3 {font-family:Verdana}
</STYLE>
</head>
<body>
<p class = "s1"> This text's fon-family value is "Arial"</p>
<p class = "s2"> This text's fon-family value is "Courer"</p>
<p class = "s3"> This text's fon-family value is "Verdana"</p>
</body>
</html>
```

显示效果如图 1.22 所示。

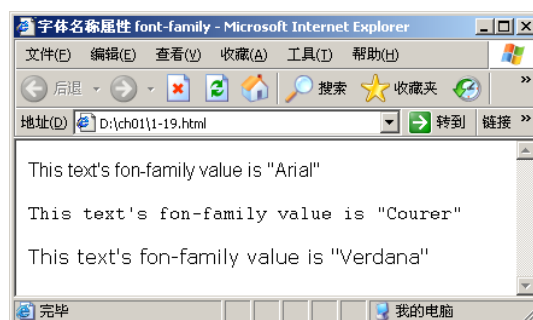


图 1.22 字体名称属性

2. 字体大小属性: font-size

font-size 属性用来设定字体的大小，字体大小的单位有多种，常见的是 pt 和 px。示例代码 1-20 演示了如何使用该属性。

代码 1-20 CSS 字体大小属性: font-size 1-20.html

```
<html>
<head>
<title>字体大小属性 font-size</title>
<STYLE>
body{font-size:10pt}
.s1 {font-size:16px}
.s2 {font-size:16pt}
</STYLE>
</head>
<body>
<p>此段文字的大小为 10pt。</p>
<p class = "s1">此段文字的大小是 16px</p>
<p class = "s2">此段文字的大小是 16pt</p>
</body>
</html>
```

显示效果如图 1.23 所示。



图 1.23 字体大小属性

3. 字体风格属性: font-style

这个属性有 3 个值可选: normal (正常显示)、italic (斜体显示)、oblique (斜体显示)。normal 是缺省值。示例代码 1-21 演示了如何使用该属性。

代码 1-21 CSS 字体风格属性: font-style 1-21.html

```
<html>
<head>
<title>字体风格属性 font-style</title>
<STYLE>
.s1 {font-style:italic}
.s2 {font-style:oblique}
</STYLE>
</head>
<body>
<p>此文字风格是 normal </p>
<p class = "s1">此段文字的字体风格属性值是 italic</p>
<p class = "s2">此段文字的字体风格属性值是 oblique</p>
</body>
</html>
```

显示效果如图 1.24 所示。



图 1.24 字体风格属性

1.5.3 设置页面颜色和背景

使用 CSS 背景颜色属性和背景图片属性，可以很方便地设置 Web 页面的颜色和背景。

1. 背景颜色属性: background-color

此属性为 HTML 元素设定背景颜色，示例代码如下。

```
body {background-color:#99FF00;}
```

上面的代码表示 body 这个 HTML 元素的背景颜色值是#99FF00。

2. 背景图片属性: background-image

此属性为 HTML 元素设定背景图片，示例代码 1-22 演示了如何使用背景图片属性。

代码 1-22 用 CSS 设置背景图片 1-22.html

```
<html>
<head><title>背景图片 background-image</title></head>
<body style="background-image:url(background.jpg)">
<p>这个 HTML 文档使用了 CSS 的 background-image 属性来设置 Web 页面的背景图<p>
</body>
</html>
```

显示效果如图 1.25 所示。



图 1.25 页面背景效果

1.5.4 处理页面的边距和填充

边距属性是用来设置页面中一个元素所占空间的边缘到相邻元素之间的距离。

❑ 左边距属性: `margin-left`: 这个属性用来设定左边距的宽度。示例代码如下。

```
.d1{margin-left:1cm}
```

❑ 右边距属性: `margin-right`: 这个属性用来设定右边距的宽度。示例代码如下。

```
.d1 {margin-right:1cm}
```

❑ 上边距属性: `margin-top`: 这个属性用来设定上边距的宽度。示例代码如下。

```
.d1 {margin-top:1cm}
```

❑ 下边距属性: `margin-bottom`: 这个属性用来设定下边距的宽度。示例代码如下。

```
.d1{margin-bottom:1cm}
```

❑ 边距属性: `margin`: 这个属性是设定边距宽度的一个快捷的综合写法, 用这个属性可以同时设定上下左右边距属性。还可以为上下左右边距设置相同的宽度。示例代码如下。

```
.d1 {margin:1cm}
```

也可以分别设置边距, 顺序是上, 右, 下, 左。示例代码如下。表示上边距为 1cm, 右边距为 2cm, 下边距为 3cm, 左边距为 4cm。

```
.d1 {margin:1cm 2cm 3cm 4cm}
```

1.5.5 理解 HTML 层的概念

HTML 中, 使用标签 `<div>` 和 `</div>` 来定义一个层, 通过 CSS 指定不同的属性值, 可以定位层, 从而实现页面的布局。

利用层可以非常灵活地放置内容, 例如可以将层前后放置、隐藏某些层而显示其他层、在屏幕上移动层等。可以在一个层中放置背景图像, 然后在该层的前面放置第二个层, 从而包含带有透明背景的文本。通常, 可以把层看成一个容器, 在层里可以放置其他更多的 HTML 元素。

1.5.6 按 Web 标准建立网页

传统 HTML 布局, 一般都是使用表格。表格定位比较简单快捷, 但容易出现表格嵌套表格的现象,

这样对页面的后期维护很不方便，而且表格越多，越会影响到浏览器解析 HTML 文档的速度，使页面打开的速度变慢。如今，页面设计人员更多地使用 HTML 层和 CSS，用于 Web 页面的布局。

所谓的 Web 标准，其实并没有统一的标准。不过，它似乎朝着使用<div>和 CSS 进行页面布局的方向行进。DIV+CSS 只是具体的实现技术手段，并不能涵盖 web 标准。web 标准不仅仅是布局的问题，更重要的是信息结构清晰、内容与表现相分离，而 DIV+CSS 技术能较好的实现这种思想。因此，当前看到的多数符合标准的页面都是采用 DIV+CSS 制作。

1.5.7 在网页中引入样式表

最常用的样式表引入方式有：内嵌样式（Inline Style）、内部样式表（Internal Style Sheet）和外部样式表（External Style Sheet）。

❑ 内嵌样式：内嵌样式是写在标签里面的。内嵌样式只对所在的标签有效。如下代码所示。

```
<p style="font-size:20pt; color:red">用 Style 定义字体</p>
```

这里在<p>标签内增加样式：style="font-size:20pt; color:red，这个样式只对当前<p>标签有效。

❑ 内部样式：内部样式表是写在 HTML 的<head></head>里面的。内部样式表只对所在的网页有效。如下面代码所示。

```
<HTML>
<HEAD>
<STYLE type="text/css">
H1.mylayout {border-width:1; border:solid; text-align:center; color:red}
</STYLE>
</HEAD>
<BODY>
<H1 class="mylayout"> 这个标题使用了 Style。</H1>
<H1>这个标题没有使用 Style。</H1>
</BODY>
</HTML>
```

❑ 外部样式：外部样式是指，将样式(Styles)写在一个以.css 为后缀的 CSS 文件里，然后在每个需要用到这些样式(Styles)的网页里引用这个 CSS 文件。如下面的 HTML 文档引入当前目录下的 CSS 文件：style.css。

```
<HTML>
<HEAD>
<link href="style.css" rel="stylesheet" type="text/css">
</HEAD>
<BODY>
<H1 class="mylayout"> 这个标题使用了 Style。</H1>
<H1>这个标题没有使用 Style。</H1>
</BODY>
</HTML>
```

1.6 Web 编程的基础知识之三：JavaScript 基础

JavaScript 是一种基于对象和事件驱动并具有安全性能的脚本语言，它是 Web 页面设计的重要组成部分，它被嵌入在 HTML 文档中，由浏览器解释执行。可以使用 JavaScript 设计出更有效果的 Web 页

面、验证 Web 页面表单数据、创建 cookie 等。

1.6.1 网页中的 JavaScript

JavaScript 可以出现在 HTML 文档的任何地方，但必须包含在 “<script language=“JavaScript”>” 和 “</script>” 之间。一般情况下，使用 “<script language=“JavaScript”>” 和 “</script>” 包含的 JavaScript 代码。代码 1-23 演示了在一个 HTML 文档中包含一段 JavaScript 代码。

代码 1-23 HTML 文档中的可执行的 JavaScript 代码 1-23.html

```
<html>
<head>
My first JavaScript!
</head>
<body>
<br>
This is a normal HTML document.<br/>
<script language="JavaScript">
document.write("这段文字是由 JavaScript 输出的!")
</script>
<br/>HTML Content
</body>
</html>
```

以上 JavaScript 代码就是可以执行 JavaScript 脚本。其中 document 是 JavaScript 对内置对象，读者可以将其理解为，该对象就代表当前的 HTML 文档。开发人员在 JavaScript 中可以直接使用该对象。write 是该对象的一个方法，它的作用是向 Web 页面输出一行文字，接下来的几个 JavaScript 示例程序将还会用到它们，读者目前只要知道存在这个对象并且可以使用它们即可，后面将会讲到这些内置对象及其用法。这段脚本通过 JavaScript 的 document 对象的 write 方法输出一段文字，显示结果如图 1.26 所示。

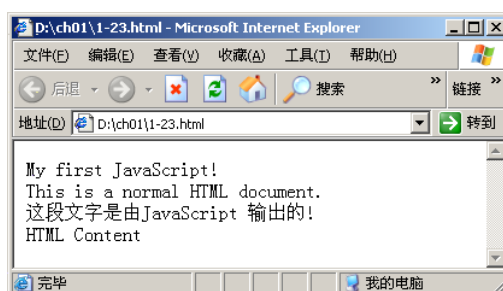


图 1.26 html 中可执行的 JavaScript

为便于初学者掌握 JavaScript，本书主要介绍在 HTML 文档中的 JavaScript 应用，及如何编写在 HTML 中的 JavaScript 代码。事实上，JavaScript 代码可以编写在独立的文件中，它们通常以后缀名 “.js” 保存，当 HTML 文档需要时，将这些 .js 文件包含到当前 HTML 文档中。

1.6.2 JavaScript 的变量

JavaScript 变量是用来存储数据的地方。这样在需要用这个值的地方就可以用变量来代表，一个变量可以是一个数字、文本等。JavaScript 变量的命名区分大小写，例如，名为 computer 的变量和名为

Computer 的是不一样的。而且，变量名必须以字母或下划线开头。

JavaScript 是一种对变量的数据类型要求不太严格的语言，所以不必声明每一个变量的类型，但在使用变量之前先进行声明是一种好习惯。在 JavaScript 中用 `var` 语句声明一个变量，象下面这样：

```
var name = value
```

其中 `name` 是变量名，`value` 是变量的值。一般也可以不加 “`var`” 来声明变量，象下面这样：

```
name = value
```

下面的代码为一个变量指定了一个值。

```
var str = "Learn JavaScript"
```

上面的声明了一个变量 `str`，并为它指定值 “Learn JavaScript”。上面的代码也可以写成如下形式。

```
str = "Learn JavaScript"
```

1.6.3 JavaScript 的基本语句

这一小节仅介绍 JavaScript 的几种常用的基本语句，它们是：

- ☐ `if...else` 条件语句。
- ☐ `switch` 选择语句。
- ☐ `for` 循环语句。
- ☐ `while` 循环语句。

1. `if` 和 `if...else` 条件语句

通常，JavaScript 脚本是按代码顺序，从第一句执行到最后一句。但有时，程序需要在某个条件成立的情况下执行一条语句，而在这个条件不成立的时，执行另外一条语句，这时就需要使用 `if` 或 `if...else` 语句。JavaScript 的 `if` 条件语句的语法如下所示。

```
if(condition)
{
    statement
}
```

这个语法的含义是，当条件 `condition` 为真时（即条件成立时），程序就执行语句 `statement`。注意，这里的 `if` 必须是小写字母，否则会出现 JavaScript 错误。代码 1-24 是一个 `if` 语句的例子。

代码 1-24 JavaScript 的 `if` 语句 1-24.html

```
<html>
<head>
<title>1-24</title>
</head>
<body>

<script language="JavaScript">
var a=3;
if(a>0)
document.write("a 的值是"+a)
</script>

</body>
</html>
```

上述代码定义一个变量 `a`，其值为 3，因为条件 `a>0` 成立，所以上述代码将在浏览器显示一行 “a 的

值是 3” 的文字。执行结果如图 1.27 所示。



图 1.27 JavaScript 的 if 语句

if...else 条件语句的语法如下所示。

```
if (condition)
{
    statement1
}
else
{
    statement2
}
```

这个语法的含义是，当条件 `condition` 为真时，程序就执行语句 `statement1`，否则，则执行语句 `statement2`。if...else 语句和 if 语句的使用方法一样，这里不再重复举例。

2. switch 语句

switch 语句的语法如下所示。

```
switch(n)
{
    case 1:
        statement_1
        break
    case 2:
        statement_2
        break
    ....
    default:
        statement_n
}
```

该语句中的花括号将整个 switch 语句包含在内，该语句首先计算表达式 `n` 的值，然后用 `n` 的值和 `case` 后的值比较，如果匹配，则执行该 `case` 后的语句，直到遇到 `break` 语句。如果所有的 `case` 值都不匹配 `n` 的值，则执行 `default` 对应的语句 `statement_n`。`break` 语句在这里用来中断 switch 语句，使 switch 语句不再继续执行下去。代码 1-25 是 switch 语句的一个示例程序。

代码 1-25 JavaScript 的 switch 语句 1-25.html

```
<html>
<head>
<title>1-25</title>
</head>
<body>
```

```

<script language="JavaScript">
var d=new Date()      //生成一个 Date 对象 d
theTime=d.getHours() //使用 Date 对象 d 的 getHours()方法获得当前时间的小时值,并把这个值赋给变量 theTime

switch(theTime)      //根据变量 theTime 的值,选择不同的执行语句
{
case 7:
    alert("Good Morning!");
    break;
case 15:
    alert("Good Afternoon!");
    break;
case 19:
    alert("Good Evening!");
    break;
case 23:
    alert("It's time to go to sleep!");
    break;
default:
    alert("Hello,My friend!");
}
</script>

</body>
</html>

```

上述代码首先通过 JavaScript 对 Date 对象的 getHours()函数获得当前时间的小时值,然后使用 switch 语句根据不同的小时值,通过 JavaScriptalert()函数弹出 Web 页面对话框,在该对话框内显示一句话。执行效果如图 1.28 所示。

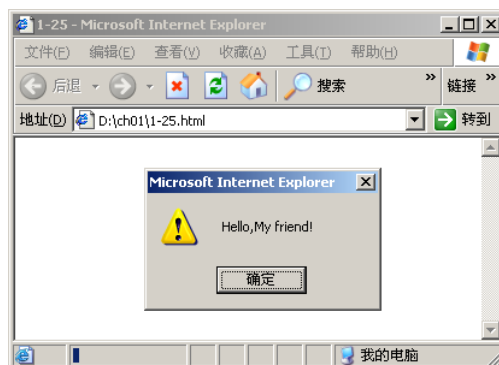


图 1.28 JavaScript 中的 switch 语句

3. for 循环语句

JavaScript 程序有时需要根据某些条件成立,反复执行某些语句。可以通过 for 循环语句实现这个功能。for 循环语句的语法如下所示。

```

for (var=变量初始值;表达式;变量 var 更新语句)
{
    statement
}

```

该语句的含义是:首先对变量赋初值,然后判断表达式是否成立,如果成立,则执行语句 statement,

然后执行变量 `var` 的更新语句，更新语句就是对变量 `var` 的值加 1。更新之后，再判断表达式的值，接着程序按上面的步骤继续执行。如果表达式不成立，那么程序跳出 `for` 循环，不再执行语句 `statement`。代码 1-26 是 `for` 循环语句的示例程序，它通过 `for` 循环，在 Web 页面上输出数字 0 到 9。

代码 1-26 JavaScript 的 `for` 循环语句 1-26.html

```
<html>
<head>
<title>1-26</title>
</head>
<body>
<script language="JavaScript">
var i=0
for (i=0;i<10;i++)
{
document.write("The number is " + i)
document.write("<br />")
}
</script>
</body>
</html>
```

其执行结果如图 1.29 所示。

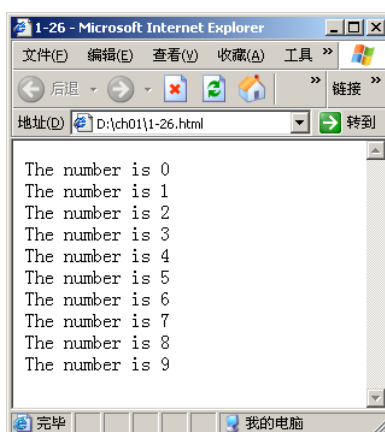


图 1.29 JavaScript 的 `for` 循环语句

4. `while` 循环语句

除了 `for` 循环语句，JavaScript 还有另外一种循环语句——`while` 循环语句，可以使某些语句在满足某些条件的时候，循环执行。`while` 循环语句的语法如下所示。

```
while (expr)
{
    statement
}
```

该语句的含义是，当表达式 `expr` 成立时，则执行语句 `statement`，执行完语句 `statement` 后，程序返回继续判断表达式 `expr` 的值，如此反复执行。为了让循环不会无限次的执行下去，需要有使表达式 `expr` 不成立的情况。代码 1-27 通过 `while` 循环语句，在 Web 页面上显示数字 0 到 9。

代码 1-27 JavaScript 的 while 循环语句

```
<html>
<head>
<title>1-27</title>
</head>
<body>

<script language="JavaScript">
var i=0
while (i<10)
{
document.write("The number is " + i)
document.write("<br />")
i=i+1
}
</script>

</body>
</html>
```

提示：上述代码的执行结果和图 1.29 所示完全一样。

1.6.4 使用简单的对话框

通常会使用 JavaScript 在网页上产生弹出对话框的效果。用 JavaScript 可以创建的对话框主要有以下所示的两种。

- ☐ 警告对话框。
- ☐ 确认对话框。

下面将通过实例代码，来了解如何使用 JavaScript 创建这些对话框。使用函数 `alert()` 创建警告对话框，它的语法格式如下所示。

```
alert("message")
```

其中 `message` 是显示在对话框上的警告或说明信息，警告对话框带有一个“确定”按钮。代码 1-28 演示的效果是，当用户浏览该 HTML 文档时，立刻弹出一个警告对话框。该代码执行效果如图 1.30 所示。

代码 1-28 使用 JavaScript 创建警告对话框 1-28.html

```
<html>
<head>
<title>1-28</title>
</head>
<body>

<script language="JavaScript">
alert("这是一个警告对话框！")
</script>

</body>
```

```
</html>
```

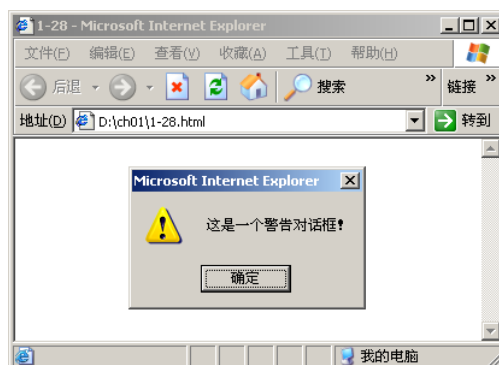


图 1.30 使用 JavaScript 创建警告对话框

使用函数 `confirm()` 创建一个确认对话框，它的语法格式如下所示。

```
confirm("message")
```

其中 `message` 是显示在对话框上的确认信息，确认对话框有两个按钮，分别是“确定”按钮和“取消”按钮。代码 1-29 演示的效果是，当用户浏览该 HTML 文档时，立刻弹出一个确认对话框。该代码执行效果如图 1.31 所示。

代码 1-29 使用 JavaScript 创建确认对话框 1-29.html

```
<html>
<head>
<title>1-29</title>
</head>
<body>

<script language="JavaScript">
confirm("这是一个确认对话框，它有两个按钮")
</script>

</body>
</html>
```



图 1.31 使用 JavaScript 创建确认对话框

1.6.5 JavaScript 的基本事件

所谓事件，是指用户与 Web 页面交互时产生的各种操作，例如单击 Web 页面上的超链接或按钮时，就产生一个单击（click）事件（event）。浏览器为了响应某个事件而进行的处理过程，叫做事件处理。

注意：浏览器在程序运行的大部分时间都等待事件的发生，事件能够被 Web 页面中的 JavaScript 捕捉到，并在事件发生时，自动调用 JavaScript 的事件处理函数，完成事件处理。

除此之外，浏览器自己的一些动作也可以产生事件，例如，当浏览器载入一个 Web 页面时，就会发生载入（load）事件，卸载一个页面时，就会发生卸载（unload）事件。本小节将介绍几种常见的 JavaScript 事件，它们是以下所示的 3 种。

- ❑ 鼠标单击事件（onClick）：当用户单击 Web 页面上的超链接或者按钮时，就会触发 onClick 事件。
- ❑ 页面载入和卸载事件（onload 和 onUnload）：onload 事件发生在 Web 页面完全下载完成后。onUnload 事件发生在用户离开（即关闭）当前 Web 页面时。onload 事件通常用来检测浏览器的一些信息，如浏览器类型、版本等。它们也可以用来处理用户 cookie。
- ❑ 提交事件（onSubmit）：该事件发生在页面表单的“提交”按钮按下并放开之后。通常使用该事件的处理过程验证表单数据的有效性。

因为事件发生后通常需要被处理，而处理的过程要在 JavaScript 函数中完成，所以关于事件应用的示例将在下小节介绍函数时，和函数的应用一同来举例。

1.6.6 JavaScript 的函数

函数是一个可以完成特定功能可执行的代码块，它由一条或多条 JavaScript 语句组成。通常将那些会反复使用，或者用来完成某种功能的代码写成函数，以便利于代码的重用。JavaScript 中函数的语法如下所示。

```
function name(var_1,var_2,...,var_n)
{
    statement
}
```

JavaScript 的函数定义由关键字“function”开始，这意味着，每个函数的定义前，都要有“function”一词。关键字“function”之后的 name 是函数名，它是必须的。接着后面括号里的“var_1,var_2...var_n”是函数的参数列表，表示传入函数里的、需要被函数处理的值。statement 是函数体，即调用函数时，函数要执行的语句。它可以是一条语句，也可以是多条语句。如果一个函数不需要传入参数，那么函数名后面的括号不能没有，仍然紧跟在函数名后面，如下面代码所示。

```
function name()
{
    statement
}
```

提示：Web 页面中的 JavaScript 函数通常可以写在<head>元素或者<body>元素内。

JavaScript 中的函数，通常由于某个事件被触发而调用执行。例如，代码 1-30 演示了当一个 Web 页面按钮按下时，触发 onclick 事件，该事件调用一个函数，该函数会弹出一个对话框，表示函数调用成功。

代码 1-30 onclick 事件及其处理函数的调用 1-30.html

```
<html>
<head>
<title>1-30</title>
<script language="JavaScript">
    function show_msg()
    {
        alert("onclick 事件成功调用了函数 show_msg! ")
    }
</script>
</head>

<body>
<form>
<input type="button" value="单击这里" onclick="show_msg()" >
</form>

</body>
</html>
```

上述代码在<head>和</head>标签中定义 JavaScript 函数 show_msg，该函数的功能很简单，即在其被调用时，弹出一个警告对话框，并显示一些信息。接着在 HTML 文档中，指定一个按钮，为其添加 onclick 事件处理函数 show_msg。代码“onclick=show_msg()”表示，当发生 onclick 事件时，调用其处理函数 show_msg。用浏览器打开上述 HTML 文档，单击页面上的“单击这里”按钮，将看到如图 1.32 所示的效果。



图 1.32 onclick 事件及其处理函数的调用

1.6.7 JavaScript 的对象

JavaScript 是一种面向对象的编程语言。一段文字、一个图片、一个表单都可以看作是一个对象，每个对象都有自己的属性、方法和事件。属性表示了该对象的某些特征，如字符串的长度、文本框里的文字等。方法可以理解成该对象可以处理的一些事情。

JavaScript 提供了很多对象，这些对象可以直接在 JavaScript 里使用。本小节介绍 JavaScript 的一些常用的基本对象。

1. String 对象

String 对象代表一个字符串，定义该对象的最简单办法就是直接赋值。如下代码所示。此时，变量 str 就是一个 String 对象，它的值是“Hello JavaScript”。

```
var str = "Hello JavaScript"
```

String 对象的一个重要属性是 length，该属性返回 String 对象的长度，即字符串包含的字符数。如下代码将在 Web 页面上显示数字 16，因为 String 对象 str 包含 16 个字符。

```
document.write(str.length)
```

下面介绍 String 对象几个常见的方法。

❑ substring(): 返回字符串的子字符串。示例代码如下所示。代码中 str.substring(0,5)表示从 str 中位置为 0 的字符（即第一个字符）开始，取 5 个字符。所以，变量 str_sub 的值是“Hello”。

```
str_sub = str.substring(0,5)
```

❑ indexOf(): 该方法查找一个 String 对象中另外一个字符串对象的起始位置，如果找到，则返回其位置，否则返回-1。

❑ toLowerCase(): 该方法将 String 对象中的所有大写字母都变成小写字母。

❑ toUpperCase(): 该方法将 String 对象中的所有小写字母都变成大写字母。

2. 日期对象——Date

顾名思义，日期对象就是用来处理时间的 JavaScript 对象。要定义一个日期对象，需要使用 new 运算符，如下代码所示。

```
var some_date = new Date()
```

上面的代码通过 new 运算符定义了一个日期对象 some_date，它的初始值就是当前系统的日期和时间。如果要指定初始值，可以使用类似下面的代码。

```
var some_date = new Date(2007,10,01)
```

上述代码指定日期对象 some_date 的值为 2007 年 10 月 1 日。因为，如果不指定时区，Date 对象都采用 UTC（世界时）表示其值，所以上面的变量 some_date 的值看起来是：Thu Nov 1 00:00:00 UTC+0800 2007。

Date 对象有以下常用的方法。

❑ setYear(): 设置 Date 对象的年份，如果使用两位数，该方法会自动为其前面加上“19”。如果要设置年份为 2007 年，则需要写全 4 位数。

❑ getYear(): 取得 Date 对象的年份。

❑ setMonth(): 设置 Date 对象的月份。

❑ getMonth(): 取得 Date 对象的月份。

❑ getDay(): 取得 Date 对象的星期，0 表示星期天。

❑ setHours(): 设置 Date 对象小时数，采用 24 小时制。

❑ getHours(): 取得 Date 对象的小时数，采用 24 小时制。

❑ setMinutes(): 设置 Date 对象的分钟数。

❑ getMinutes(): 取得 Date 对象的分钟数。

❑ setSeconds(): 设置 Date 对象的秒数。

❑ getSeconds(): 取得 Date 对象的秒数。

❑ toLocalString(): 返回一个有本地时间格式指定的 Date 对象所指定日期。

代码 1-31 演示了其中几个方法的使用法。

代码 1-31 Date 对象方法的示例代码 1-31.html

```
<html>
```



```

<head>
<title>1-31</title>
</head>
<body>

<script language="JavaScript">
var some_date = new Date()
document.write(some_date.getDay()+"<br/>")
document.write(some_date+"<br/>")
document.write(some_date.toLocaleString())
</script>

</body>
</html>

```

用浏览器打开该 HTML 文档，会看到类似图 1.33 的执行结果。

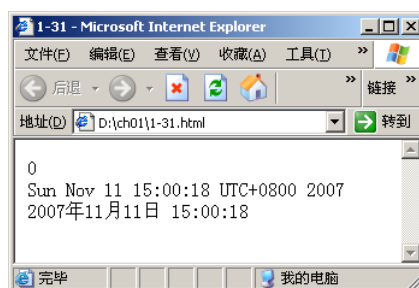


图 1.33 Date 对象的方法执行结果

3. 数组对象——Array

数组对象用来存放一组数据（或者说是对象），这些数据可以是不同类型的。数组中的每个对象都有一个“下标”，用来表示它在数组中的位置。数组的下标从 0 开始算起，所以，第一个对象的位置是 0，第二个对象的位置是 1，以此类推。JavaScript 中定义一个数组的方法如下代码所示。

```
var my_arr = new Array()
```

上面的代码通过 new 运算符定义了一个不包含任何数据的数组 my_arr。定义完数组后，可以为其指定值，如下代码所示。

```

var myFavorite = new Array()
myFavorite[0]="mango"
myFavorite[1]="banana"
myFavorite[2]="apple"

```

上面的代码为数组 myFavorite 添加了 3 个值。也可以在定义数组的同时指定数组的大小，如下代码所示。

```

var myFavorite = new Array(3)
myFavorite[0]="mango"
myFavorite[1]="banana"
myFavorite[2]="apple"

```

还有一种为数组添加值的方式，即在定义数组的同时，给数组添加值，如下代码所示。

```
var myFavorite = new Array("mango","banana","apple")
```

可以通过数组下标来访问数组中的某个值，如下代码将在 Web 页面上显示“apple”。

```
document.write(myFavorite[2])
```

Array 对象的一个重要属性是：length。它返回数组的长度，即数组中元素的个数。

Array 对象的常用方法有以下两种

- ❑ **reverse()**: 该方法使数组中的元素顺序颠倒过来。如有一个数组的元素顺序是 1、2、3，则该数组使用该方法后，数组元素的顺序变为 3、2、1。
- ❑ **join()**: 该方法将数组中的各元素连接起来，各元素之间使用某种分割符作为间隔，分割符以参数形式传给该方法。该方法不会影响到数组本身的内容。

代码 1-32 演示了如何使用 join 方法。

代码 1-32 Array 对象的 join 方法 1-32.html

```
<html>
<head>
<title>1-32</title>
</head>
<body>

<script language="JavaScript">
var myFavorite = new Array(3)
myFavorite[0]="mango"
myFavorite[1]="banana"
myFavorite[2]="apple"

var str = myFavorite.join(",")
document.write(str)
</script>

</body>
</html>
```

上述代码中，数组对象 myFavorite 调用其 join 方法，向其传入参数 “,”，表示用 “,” 连接数组中的各个元素。用浏览器打开该 HTML 文档，会看到如图 1.34 所示的执行效果。



图 1.34 Array 对象的 join() 方法的使用

4. window 对象

该对象表示的是一个浏览器窗口。引用该方法或属性时，不需要使用 “window.xxx” 这种形式，而是直接使用该方法或属性。该对象常见的属性如下所示。

- ❑ **name**: 当前窗口的名称。
- ❑ **states**: 表示浏览器窗口下方状态栏所显示的内容。通过对该属性发赋值，可以改变浏览器状态栏显示的内容。
- ❑ **self**: 指浏览器窗口本身。
- ❑ **history**: 浏览器历史访问对象。

❑ location: 浏览器地址对象。

❑ document: 文档对象。

window 对象常见的方法如下所示。

❑ open(): 打开一个窗口。

❑ close(): 关闭一个已打开的窗口。

❑ alert(): 弹出一个包含“确定”和“取消”按钮的对话框。该方法已经在前面介绍过。

5. document 对象

document 对象代表浏览器窗口内的文档。该对象包含了整个 HTML 文档，并且可以访问当前页面中的所有元素。该对象最常用的方法就是 write，它向当前 HTML 文档写入数据。前面的内容已多次使用过该方法，这里不再重复举例。document 有一个重要属性：cookie。接下来的一小节，就向读者介绍 JavaScript 中的 cookie。

1.6.8 JavaScript 中的 cookie

cookie 是指当用户访问某一个 Web 站点时，由服务器存储在客户端计算机中的一些变量。它通常用来区别不同的访问用户。当同一台计算机请求访问某个页面时，浏览器也会将 cookie 发送给服务器。

cookie 一般是这样的形式：cookie 名称=cookie 值。cookie 的名字一般使用字母和数字命名，cookie 的值要求是可以用 URL 编码的字符。所有的 cookie 都有一个失效期，过了失效期，计算机就会将这个 cookie 删除。JavaScript 中通常使用 document 的 cookie 属性存储 cookie。它的用法如下所示。

```
document.cookie='cookieName='+escape('cookieValue')+';expires=' + expirationDateObj.toGMTString();
```

其实上面的代码就是为 cookie 属性赋值，所赋的值是一个字符串，该字符串包含了 cookie 的名字和值，以及 cookie 对过期时间。下面通过一个具体的实例，来了解 JavaScript 中 cookie 的使用。

(1) 创建一个存储 cookie 的函数，该函数功能是保存 cookie 的值和过期时间。代码如下所示。

```
function setCookie(name,value)    //该函数有两个参数，一个是 cookie 的名字，一个是 cookie 的值
{
    var days = 30;                //cookie 的有效期为 30 天
    var exdate = new Date();
    exdate.setDate(exdate.getDate() + days);
    document.cookie = name + "=" + escape(value) + ";expires=" + exdate.toGMTString();
}
```

在上面的代码中，首先取得有效日期，然后加上 cookie 有效期的天数。之后，在 document 的 cookie 属性中存储 cookie 的名字、值和过期日期。

(2) 创建一个函数，通过该函数获取被设置的 cookie 值。代码如下所示。

```
function getCookie(name)
{
    //判断 cookie 属性的长度是否大于 0，大于 0 说明设置了 cookie
    if(document.cookie.length>0)
    {
        start=document.cookie.indexOf(name + "=");    //找到设置 cookie 名称的位置
        if(start!=-1)
        {
            start=start + name.length+1;
            end=document.cookie.indexOf(";",start);
            if(end==-.1)
```

```

        end=document.cookie.length;
        return unescape(document.cookie.substring(start,end));
    }
}
return "";
}

```

该函数首先通过 cookie 的 length 属性判断 cookie 是否有值，如果有值，则找到该 cookie 的值，然后返回该值。否则，返回空字符串。

(3) 创建一个函数，如果 cookie 被设置了，那么该函数弹出一个提示对话框，否则弹出一个输入框，让用户输入名称。该函数代码如下：

```

function checkCookie()
{
    user_name=getCookie('user_name');
    if(user_name!=null && user_name!="")
    {
        alert('欢迎您, '+user_name+'! ');
    }
    else
    {
        user_name=prompt('请输入您的名字:', "");
        if(user_name!=null && user_name!="")
        {
            setCookie('user_name',user_name);
        }
    }
}

```

该函数首先通过函数 getCookie 取 cookie 的值，如果取到 cookie 的值，则弹出一个对话框，显示已段文字信息。否则，使用方法 prompt 弹出“输入提示对话框”，等待用户输入名字，然后将这个名字作为 cookie 的值，通过 setCookie 函数进行保存。

(4) 以上示例的完整代码如 1-33 所示。

代码 1-33 JavaScript 中 cookie 的使用 1-33.html

```

<html>
<head>
<title>1-33</title>
<script language="JavaScript">
function setCookie(name,value)    //该函数有两个参数，一个是 cookie 的名字，一个是 cookie 的值
{
    var days = 30;                //cookie 的有效期为 30 天
    var exdate = new Date();
    exdate.setDate(exdate.getDate() + days);
    document.cookie = name + "=" + escape(value) + ";expires=" + exdate.toGMTString();
    alert(document.cookie);
}

function getCookie(name)
{
    //判断 cookie 属性的长度是否大于 0，大于 0 说明设置了 cookie

```

```
if(document.cookie.length>0)
{
    start = document.cookie.indexOf(name + "=");    //找到设置 cookie 名称的位置
    if(start!=-1)
    {
        start = start + name.length+1;
        end=document.cookie.indexOf(";",start);
        if(end==-1)
            end=document.cookie.length;
        return unescape(document.cookie.substring(start,end));
    }
}
return "";
}

function checkCookie()
{
    user_name=getCookie('user_name');
    if(user_name!=null && user_name!="")
    {
        alert('欢迎您, '+user_name+'! ');
    }
    else
    {
        user_name=prompt('请输入您的名字:', "");
        if(user_name!=null && user_name!="")
        {
            setCookie('user_name',user_name);
        }
    }
}

</script>
</head>

<body onLoad="checkCookie()">
</body>
</html>
```

(5) 当第一次用浏览器打开该文档时, 将看到如图 1.35 所示的效果。因为此时没有设置过 cookie, 所以弹出对话框让用户输入名字。

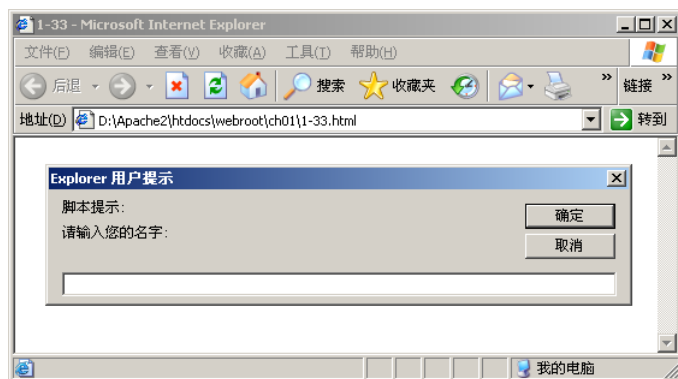


图 1.35 第一次请求

(6) 在此对话框中输入名字，单击确定按钮，如图 1.36 所示。注意，之后再刷新该页面，可以看到如图 1.37 所示的执行效果。因为此时已经设置过 cookie，所以会弹出信息对话框。而对话框中有用户刚刚输入的名字，这个名字是从 cookie 中取得的。

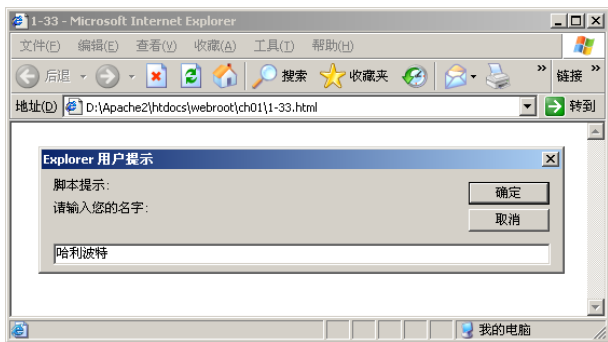


图 1.36 输入名字

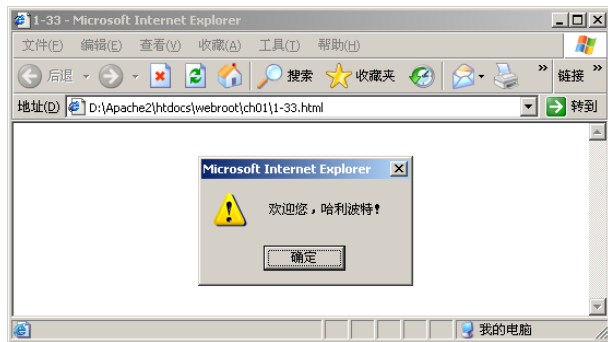


图 1.37 保存 Cookie

1.7 开始 PHP 之旅：搭建 PHP 开发环境

从这节开始，正式进入和 PHP 有关的内容。本节将介绍 Apache、MySQL 和 PHP 的下载、安装及配置，分别讲述在 WindowsXP 和 Linux/UNIX 下开发环境的搭建。为了便于初学者理解和掌握，本书将按 Windows 操作系统下的开发环境讲述 PHP 语言及其编程。

1.7.1 下载 Apache、MySQL 和 PHP

Apache、MySQL 和 PHP 都是开源产品，均可从其官方网站下载，并且可以免费使用。

1. Apache 的下载

Apache 服务器的官方网站是 <http://httpd.apache.org>。当前，可以通过镜像地址 <http://apache.mirror.phpchina.com/httpd/binaries/win32/> 来下载 Windows 的 Apache 安装程序，这个镜像地址列有各个版本的 Apache 安装程序，如图 1.27 所示。在这个页面中单击合适的下载链接，下载 Apache 安装软件。

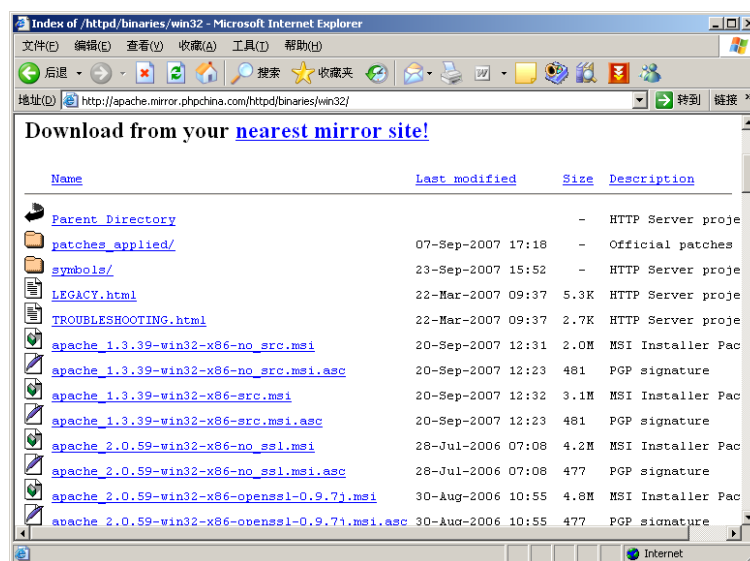


图 1.27 Apache 下载页面

本书以 Apache2.0.52 的 Window 版本作为例子，在官网上下载 `apache_2.0.52-win32-x86-no_ssl.msi` 安装程序。

2. MySQL 的下载

MySQL 的官方网站是 `http://www.mysql.com`。可以通过 `http://dev.mysql.com/downloads/mysql/5.0.html` 下载最新版本的 MySQL，如图 1.28 所示。对于 Windows 建议下载安装版本的可执行文件（即 exe 文件），本书以安装 MySQL5.0 作为例子。

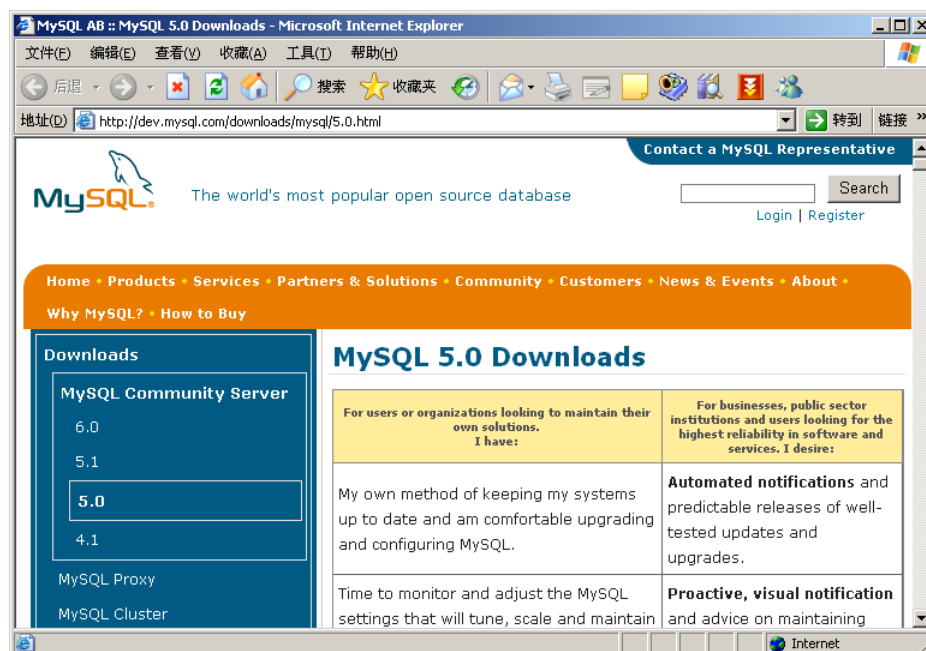


图 1.28 MySQL 下载页面

3. PHP 的下载

PHP 的官方网站是 `http://www.php.net`。可以通过 `http://www.php.net/downloads.php` 下载最新的 PHP

版本。对于 Windows 系统，可以下载安装版本或者 PHP 压缩包文件，如图 1.29 所示。笔者建议下载 PHP 的 Windows 压缩包（.zip 文件）来安装 PHP。本书以安装 PHP5.1 作为例子。



图 1.29 PHP 下载页面

1.7.2 在 Windows 下配置开发环境

下载完需要的软件后，本节介绍如何实现这些软件的安装和配置。

1. Apache 的安装

Windows 下的 Apache 安装，和其他 Windows 软件安装类似，只需按步骤单击“下一步”就可以了。这里主要说明安装过程中，Server Information 对话框的填写，如图 1.30 所示。

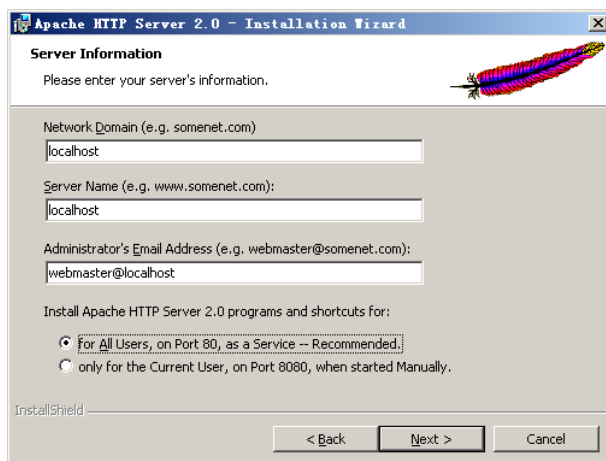



图 1.30 Apache 安装

因为这里只是搭建一个学习用的环境，并不需要一个真正意义上的网络域名，所以这里“Network

Domain”项填写 localhost 即可。同理，Server Name 项也填写 localhost，Administrator’s Email Address 项可填写 webmaster@localhost 或其他 Email 地址。其他选项不用改变，单击下一步完成后续的安装步骤。本书选择将 Apache 安装在 D 盘根目录下。

安装完成后，Apache 服务器就会试着自动启动。如果启动成功，会在 Windows 的托盘区域看到 Apache 正在运行的图标（一个带有紫红色羽毛的绿色箭头），如下所示。

这个图标表示 Apache 已经启动



2. PHP 的安装

PHP 下载完成后，将压缩包解压至想要安装 PHP 的目录，即基本完成了 PHP 的安装。本书的后续讲述，将默认 PHP 是安装在 D 盘根目录下。读者可以选择想要安装 PHP 的目录，只是在后续的配置中，需要将相关配置换成读者自己的 PHP 安装目录。

3. MySQL 的安装

像安装其他 Windows 软件一样，单击 MySQL 的安装程序，按步骤单击“下一步”完成安装。MySQL 软件可以安装在 Windows 任何一个盘符下，但笔者强烈建议初学者将 MySQL 安装在 C 盘根目录下，这样不需做任何设置，MySQL 就可以顺利启动并运行。本书以 MySQL 安装在 C 盘根目录下作为例子。

4. 配置开发环境

为了让 Windows 可以执行 PHP，需要做进一步的工作。读者可先按以下步骤搭建好开发环境，至于这样操作步骤的原因与意义，将在后面详细讲述。

（1）在 PHP 安装目录，找到文件 **php5ts.dll**，对于 WindowsXP 用户将其拷贝到 C:\WINDOWS\system32 目录下，对于 Windows2000 用户将其拷贝到 C:\WINNT\system32 目录下。找到文件

（2）仍然在 PHP 安装目录下，找到文件 **libmysql.dll**，对于 WindowsXP 用户将其拷贝到 C:\WINDOWS\system32 目录下，对于 Windows2000 用户将其拷贝到 C:\WINNT\system32 目录下。

（3）接下来，转到 PHP 的安装目录，找到文件 **php.ini-recommended**，对于 WindowsXP 用户将其拷贝到 C:\WINDOWS 下，对于 Windows2000 用户将其拷贝到 C:\WINNT，然后改名为 **php.ini**。这个文件是配置 PHP 的基本文件，如设置服务器根目录等配置。用文本编辑器打开 **php.ini**，找到“**doc_root=**”，因为本书默认 Apache 安装在 D 盘根目录下，所以，修改这个配置项的值如下所示。

```
doc_root = d:\Apache2\htdocs
```

提示：用户需要根据自己安装 Apache 的路径，修改该配置项的值。

（4）在 **php.ini** 中找到“**extension_dir = \.**”，修改该配置项如下所示。这个配置项用来载入 PHP 的扩展模块。

```
extension_dir = d:\php\ext
```

（5）还是在 **php.ini** 中找到“**;extension=php_mysql.dll**”，把这项前面的分号去掉，改为如下所示的内容。

```
extension=php_mysql.dll
```

这个配置项用来使 PHP 支持 MySQL。这里的分号起注释作用，去掉分号，表示使这个项生效。修改完 PHP 的配置文件 **php.ini** 之后，保存退出该文件。

（6）接下来进行 Apache 的配置。转到 Apache 的安装目录下的 **conf** 目录，用文本编辑器打开 Apache 的配置文件 **http.conf**。在此文件中，找到“**Dynamic Shared Object (DSO) Support**”，在 **LoadModule** 行的最后添加如下所示的配置项。

```
LoadModule php5_module d:/php/php5apache2.dll
```

这项是通过 Apache 来载入 PHP 模块 **php5apache2.dll**，即 PHP 以 Apache 的一个模块来运行。因为本书将 PHP 安装在 D 盘根目录下，所以这里载入位置是 **d:/php/php5apache2.dll**。如果读者没有按本书默认的安装目录安装 PHP，那么读者需要按自己的安装目录修改 PHP 的载入位置。

(7) 为了让 Apache 支持 .php 后缀的文件，在 http.conf 中找到 “**AddType**” 项，在最后添加如下所示配置项

```
AddType application/x-httpd-php .php
```

找到 **DirectoryIndex**，在最后添加 **index.php**。为了支持中文，找到 **AddDefaultCharset ISO-8859-1** 改为：**AddDefaultCharset GB2312**。保存修改后的 http.conf，即完成 Apache 的基本配置。

(8) 至此，在 Windows 下的 PHP 开发环境就基本搭建好了。

注意：强烈建议对 Apache 配置文件 httpd.conf 文件的修改，一定遵守修改一点测试一点的原则，即每次修改完并保存 httpd.conf 后，要重新启动一下 Apache，如果 Apache 能正常启动，则可以进行后续的修改，如果 Apache 启动时报错，则说明这次的修改有问题，需要仔细检查并重新修改。

1.7.3 在 Linux/UNIX 下配置开发环境

因为在 Linux/UNIX 下配置 PHP 的开发环境，对读者的技术要求稍微高一些，而且涉及到一些 Linux/UNIX 方面的基础知识，如命令的使用、程序编译等，考虑到便于初学者学习和掌握，本节讲述比较简略。另外，本书主要以 Windows 为平台讲述 PHP 语言及其开发，因此初学者也可以跳过本节的学习。

1. MySQL 的安装

本书以在 Linux 下通过 .rpm 文件安装 MySQL 为例。在 MySQL 的官方网站下载 MySQL 的安装文件 MySQL-server-5.0.45-0.i386.rpm，执行下面的命令完成 MySQL 的安装。

```
rpm -ivh MySQL-server-4.0.20-0.i386.rpm
```

在没有设置 MySQL 密码的情况下，可以通过以下命令，测试 MySQL 是否成功安装。

```
mysql -u root
```

如果出现类似 “Welcome to the MySQL monitor. Commands end with ; or \g.” 这样的文字，说明 MySQL 安装成功了。

2. 安装 Apache

从 Apache 官方网站下载 UNIX 版本的压缩包，本书以 httpd-2.0.52.tar.gz 为例。假设资源包放在目录 /usr/local/src 下，进入这个目录后解压缩，命令如下所示。

```
cd /usr/local/src
```

```
tar -zxvf httpd-2.0.52.tar.gz
```

解压缩后，进入目录 httpd-2.0.52，执行以下命令：

```
./configure --prefix=/usr/local/apache2 --enable-module=so
```

其中 --prefix=/usr/local/apache2 用来指定 Apache 的安装目录。接下来进行编译，执行以下命令完成安装。

```
make
```

```
make install
```

3. PHP 的安装

(1) 从 PHP 官方网站上下载 PHP5.1.4 的压缩包，解压缩该文件包，命令如下所示。

```
tar -zxf php5.1.4.tar.gz
```

(2) 然后，转到解压缩后的目录，执行以下所示的命令。

```
./configure --prefix=/usr/local/php -with-mysql=/var/lib/mysql
```

(3) 以上命令完成 PHP 安装目录等相关配置。接着编译 PHP，命令如下所示。

```
make
make install
```

(4) 最后还需要拷贝当前目录下的 php.ini-dist 文件 PHP 的到安装目录的 lib 目录下，并改名为 php.ini，命令如下所示。

```
cp php.ini-dist /usr/local/php/lib/php.ini
```

(5) 以上安装完成后，可以参见 1.7.2 的内容进行有关配置，这里的配置项和配置方法和 1.7.2 所述完全类似。

1.7.4 善其事利其器——PHP 编辑器的选择

毋庸置疑，开发人员需要一个功能强大的 IDE (Integrated Development Enviroment, 集成开发环境)，就好似一个好的猎人，需要一支好的猎枪一样。使用 PHP 做 Web 应用的开发，无论是从开发人员的需求上考虑，还是项目的开发效率上考虑，都需要有一个功能强大而且易用的编辑器做支持。如今有许多编辑器可供 PHP 开发人员选择，它们各有优点，开发人员可以根据自己的需求、使用习惯等方面选择这些编辑器，将它们作为编写代码、开发程序的高效工具。

本节将简单介绍三款常用也是比较受开发人员欢迎的 PHP 编辑器，希望对初学者有所帮助。这三款编辑器是：UltraEdit、Eclipse 和 Zend Studio。

1. UltraEdit 编辑器

UltraEdit 是个有十多年历史的老牌编辑器，它支持 HTML、C/C++、Java、Perl、PHP 等多种语言语法的高亮显示，支持 FTP 获取和保存文件、文件比较、二进制文件修改、宏等高级功能。最主要的是，它使用起来很简单，而且功能不弱，对 PHP 初学者来说，用它来编辑 PHP 程序是相当不错的选择。不过，UltraEdit 是共享软件，并不免费。图 1.31 是 UltraEdit 的编辑界面。

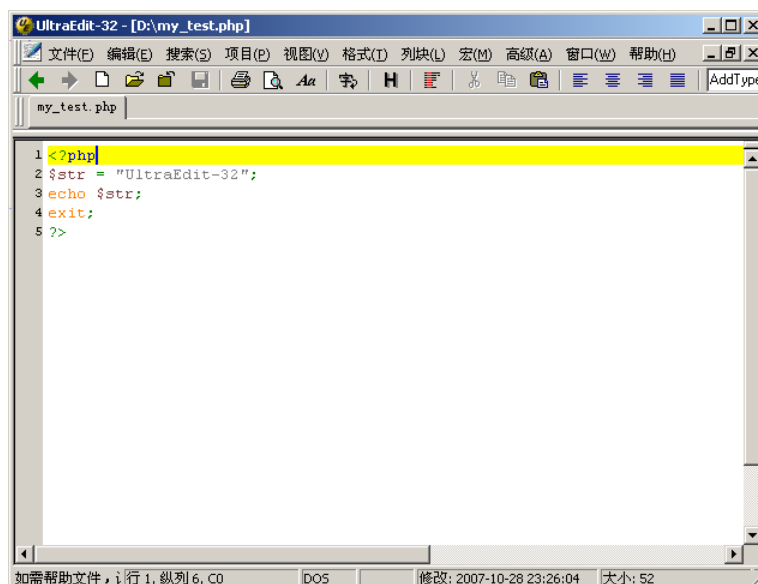


图 1.31 UltraEdit 的编辑界面

2. Eclipse 编辑器

Eclipse 是一个开源的、基于 Java 的、可扩展的开发平台，Eclipse 通过插件组件构建开发环境。Eclipse

不只是用来做 Java IDE，通过一系列的插件，Eclipse 可以支持更多的语言，如 C/C++、PHP 等。

安装了支持编辑 PHP 的插件以后，Eclipse 就可成了一个 PHP 的 IDE，而且它还集成了管理 MySQL 和 Apache 的工具。与 UltraEdit 不同的时，Eclipse 编辑 PHP 时，可以做语法检查、代码补全、括号自动匹配等功能，这些功能让开发人员编辑 PHP 代码时感觉更加得心应手。Eclipse 更适合开发大型 PHP 项目时使用。图 1.32 是 Eclipse 的编辑界面。

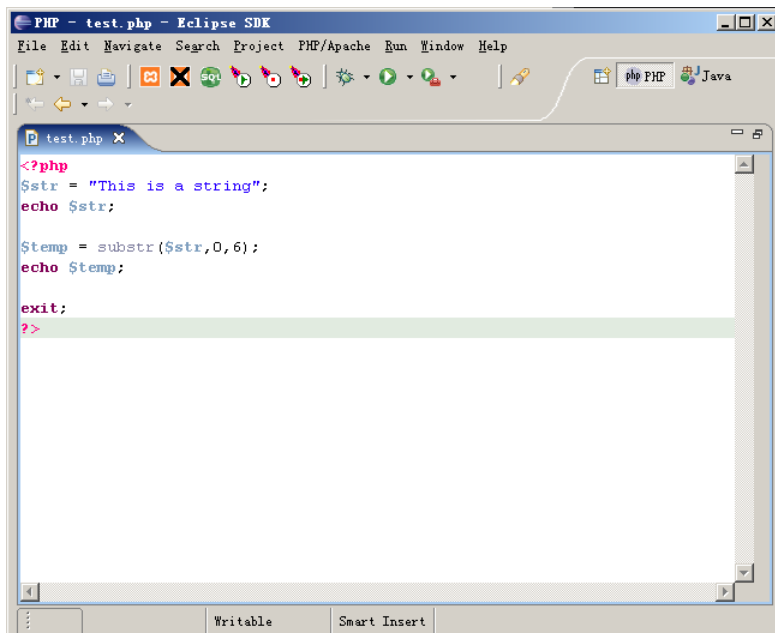


图 1.32 Eclipse 的编辑界面

3. Zend Stdio IDE 开发环境

Zend Stdio 是 PHP 的专业集成开发环境，它提供了 PHP 代码编辑、调试、分析、优化及数据库管理的全套工具，可以有效地提高 PHP 项目的开发效率。Zend Stdio 不仅专业而且易用，适合一般 PHP 程序编写和大型 PHP 项目开发。图 1.33 是 Zend Stdio 的编辑界面。

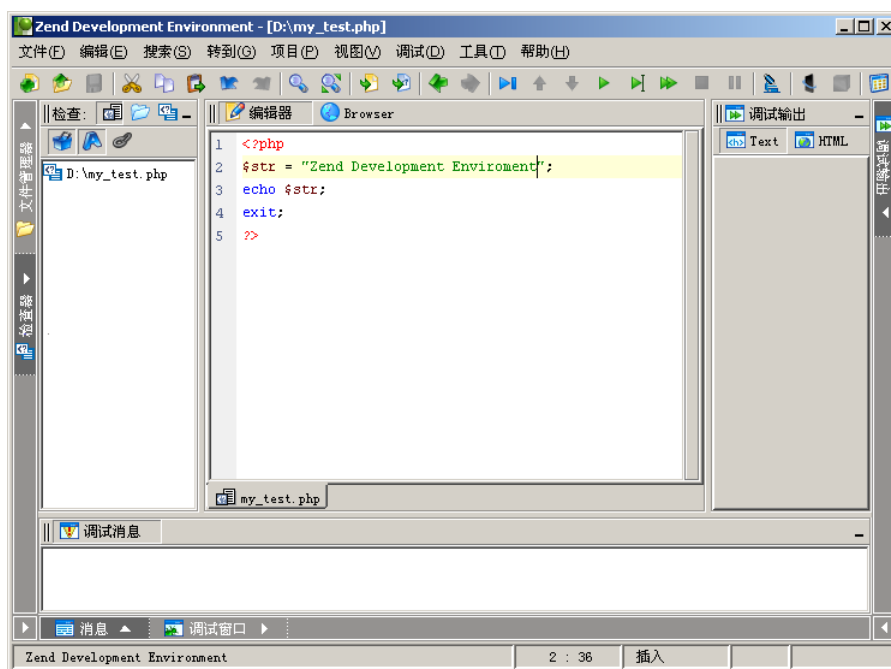


图 1.33 Zend Studio 的编辑界面

1.8 立竿见影

经过上几节的讲述，配置好了 PHP 的开发环境。从现在开始，通过一些小例子体验一下 PHP，看看 PHP 是怎样一种计算机程序语言，简单了解一下用 PHP 能做哪些事。从向 PHP 问好开始，向它说声“Hello,PHP!”！

1.8.1 编写第一个 PHP 程序——“Hello,PHP!”

打开最上手的编辑器，来编写第一个 PHP 程序。这个 PHP 程序非常简单，代码如下所示。

代码 1-34 第一个 PHP 程序 1-34.php

```
<?php  
echo "Hello,PHP!";  
?>
```

在编辑器键入以上代码后，按文件名 hello.php 保存在 Apache 安装目录的 htdocs 目录下。然后打开浏览器，键入地址：<http://localhost/hello.php>，如果一切正确无误的话，将会看到浏览器显示出“Hello,PHP!”字样，如图 1.34 所示。

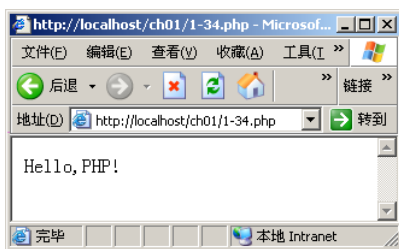


图 1.34 Hello,PHP!

下面对这个 PHP 程序做简单说明。所有的 PHP 代码都以 “<?php” 标识符起始，然后加入 PHP 语句，最后以标识符 “>” 结束。echo 是 PHP 的一个语句，用来输出一串字符，这个例子使用 echo 语句输出 “Hello,PHP!” 这个串字符。一般情况下在 HTML 文档中，要像上述这样加入 PHP 代码。需要注意的是，每个 PHP 语句都要以英文分号 “;” 结束，如果一条语句之后没有分号，那么 PHP 程序将会无法执行。

注意：虽然很多文本编辑器都可以用来编写 PHP 代码，但最终保存文件时，文件的后缀名应该为 .php。这样，当浏览器请求访问 PHP 文件时，服务器会找到该文件，并将其提供给 PHP 解释，最终由服务器将解释结果返回给浏览器。

1.8.2 使用 PHP 处理 HTML 表单

这一小节介绍如何使用 PHP 处理 HTML 表单数据。在这小节，将编写一个 PHP 程序，试着用它处理 HTML 文档提交的数据。假如有如图 1.35 所示的一个表单，其代码如 1-35 所示。

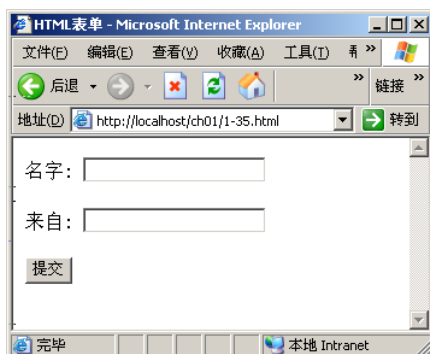


图 1.35 HTML 表单

代码 1-35 含有表单的 HTML 文档 1-35.html

```
<html>
<head><title>HTML 表单</title></head>
<body>

<form action="1-36.php" method="post">
<p>名字: <input type="text" name="user_name" /></p>
<p>来自: <input type="text" name="hometown" /></p>
<p><input type="submit" value="提交"/></p>
</form>

</body>
```

```
</html>
```

接下来，编写一个 PHP 程序，此程序将打印出由这个表单提交来的数据，如代码 1-36 所示。

代码 1-36 使用 PHP 处理表单数据 1-36.php

```
大家好，我是<?php echo $_POST['user_name'];?>!  
<?php echo "<br/>" ?>  
<?php echo "<br/>" ?>  
我来自<?php echo $_POST['hometown'];?>。
```

假如在上面的表单里，名字填写“木兰”，来自填写“中国”，如图 1.36 所示，按提交按钮后，1-36.php 将被调用，最后将看到如图 1.37 所示的效果。

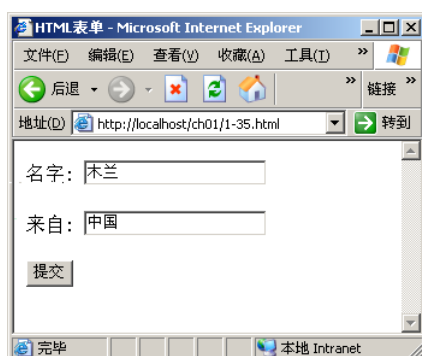


图 1.36 填写 HTML 表单

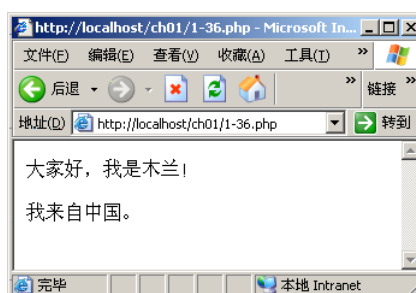


图 1.37 PHP 程序处理表单数据的结果

代码 1-36 所示的 PHP 程序并不复杂，其中使用了 PHP 的自动全局变量 `$_POST`，它包含了所有 POST 数据。所有用 POST 方法提交的表单数据，都将保存在这个全局变量中。上例中，通过 `$_POST['user_name']` 取得用户提交的名字，这里的 'user_name' 正是表单输入框 name 属性的值，最后通过 echo 语句，将用户的名字打印出来。

1.9 深入了解 Apache、PHP 和 MySQL

在正式学习 PHP 语言之前，还需要对 Apache、PHP 配置和 MySQL 的有关方面做一些比较深入的说明，这样即有助于读者对前几节知识加深认识，也有助于读者学习和理解后续内容。以下几小节的内容，主要将以 Windows 平台下的配置做讲解，Linux/UNIX 平台下的情形类似。

1.9.1 Apache 服务器目录

Apache 安装完成后，有一些目录需要读者进一步了解，如：conf 目录、htdocs 目录、logs 目录和 modules 目录。

- ❑ conf 目录：conf 目录下存放着一些 Apache 配置文件，其中最常用到的就是 httpd.conf，这是 Apache 的核心配置文件，Apache 服务器的很多重要配置及功能实现都要在这个文件里完成。这个文件也是 PHP 开发人员需要经常改动的文件，本书将在下一小结讲述 httpd.conf，设置一些重要的配置项。
- ❑ htdocs 目录：这个目录被 Apache 默认为服务器的根目录。这就是说，在默认情况下，开发人员

编写的 HTML 文档和 PHP 程序，只有放到这个目录下，才可以被访问或被执行。

- ❑ logs 目录：这个目录下存放着服务器级别的日志文件。如 access.log 记录用户访问的文件及其访问日期时间、方式等。这个目录下的有些文件，有时可以用来做 PHP 程序调试之用，因为服务器记录了些错误在这些日志里，开发人员可以通过这些错误，来调试 PHP 程序。
- ❑ modules 目录：这个目录下放有 Apache 执行的核心模块，当 Apache 启动时，它会根据配置从这个目录里载入需要的模块。一般情况下，PHP 开发人员不需要对这个目录了解更多。

1.9.2 进行基本的 Apache 配置

Apache 服务器的很多功能、任务等重要配置，都是通过修改 httpd.conf 来完成的。如设置服务器根目录、服务器超时时间、监听端口、Apache 运行模块的载入、服务器语言字符设置等。下面这段文字就是从 httpd.conf 摘出的一部分。

```
#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300
```

httpd.conf 中，凡是以“#”号开头的文字，表示注释，也就是说，这些行的内容只是对配置项的一些说明，Apache 启动时它们并不会被加载。上例中，有三行注释，而 Timeout 是一个配置项，300 是这个配置项的值，“Timeout 300”表示服务器的处理访问的超时时间为 300 秒。这小节，将讲述几项重要的 Apache 配置项。转到 Apache 安装目录下的 conf 目录，打开 httpd.conf。找到这几项内容：“Listen”、“LoadModule”、“DocumentRoot”、读者先有个大概的认识。

(1) Listen：这个配置项用来绑定 Apache 监听的 IP 地址或端口号，一般情况下，此配置项设置为 80，即 Apache 监听 80 端口，这也是 HTTP 协议所使用的端口。

(2) LoadModule：正如读者所看到的那样，在 httpd.conf 中有很多 LoadModule 项，每一个用来载入一个模块。如下所示。

```
LoadModule access_module modules/mod_access.so
```

表示载入 access_module 模块，载入路径为 modules/mod_access.so。并不是所有的模块都要从 modules 目录下载入，比如，将 PHP 作为一个 Apache 模块运行，就需要载入有关 PHP 的模块，在 httpd.conf 中加入如下所示的配置。

```
LoadModule php5_module d:/php/php5apache2.dll
```

这就是从 php 的安装目录载入由 Apache 执行的 PHP 模块 php5apache2.dll，本书例子将 PHP 安装在 D 盘，所以载入路径是 d:/php/php5apache2.dll。如果不需要使用某个模块，或者说关闭某个服务器功能，只需将这个模块的载入配置注释掉就可以了。如使服务器不支持 URL 重写机制，采取的设置可以如下所示。

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

(3) DocumentRoot：这个配置项用来设置服务器的根目录，默认设置为 Apache 安装目录下的 htdocs 目录。用户可以修改这个配置项的值，修改服务器根目录。比如将根目录设置为 D 盘的 webroot 目录，可以先注释掉默认配置，然后添加新的配置，如下所示。

```
#DocumentRoot "D:/Apache2/htdocs"
DocumentRoot "D:/webroot"
```

这样，对于所有 HTTP 请求，Apache 服务器会去 D:\webroot 目录下找客户端要访问的文件。

1.9.3 Apache 的启动与停止

对于 Windows 用户来说，可以通过 Windows “服务” 来启动和关闭 Apache 服务器。进入 Windows 控制面板里的管理工具，打开服务，找到 Apache 一项，通过“操作”菜单项或图标按钮即可完成 Apache 的启动与停止。如图 1.38 所示。

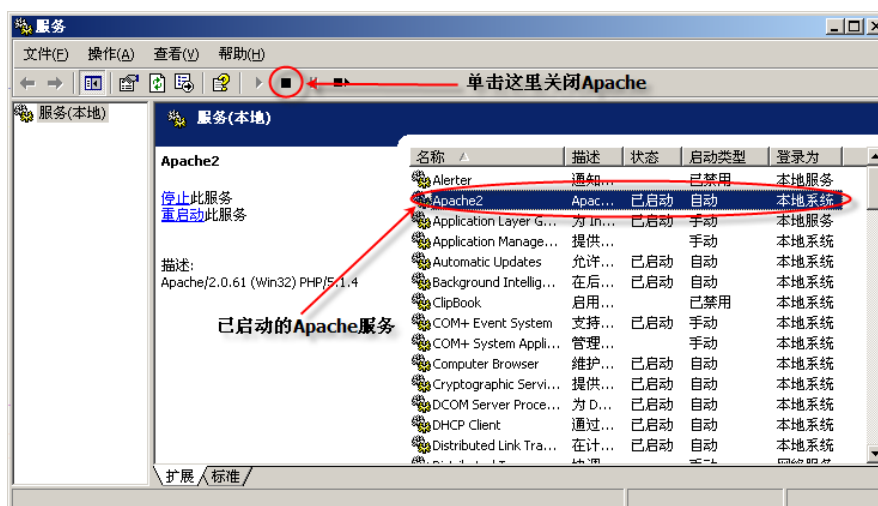


图 1.38 Apache 的关闭与启动

1.9.4 PHP 的核心配置——php.ini

与 Apache 相似，PHP 也有自己的核心配置文件 `php.ini`，PHP 分析器启动时会读取这个文件。与 `httpd.conf` 不同，`php.ini` 以英文分号 “;” 做为注释符。本节主要讲述以下 PHP 基本的配置项，这些配置项均在 `php.ini` 中设置。

- ❑ `doc_root`: 该项用来指定 PHP 页面在服务器的根目录，以本书安装情况为例，该项配置如下所示。

```
doc_root = d:\Apache2\htdocs
```

注意：`httpd.conf` 配置项与其值之间用空格分割，而 `php.ini` 使用 “=”。

- ❑ `extension_dir`: 该项用来指定 PHP 在哪个目录下查找扩展动态库。以本书安装情况为例，该项配置如下所示。表示从 PHP 安装目录下的 `ext` 目录里查找扩展动态库。

```
extension_dir = d:\php\ext
```

1.9.5 PHP 文件上传功能的配置

关于 PHP 文件上传功能的配置，主要有以下所示的两项。

- ❑ `upload_tmp_dir`: 当进行文件上传时，临时文件存放的目录，对于 Linux/UNIX 用户来说，要特别注意，当前 PHP 用户必须拥有这个目录的写权限。
- ❑ `upload_max_filesize`: 这个配置项指定了允许上传文件大小的最大值，`php.ini` 默认值是 2M。

1.9.6 PHP 中 session 的配置

在 php.ini 中，有关 session 的基本配置有以下所示的几项。

- ❑ session.save_handler: 这项配置用来设置 session 的存储方式，一般使用默认值 files 即可，代表用文件储存。
- ❑ session.save_path: 这项用来设置 session 的保存路径，以本书为例，将 session 保存在 PHP 安装目录的 session 目录下，如下所示。

```
session.save_path = d:\php\session
```

- ❑ session.use_cookies: sessionid 的传递方式，默认是 cookie，推荐使用。

1.9.7 PHP 中和电子邮件有关的配置

在这里 PHP 的邮件配置目前只需了解一个 sendmail_path 项即可，其他的配置项将在以后有更多讲述。

sendmail_path 这项仅针对 Linux/UNIX 用户来说，它用来指定 sendmail 程序的目录位置，通常会为 /usr/sbin/sendmail 或 /usr/lib/sendmail。

1.9.8 PHP 基本的安全设置

这里的安全设置主要是指 PHP 安全模式方面的内容。基本的配置有以下几项。

- ❑ safe_mode: 是否允许 PHP 的安全模式，默认情况此项配置的值为 Off，即关闭安全模式。
- ❑ safe_mode_exec_dir: 该项表示安全模式下，系统可执行系统程序的目录。这个配置项取决于上一项，如果 PHP 运行于安全模式下，一些系统函数将会拒绝执行不在该目录下的系统程序。

1.9.9 MySQL 数据库系统的启动与关闭

对于 Windows 用户，和 Apache 类似，可以通过 Windows 服务来关闭和启动 MySQL。进入 Windows 控制面板里的管理工具，打开服务，找到 MySQL 一项，通过“操作”菜单项或图标按钮即可完成 MySQL 的启动与停止，如图 1.39 所示。

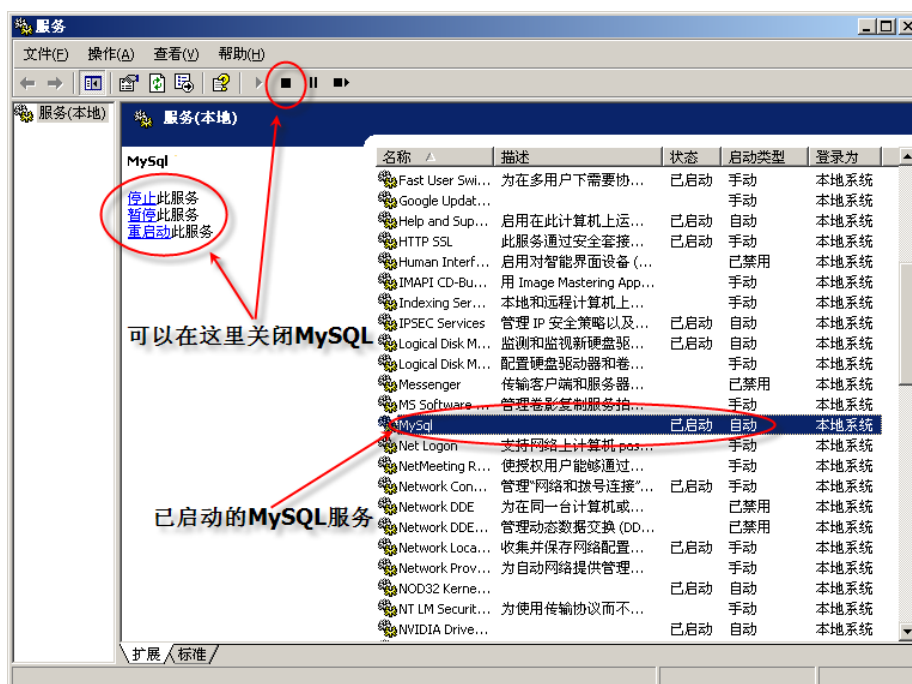


图 1.39 MySQL 的启动与关闭

1.9.10 MySQL 对数据的存储

首先了解一下 MySQL 的目录结构。Windows 平台下，MySQL 目录结果如图 1.40 所示。

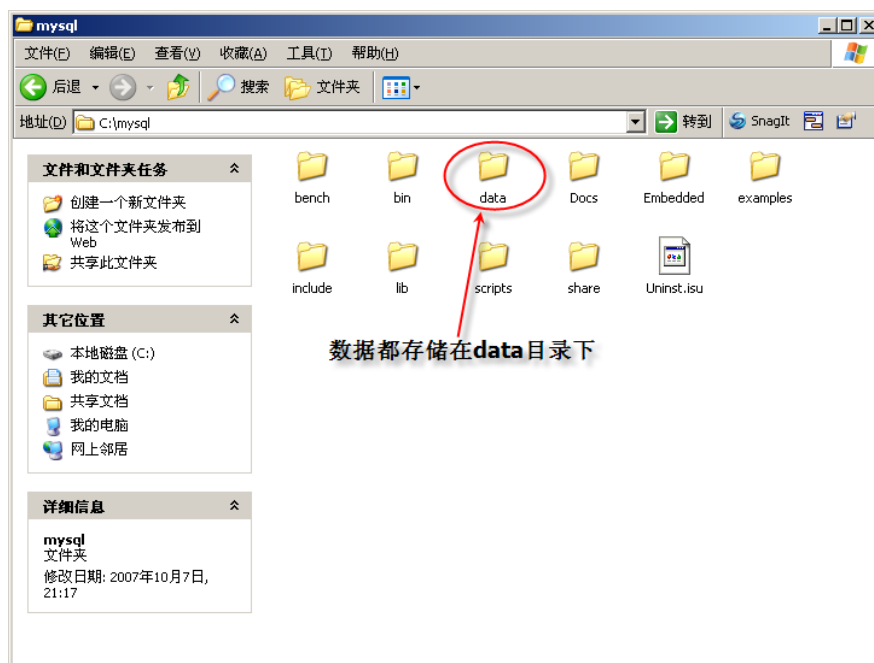


图 1.40 MySQL 的目录结构

MySQL 会把所有的数据以文件的形式存储在 MySQL 安装目录的 `data` 目录下。开发人员可以通过 MySQL 创建许多数据库，每创建一个数据库，就会在 `data` 目录下生成一个子目录，这个子目录的名字

就是数据库的名字，在这些子目录下存放的就是真正的数据库文件。关于 MySQL 更多的概念与内容，将在第 12 章详细讲述。

1.10 实例：使用 JavaScript 验证 HTML 表单数据

本章介绍的重点内容是 HTML、CSS、JavaScript 和 PHP 环境的搭建。其中 HTML、CSS 已经 JavaScript 的相关内容又是 Web 开发的最基础知识，掌握这方面的内容对 Web 开发至关重要。因为在 HTML、CSS 及 JavaScript 三者之间，JavaScript 是比较难掌握的，所以，本节以一个 JavaScript 的应用为实例，作为对 Web 开发基础知识的一个总结。

这个 JavaScript 实例是用来验证 HTML 表单数据的简单程序。任何一个 Web 应用，都离不开数据的提交和处理，这些数据要么被应用程序直接使用，要么被应用程序存入数据库。无论哪种情况，都应该保证浏览器端所提交数据的有效性和正确性。这就要求服务器端程序，在处理数据之前，先对 HTML 表单所提交的数据进行合法性验证，以保证应用程序执行正常，或者保证存入数据库的数据完整有效。

通常需要验证的项，包括数据长度是否满足、字符是否合乎需求（如只能为数字或只能为英文字母等），还有对一些特定格式的验证，如对 Email 地址的验证、URL 的验证、IP 地址的验证等等。验证可以在服务器端，由应用程序完成，或者在浏览器端，由 JavaScript 完成。本节将介绍后一种验证方式。

（1）下面就来建立一个 HTML 页面，该页面有一些表单元素组成，当这个表单被提交时，将由这个页面中内嵌的 JavaScript 程序完成对这些元素值的验证。HTML 页面如代码 1-37 所示。

代码 1-37 一个含有表单元素的 Web 页面 1-37.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>1-37.html - 使用 JavaScript 验证 HTML 表单数据</title>

<script language="JavaScript" src="1-38.js">
</script>

</head>
<body>
<h1>使用 JavaScript 验证 HTML 表单提交数据</h1>
<br/>
<br/>
<form id="fdata" name="fdata" method="post" action="" onsubmit="return checkForm()">
  <table width="476" height="154" border="1" align="center" cellpadding="1" cellspacing="0">
    <tr>
      <td width="254"><div align="right">请只输入数字（长度小于 10）：</div></td>
      <td width="212"><label>
        <input type="text" name="num" />
      </label></td>
    </tr>
    <tr>
```

```

        <td><div align="right">请只输入英文字母: </div></td>
        <td><label>
            <input type="text" name="a1" />
        </label></td>
    </tr>
    <tr>
        <td><div align="right">请做选择: </div></td>
        <td><label>
            <select name="lang">
                <option value="">请做选择</option>
                <option value="english">英语</option>
                <option value="french">法语</option>
                <option value="panish">西班牙语</option>
            </select>
        </label></td>
    </tr>
    <tr>
        <td height="38" colspan="2"><label>
            <input type="submit" name="Submit" value="提交" />
        </label></td>
    </tr>
</table>
</form>

</body>
</html>

```

代码 1-37 所示的 HTML 文档中, 定义了一个表单, 其中有两个文本框元素和一个下拉列表框元素。这个表单命名为 fdata, 如代码第 16 行 (name="fdata")。1-37.html 的第 8、9 两行, 引入了一个 JavaScript 文件 1-38.js, 验证该表单数据是否完整有效的 JavaScript 程序将在这个 JS 文件中完成。请特别注意 1-37.html 中如下所示的这行代码。

```
<form id="fdata" name="fdata" method="post" action="" onsubmit="return checkForm()">
```

这行代码使用了 onsubmit 事件, 这个事件触发时会调用 JavaScript 函数 checkForm(), 在函数 checkForm() 中实现对表单数据的验证, 本节稍后将完成这个 JavaScript 程序。浏览 1-37.html 将会看到如图 1.41 所示的效果。

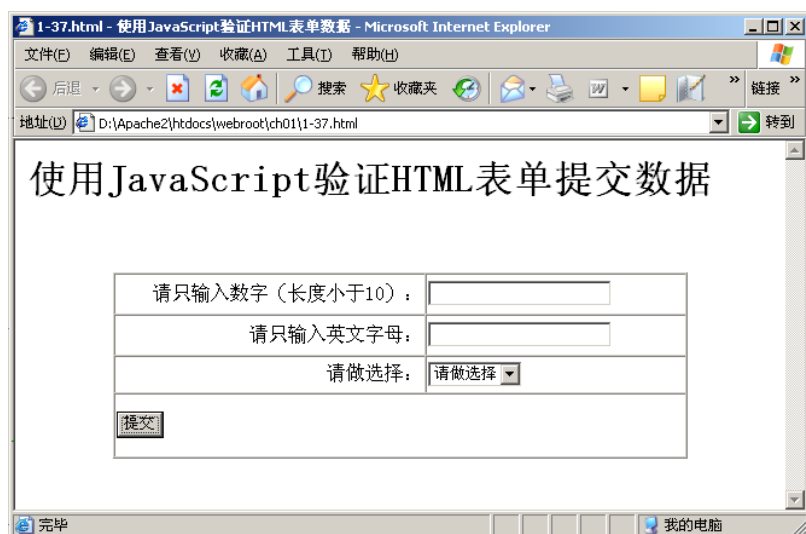


图 1.41 一个含有 HTML 表单的 Web 页面

(2) 接下来实现验证表单数据的 JavaScript 程序 `checkForm()`。为了使读者慢慢消化理解，先实现验证表单域不为空的部分。其代码如下所示。

```
function checkForm()
{
    if(document.fdata.num.value==""
    || document.fdata.al.value==""
    || document.fdata.lan.value=="")
    {
        alert("表单中的每一项都必须填写");
        return false;
    }
    else
        return true;
}
```

这段 JavaScript 代码通过 `document.fdata.num.value` 获得表单域各个元素的值，当它们之间任意一个为空时，都会弹出 JavaScript 警告对话框。将这段程序保存至 `1-38.js`，然后在 `1-37.html` 页面填写任何数据，也不做任何选择的情况下，按提交按钮，此时会调用函数 `checkForm()`，从而会看到如图 1.42 所示的效果。

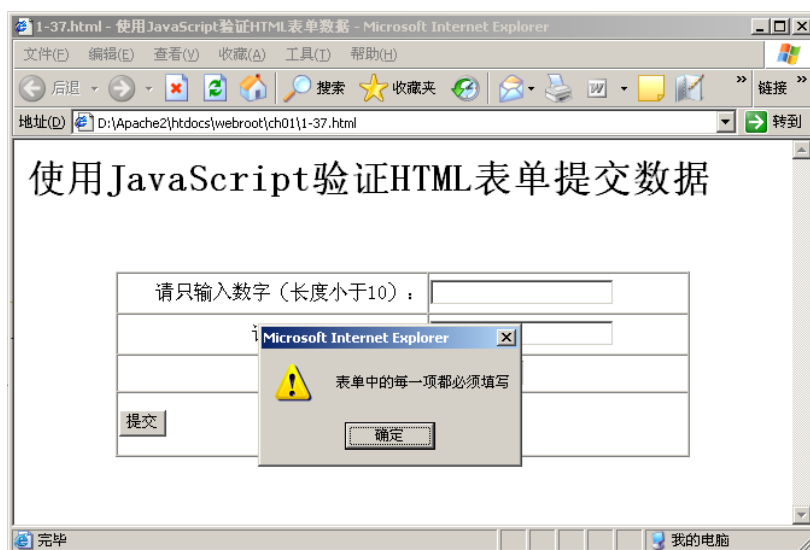


图 1.42 JavaScript 验证提交数据是否为空

(3) 下面要完成的是，判断提交数据只为数字或字母。如下所示的 JavaScript 代码是这两种判断的一个实现。

```
if(document.fdata.num.value!="")
{
    var s = document.fdata.num.value;
    for(var loc=0; loc<s.length; loc++)
    {
        if((s.charAt(loc) < '0') || (s.charAt(loc) > '9'))
        {
            alert("第 1 个文本框只能输入数字！");
            return false;
        }
    }
}

if(document.fdata.al.value!="")
{
    var t = document.fdata.al.value;
    for(var loc=0; loc<t.length; loc++)
    {
        if((t.charAt(loc) < 'a') || (t.charAt(loc) > 'z') || (t.charAt(loc) < 'A') || (t.charAt(loc) > 'Z'))
        {
            alert("第 2 个文本框只能输入英文字母！");
            return false;
        }
    }
}
```

判断是否为数字时，首先通过代码 `document.fdata.num.value` 获取第 1 个文本框的值，然后循环判断所输入值的每一位是否都是数字，如代码第 4 行到第 11 行所示。判断是否为英文字母的方法，与判断数字的类似。因为这个实例中，还要求输入的数字长度要小于 10，所以还需加一段代码，来判断数字字符串的长度是否小于 10，这个非常好实现，只需在上述代码判断数字的部分加入如下所示的一行代码

即可。

```
if(s.length > 9)
{
    alert("第 1 个文本框字符串长度要小于 10! ");
    return false;
}
```

(4) 最终如果用户填写的内容和所做的选择都正确, 就将用户填写的内容和所做选择用 JavaScript 弹出对话框返回给用户。完整的验证表单数据的函数 checkForm() 如代码 1-38 所示。

代码 1-38 验证表单数据的 JavaScript 程序 1-38.js

```
function checkForm()
{
    if(document.fdata.num.value==""
    || document.fdata.al.value==""
    || document.fdata.lan.value=="0"
    )
    {
        alert("表单中的每一项都必须填写! ");
        return false;
    }

    if(document.fdata.num.value!="")
    {
        var s = document.fdata.num.value;
        if(s.length > 9)
        {
            alert("第 1 个文本框字符串长度要小于 10! ");
            return false;
        }
        for(var loc=0; loc<s.length; loc++)
        {
            if((s.charAt(loc) < '0') || (s.charAt(loc) > '9'))
            {
                alert("第 1 个文本框只能输入数字! ");
                return false;
            }
        }
    }

    if(document.fdata.al.value!="")
    {
        var t = document.fdata.al.value;
        for(var loc=0; loc<t.length; loc++)
        {
            if(((t.charAt(loc) < 'a') || (t.charAt(loc) > 'z')) && ((t.charAt(loc) < 'A') || (t.charAt(loc) > 'Z'))))
            {
                alert("第 2 个文本框只能输入英文字母! ");
                return false;
            }
        }
    }
}
```



```
}  
}  
  
var v = document.fdata.lan.value;  
  
var str = "您的填写和选择为: \r\n" + s + "\r\n" + t + "\r\n" + v;  
alert (str);  
}
```

当表单数据填写符合要求，按提交按钮提交表单后，将看到如图 1.43 所示的效果。

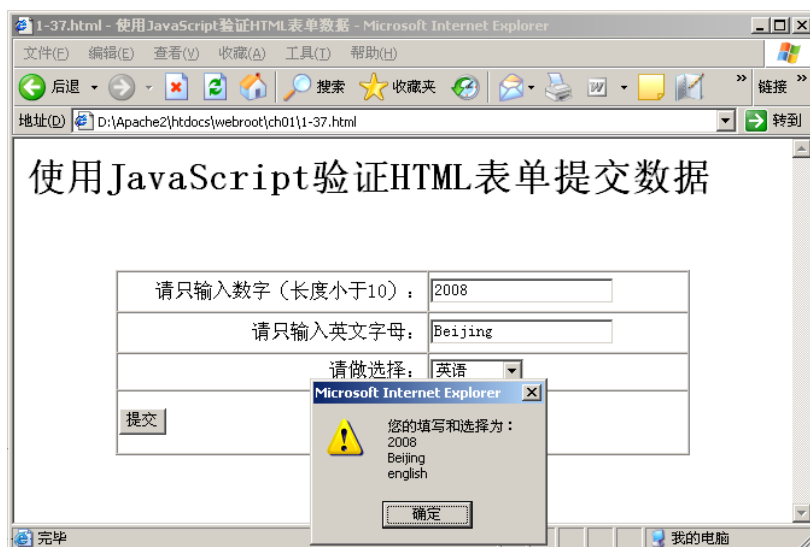


图 1.43 表单验证通过后的效果

1.11 小结

本章较为详细地介绍了 Web 编程的基础知识和 PHP 开发环境的配置。Web 编程基础部分详细讲述了 Web 基本体系结构、HTML 语言、CSS 和 JavaScript 基础。PHP 开发环境的配置讲述了 Apache、MySQL 和 PHP 的安装及基本配置，其中重点阐述了如何修改 httpd.conf、php.ini 等核心配置文件，及其之间的相互关系。最后通过一个实例验证了一个 PHP 的开发过程，读者也可以通过此例，验证机器的配置是否已经正常使用。